Doing enterprise architecture

Process and practice in the real enterprise

Tom Graves

Tetradian Enterprise Architecture Series

Doing enterprise architecture

Process and practice in the real enterprise

Last published on 2012-03-15

Contents

Doing enterprise architecture

Process and practice in the real enterprise

Tom Graves
Tetradian Consulting

Published by

Tetradian Books
Unit 215, Communications House
9 St Johns Street, Colchester, Essex CO2 7NN
England

http://www.tetradianbooks.com

First published April 2009

ISBN 978-1-906681-18-0 (paperback) ISBN 978-1-906681-19-7 (e-book)

Acknowledgements

Amongst others, the following people kindly provided comments and feedback on the early drafts of this book: Michael Ater (CH), Daljit Banger (GB), Stuart Curley (GB), Charles Edwards (GB), Geoff Elliott (GB), Nigel Green

(GB), Bas van Gils (NL), Ziggy Jaworowski (AU), Alexander Lorke (GB), Graham McLeod (ZA), Bob MacDonald (AU), Helen Mills (AU), Jos van Oosten (NL), Kim Parker (AU), Liz Poraj-Wilczynska (GB), Erik Proper (NL), Marlies van Steenbergen (NL), Kevin Smith (GB), Peter Tseglakof (AU), Jaco Vermeulen (GB).

Book contents

The complete contents of the book are as follows - chapters included in this sample are shown in *italics*:

Introduction

Doing enterprise architecture

Enterprise architecture is a relatively new discipline, though one that is rapidly becoming more important in business, especially in large multi-partner enterprises. But what do enterprise architects actually *do*? And what kind of business value can be assigned to the results of that work? To put it in the baldest business terms, what's the return on investment?

Enterprise architects manage a body of knowledge about enterprise structure and purpose. There are a fair few books available now about the principles and even some of the overall practices. (You'll find a selection of books and other key references on this listed in the *Resources* section at the end of this chapter.) But in most cases, that's about as far as they go. What this book does instead is focus on what is done in practice, in what order, and *why* it is done – the real business concerns that need to be addressed at each level of architecture maturity.

At present, enterprise architecture is often described solely as an aspect of IT, responsible for improving 'business/IT alignment'. But in practice, it can, and should, cover a much broader scope than just IT. Although it will often start out from there, it should eventually encompass the architecture of the entire enterprise. So the examples shown here are drawn not only from the usual information-centric industries such as finance, banking, insurance and the like, but also other industries in which IT is merely one of many different 'enablers' – such as in telecoms, logistics, utilities, manufacturing, government and much else besides.

Enterprise-architecture is a strategic skill that can be of real value to *every* enterprise, regardless of its size, its type, its industry or context. The same principles apply as much to a theatre, an engineering works, a retail floor, a chemical plant or a bank: it really doesn't matter. And in some aspects of enterprise architecture there may be no IT at all: as one commentator put it, an enterprise has an architecture even if it doesn't have electricity.

Whatever your enterprise, this book will show you what to do, when and how and why, to make it all work in practice.

Who should read this book?

This book is intended for enterprise architects, strategists and any others tasked with understanding the enterprise as a whole; and for programme and portfolio managers and others who guide the changes to business practice that arise from that work.

Enterprise-architecture provides a 'big-picture' overview that outlines the business context for other architecture disciplines: so this book would also be useful for business architects, process architects, security architects, solution architects, software architects and the like.

And executives, service-managers, process-managers and many others may find this book valuable as a guide to what enterprise architects do, and why they do what they do, in support of the overall enterprise.

What’s in this book?

There are three main parts to this book. The first deals with what’s involved in setting up the architecture capability, and getting down to work – see chapter *Preparing for architecture*. Next, we explore the typical kinds of work, and the business concerns they each address, at each stage of enterprise-architecture maturity – see chapter *A matter of maturity*. Finally, we’ll explore what you’d do to keep on enhancing the depth and richness and value of the architecture capability and its artefacts – see chapter *What next?*.

Each section includes lists of questions to guide the work. These lists are fairly comprehensive, but obviously cannot cover everything – given the vast range of possible circumstances in which they may be used – and in some cases the questions themselves must necessarily be somewhat generic. So whilst you should find all the essentials here, do expect to do some translation and adaptation to suit your own specific context.

What’s *not* in this book? In essence, anything that is context-specific, or is easily available elsewhere. For example, you’ll need a suitable architecture-development method: for that, you might turn to the current *de facto* standard, TOGAFâ□˘ Version 9 – all 780 pages of it, for which obviously there's no point in repeating here.

> Since TOGAF is designed only for enterprise IT-architecture, you may want to supplement it with *Bridging the Silos: enterprise architecture for IT-architects*, another book in this series, which shows how to adapt TOGAF to the real whole-of-enterprise scope.

You'll need a suitable reference-framework, or set of frameworks – though these should be chosen from examples tailored to your specific industry, such as eTOM for telecoms, SCOR for logistics or FEAF PRM for government. And you'll need some kind of formal governance-framework, to guide architecture development and its relationship with change-management: but again, this is described in part in the TOGAF specification, and will need to be linked to whatever techniques you already use to govern enterprise change. The same applies to metrics, to detailed definitions of skillsets, and so on: the extra information you'll need there will depend on your context, but you should be able to find that easily enough from your industry's existing sources.

This book deals with the one specific practical concern: what do you *do* when doing enterprise architecture? Our aim has been to keep this book to a practical size that you can keep beside you as you work. So to that end, there are plenty of cross-references in the *Resources* sections at the end of each chapter: use them to fill in any gaps and keep you on track for your own business context.

Resources

The Tetradian Enterprise Architecture Series

- Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008) – describes a high-level framework and method for whole-of-enterprise architecture

- Tom Graves, *Bridging the Silos: enterprise architecture for IT-architects* (Tetradian, 2008) – describes how to adapt and extend IT-architecture *de facto* standards such as Zachman, TOGAF, FEAF, ITIL and PRINCE2 for use at a whole-of-enterprise scope

- Tom Graves, *SEMPER and SCORE: enhancing enterprise effectiveness* (Tetradian, 2008) – describes a suite of tools and techniques to enhance enterprise effectiveness, such as the SCORE extension to SWOT strategy-assessment, and the SEMPER diagnostic and metric for enterprise effectiveness

- Tom Graves, *Power and Response-ability: the human side of systems* (Tetradian, 2008) – describes the business implications of the dichotomy that whilst the physics definition of power is 'the ability to do work', most social definitions are closer to the ability to avoid it

- Tom Graves, *The Service Oriented Enterprise: enterprise architecture and viable systems* (Tetradian, 2009) – describes how to extend the principle of service-oriented architecture to the design and structure of the entire enterprise

Other resources

- The Open Group, *TOGAFâ˘ Version 9* (Van Haren, 2009)
- Online version of TOGAF 9: see www.opengroup.org/architecture/tog
doc/arch/[1]
- FEAF (US Federal Enterprise Architecture Framework): see www.gao.gov/special.pubs/eaguide.pdf[2] [PDF]
- Zachman framework: see www.zifa.com[3]
- ITIL (Information Technology Infrastructure Library): see www.itil.org.uk[4] and www.itil-officialsite.com [5]

[1]http://www.opengroup.org/architecture/togaf9-doc/arch/
[2]http://www.gao.gov/special.pubs/eaguide.pdf
[3]http://www.zifa.com
[4]http://www.itil.org.uk
[5]http://www.itil-officialsite.com

Preparing for architecture

What *is* 'enterprise architecture'? It seems traditional to start off a book of this kind with a set of definitions, but here we'd perhaps be better served by a more discursive exploration, because it's surprisingly hard to find anything that's definite and definitive about any aspect of this discipline…

First, perhaps, the 'architecture' part. Definitions from the software industry, from the 1960s onward, tend to emphasise the need for *structure*, a consistent description and consistent set of relationships. Elsewhere, especially from business-architecture, there's more of an emphasis on business *purpose*, on devolving outward from strategy. Somewhere between the two, these need to meld into a mutual balance, about where structure meets with purpose, and purpose is expressed in structure. Hence that earlier description:

- **Enterprise architects manage a body of knowledge about enterprise structure and purpose**

'Enterprise' is perhaps harder to describe. The usual definition is that it's some kind of group who "coordinate their functions and share information in support of a common mission or set of related missions". The catch is that 'enterprise' is *not* the same as 'organisation' – or rather, that a formal organisation such as a company or government department is just one special case of an enterprise.

> Hence perhaps *the* key distinction between business architecture and enterprise architecture: the former would formulate their strategies and suchlike from the organisation's point of view, whilst the latter must also look at least one step above and pay detailed attention to the common enterprise that is shared by all stakeholders.

So unlike business-architects, who maintain focus on the strategic concerns of the company, enterprise architects must be able to work at any level, at any scope. Typically, they will deal more with overview than with detail, though early-maturity enterprise-architecture will often hold a firm emphasis on the finer details of IT-system design.

In essence, though, it's not just about improvements in IT-systems, or in the infamous and often troublesome 'business/IT alignment', but about creating a better understanding of how everything in the enterprise works with everything else – IT, people, machines, strategies, tactics, processes, products, services, *everything.* And it's not just about improved efficiency, but improved *effectiveness* – 'efficient on purpose'.

So where does this all fit, in organisational terms? The short answer is "it all depends", because it'll change with the nature of the enterprise, and usually with the level of architecture maturity. In the early stages it'll often be included within a project, or be a project in its own right; but as maturity develops, it'll need to be set up as a continuing capability, often acting as a bridge between strategy on one side and change-management on the other:

- **strategy specifies requirements for change;**
- **enterprise-architecture identifies capabilities for change;**
- **change-management guides change in the enterprise.**

The common view of enterprise architecture, as espoused in the major standards such as FEAF and TOGAF, is that its role is to define the 'future state' of the structure of the organisation's IT. Amazingly, that assertion manages to be misleading in almost every possible way: it's not just about the organisation, it's not just about the IT, there is no 'the' – there's always '*an* architecture', but never '*the* architecture' – and there is no such thing as a 'future state', because everything is changing dynamically, all the time.

- **The world is not static: there is no state.**

Those standard models then manage to compound the problems by insisting on a 'big-bang' approach to architecture development: we supposedly define 'the future architecture for everything' in one big project, and then it's finished forever. Which simply does not work for a real enterprise architecture: it'll be hopelessly out of date before we're even halfway through the process. It might sort-of work for a small subset of a technology architecture that'll be changed anyway as soon as the next major project comes along, but there's no way to make it work at the scale of a whole enterprise. We *must* think in terms of Agile, of an iterative approach to architecture development, right from the start.

> To be fair, TOGAF's designers do insist that its ADM (Architecture Development Method) could and should be used in an iterative style. Unfortunately, as anyone who's used it in practice will soon discover, TOGAF's predefined scope and structure will fight against this, every inch of the way. In effect it's a classic Waterfall approach with various bits of Agile vaguely grafted on, but without strict Agile-style governance, and in a way which somehow manages to combine the disadvantages of both without the advantages of either. And in my experience, the attempts to patch the problems in the recent TOGAF 9 upgrade have managed only to make it worse. Oh well.
>
> There are a few enterprise contexts for which a Waterfall approach to architecture would be required: it's all but mandatory in a military

> environment, for example. But for anything else, the sheer scale of the problems usually means that we *must* use an Agile approach to make it work: and we need to do that Agile properly, too.

But there's a catch here. Without architecture, a framework of context in which to place each iteration, Agile is little more than an undisciplined messing-about: so it seems we'd need an architecture already in place before we can do Agile architecture...

The way out of this dilemma is to cheat: we *invent* an architecture – almost anything that seems appropriate, really – to an act as an initial framework. (More on that when we look at planning and frameworks) below.) We then refine that framework with each subsequent iteration, steadily filling in more and more detail as we go.

What this is really about is a subtle shift in perspective. The classic Waterfall approach is like a photograph: we don't get the full picture – or, in this context, much if any business value – unless we can see the whole thing in one go. If we cut up a photograph, each small piece contains the full detail of that one area, but gives us no indication of how it fits in with anything else. So until the whole thing – the whole architecture – *is* in place, we're stuck.

By contrast, working with Agile is more like a holograph: we're *always* dealing with the whole picture, right from the start. Everything, however small, is always connected to everything else; and if we cut up a holograph, each small piece always contains a sense of the whole as a whole – the detail will be less, but not the scope. So when we do this right, every project not only delivers its own business value, but also contributes to the business value of everything else – *if* we use the framework holographically.

And we do that by treating every iteration, whatever its scope, in exactly the same way. *Nothing* can be considered 'special and different': so for an Agile architecture, the classic IT-centric style of so-called 'enterprise architecture', with its crass definition of 'business architecture' as "everything not-IT that might impact on IT", would be a guaranteed recipe for failure. To make it work, we have to dump the idea that IT is a special case that somehow *must* be the centre of every enterprise architecture. Instead, we need to view it as just one example amongst all of the business' enablers – a service *to* the business, not the other way round! Which might be a bit uncomfortable for some IT-architects to accept, perhaps: but that's just too bad, bluntly. The business *always* comes first.

Purpose – strategy

All activities in enterprise architecture should begin and end with an explicit business purpose. From the business perspective, architecture is a strategic tool, to guide the implementation of business strategy: so strategy and purpose must always be at its core.

> We need to remember, though, that strategy must always be driven by business need – *not* by IT hype! The fabled 'IT/business alignment' – or, more often, the lack of it – is littered with countless examples of disastrous attempts by consultants, vendors and even IT staff to foist overblown 'solutions' onto a justifiably unwilling business. Business Process Re-engineering (BPR) is perhaps the best known of these, but right now cloud-computing is shaping up nicely as the probable next candidate for the 'Expensive Failure Of The Year Award'…
>
> There's nothing wrong with BPR or cloud-computing as such: they can be very good solutions indeed, *in the right contexts*. Where it all goes wrong is when the 'solution' comes before the strategy, or when it's sold for the wrong reasons – such as the classic IT obsessions with cost-cutting or 'efficiency' for its own sake, without awareness of the broader implications or broader environment. A properly-thought-out, properly-explored strategy must always come before any suggestion of 'solutions': the business and its business-strategy *always* come first.

The real driver for any strategy is enhanced *effectiveness* in the context of continual change. And efficiency is only one of several interweaving strands of effectiveness:

- *efficient* – makes the best use of available resources
- *reliable* – can be relied upon to deliver the required results
- *elegant* – supports the human factors in the context; also 'elegant' in the scientific sense, in that clarity and the like will support structural simplicity and re-use
- *appropriate* – supports and sustains the overall purpose of the enterprise
- *integrated* – everything is linked to and supports the integration of the whole *as* whole

Strategically speaking, it's essential to keep all these strands in balance with each other. If we pay too much attention to any one strand such as efficiency – especially if it's at the expense of the others – we end up with a structure and strategy that might seem to perform well for a while, but is doomed for disaster further down the track. And again, we'll see all too many examples of this in the business press or elsewhere. Whether the strategy is in response to a new opportunity, a change in regulation, a new technology or a new market, the same always applies: we'll need to assess the issue's impact on the *whole* of the enterprise, not just on some selected subsection such as IT. Overall effectiveness *matters.*

> There's more on those effectiveness-themes in two other books in this series, *Real Enterprise Architecture: beyond IT to the whole enterprise* and *SEMPER & SCORE: enhancing enterprise effectiveness* – more details on those in the *Resources* section at the end of the chapter.

The effective purpose of what we do in enterprise architecture will necessarily change at different levels of architecture maturity. For example, there's little point in even *trying* to use architecture to assist in strategy until we have some idea of what business we're in – see chapter *Step 1: Know your business*) – or made a solid start on cleaning up the chaos that will have arisen naturally from too many mergers and acquisitions, or from too many years without some kind of framework to guide consistency in business systems – see chapter *Step 2: Clean up the mess*).

More accurately, we don't so much change what we do in architecture as extend it. Each maturity-level builds on those before, but we don't stop doing the work from those previous levels: whatever maturity-level we've reached at, we'll always need to keep track of what business we're in! The same goes for all the subsequent work on system consistency and the like: if we don't stay aware of all the infrastructure changes – new systems coming on-line, old systems reaching their sunset and being retired – we'll risk falling back into the same mess as before.

Strategy on its own is not enough: we also need some consistent means to make it happen, and keep on happening, in the real world of everyday business practice. So for architecture, we need clear metrics, to keep track of what's happening; we need consistent methods, so that we're clear about what needs to happen next; we need a consistent set of frameworks, so we can make sense of what we have and what we're planning to do. But perhaps most all, we need the right skills and structure to make it happen: the right people, and the right kind of governance. Hence it's those issues that we need to turn to next.

People – governance

Strategy describes the 'why' of the business; methods describe the 'how' of what needs to happen; frameworks describe the 'what'; and governance brings us back to 'why' again, by anchoring it in *people* – their choices, their actions and responsibilities.

It's always about people. No matter how technical a problem may seem, ultimately it's always a 'people-problem' – more accurately, it's the skills and commitment and drive of individual people that provide us with the means to solve any given problem.

Where governance comes in is that it's the way by which we manage the people-side of effectiveness – making sure that things happen 'on purpose', in the right order, for the right reasons, and so forth. Formal structures such as ITIL and COBIT and PRINCE2 do this for the IT context in general, for concerns such as service-management and project-management: we now need to do the same for enterprise architecture. Perhaps the key complication is that some of the governance will change quite radically as we move up the maturity scale, but we could summarise the core themes as follows:

- it's not a project – it's a continuing process
- you'll need different skillsets at different maturities
- you'll need senior support – eventually, from the executive and above
- you'll need to identify and engage with a wide range of stakeholders
- it's really about creating an ongoing dialogue about architecture

We'll expand on each of these themes as we explore the different maturity-levels later; for now, though, these are some of the points we'll need to understand before we start.

It's not a project: Enterprise-architecture often starts as a project, or in a project, and ultimately needs to be applied to every project, but it isn't a project in itself. It's a capability whose task is to manage a body of knowledge about structure and purpose – hence, unlike a project, it's not something that we do once and then quietly forget about when it's done. To retain its business value, we need to keep it going, keep it growing, keep putting it to practical use. It's not a project.

You'll need a range of skillsets: At present, enterprise architecture is often described solely as if it's some aspect of IT, and that IT technical skills are the only ones really required. Even if this were true, we would still need to cover the whole scope of 'IT': not just the obvious skills such as data architecture, applications architecture, security architecture, infrastructure architecture, network architecture, service architecture and the like, but building layout, cooling systems, power infrastructures and a whole lot more.

Once we start to move beyond IT to include the rest of the enterprise – especially in industries which are not information-centric, which is true for most – then the required skillsets could be *anything*. At the very least, by the time we reach the second or third maturity-level, we're going to need much more awareness of the business *as* business – which means skillsets in business architecture, organisational architecture, process architecture and so on. So we'll need to plan for that from the start – including all the people-issues of how to cope with those shifts in skillsets, and in the make-up of the enterprise-architecture team.

You'll need senior support: Because enterprise architecture isn't a project, it'll need funding and other resources to keep it going, as a kind of conceptual infrastructure for the enterprise. That can be a whole lot harder than funding a once-off project – but you won't get any real value from enterprise architecture unless you do this.

And as the maturity expands, so does the scope that you'll need to cover. You may well start out as a small 'skunk-works' project tucked away in IT, but by the time you really get going you'll need the authority to touch anywhere in the enterprise, bridge between any of the organisational silos, and ask often awkward questions of just about anyone. To work at that level, you'll need the full weight of the entire executive behind you.

True, you may not need all of that right from the start – though nice to have, of course! – but you'd better plan for it right from the start. Which means you'll need to be able to prove your business value right from the start, too.

You'll need to engage with stakeholders: As the scope of the architecture extends, so too does the range of stakeholders with whom you'll need to engage. Every object, every data-entity, business-rule, business-process and everything else will have a nominal owner – the 'responsible person' – and others who will have a business stake in the use and maintenance of whatever-it-is. And every one of them will want a say in what happens, or at least be kept informed on any possible changes. That's a lot of people – and a lot of resistance to change if you *don't* engage them in the architecture. There's a key point that's worth emphasising here:

- **People don't resist change: they resist *being* changed.**

They'll resist change if they don't see the point in it – in other words, if they suspect it's 'change for change's sake'

– and they'll certainly resist it if they think it's solely for someone else's benefit at their own expense. So you need to engage them in the architecture – engage them in *co-creating* enterprise change. If you don't, what you'll get is resistance – lots of it. Your choice...

> Hence the importance of what TOGAF 9 calls a Communication Plan. But even that is nothing like enough: it's not a one-way message to be broadcast after the event. It needs to be a full two-way communication with everyone – a dialogue of equals – that starts from Day One.

It's really about dialogue: Architecture isn't something we can control. It's all too big for that – especially at the scale of an entire enterprise. The only way we'll get it to work is if we share out the load, and preferably amongst everyone in the entire enterprise.

In essence, architecture is just an *idea*: the belief and experience that things work better when they're linked together into a unified whole. Things work better for *everyone* when that happens. But trying to do that by imposing a fixed system of order will only work when the world is static – which the real world isn't. And every small change everywhere impacts on the whole – so we do need to get those changes to work *with* all other parts of the whole. The way we do that is through architecture: more specifically, a continual *dialogue* about architecture.

> It doesn't take a large number of people to guide this dialogue, though. At one of our clients, a nationwide logistics organisation with some 35000 employees, the core enterprise architecture team consists of just five people – and even they also have other duties outside of architecture itself.
>
> It's much the same with other 'pervasive' themes such as knowledge-management, quality, security, privacy, health and safety: the work is distributed throughout the enterprise, but the core team that 'hold the faith', so to speak, need only be a small handful of specialists.
>
> Or generalists, more accurately – people who link things together across many different domains. In architecture, the specialist domain-architects – data, security, applications, process, service, strategy, infrastructure and so on – are more likely to be attached to project-teams, guiding the detail-concerns of individual change-projects. It's the generalists back at the core who keep the dialogue going, to help hold everything together. And we don't need many people for that: just a core framework in which people can come and go as needed.

Architecture is also a story, about possibility, and about problems overcome. If we try to force-fit others into someone else’s story, it’s unlikely they’ll be interested in such a predefined part; but if we engage them in *co-creating* the story, they’re much more likely to be willing to play. In that sense, engagement *matters*; the story needs to make sense, in a dynamic, personal, visceral way.

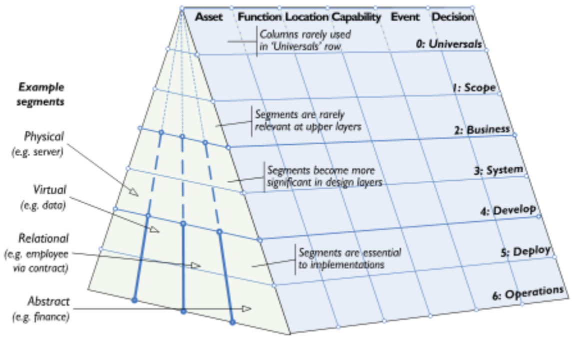
Another key complication is that people know more than they can say, and can say more than they can write down. Often the story develops through action and emotion as much as it does through ideas and plans. So the architecture story needs to encompass all those dimensions too – and likewise the framework on which we build and describe and make sense of that story.

Planning – frameworks

In principle, we could describe relationships between everything in the enterprise in terms of a single sentence-structure:

- “**with *«asset»* do *«function»* at *«location»* using *«capability»* on *«event»* because *«reason»***”

This comes from a revision of the well-known Zachman framework. It cleans up the taxonomy, and extends the original with an extra row at the top for enterprise core-constants or ‘universals’, and an extra dimension to clarify scope and implementations:


Asset
Function
Location
Capability
Event
Decision
Columns rarely used in 'Universals' row
0: Universals
1: Scope
2: Business
3: System
4: Develop
5: Deploy
6: Operations
Segments are rarely relevant at upper layers
Segments become more significant in design layers
Segments are essential to implementations
Example segments
Physical (e.g. server)
Virtual (e.g. data)
Relational (e.g. employee via contract)
Abstract (e.g. finance)

Framework rows, columns and segments

> This is described in more detail in the 'Framework' chapters in another book in this series, *Bridging the Silos: enterprise architecture for IT-architects* – see the *Resources* section below.
>
> See also tetradianbooks.com/ebook/silos_real-ea-frame-ref.pdf[6] [PDF] for a two-page summary of the framework and its core principles.

For the vertical dimension of the framework, we partition scope in terms of timescale – a set of seven distinct layers or perspectives, from unchanging constants, to items which change moment by moment. Each row adds another concern or attribute:

[6]http://tetradianbooks.com/ebook/silos_real-ea-frame-ref.pdf

- *Row 0*: '**Universals**' – in principle, should never change: core constants to which everything should align – identifies the overall region of interest and the key points of connection shared with enterprise partners and other stakeholders; added for compatibility with ISO-9000 etc

- *Row 1*: '**Scope**' (*Zachman*: 'Planner') – adds possibility of change: core entities in each category, in list form, *without* relationships - the key 'items of interest' for the enterprise

- *Row 2*: '**Business**' (*Zachman*: 'Owner') – adds relationships and dependencies between entities: core entities described in summary-form for business-metrics, including relationships between entities both of the same type ('primitives') and of different types ('composites')

- *Row 3*: '**System**' (*Zachman*: 'Designer') – adds attributes to abstract 'logical' entities: entities expanded out into *implementation-independent* designs - includes descriptive attributes

- *Row 4*: '**Develop**' (*Zachman*: 'Builder') – adds details for real-world 'physical' design: entities and attributes expanded out into *implementation-dependent* designs, including additional patterns such as cross-reference tables for 'many-to-many' data-relationships

- *Row 5*: '**Deploy**' (*Zachman*: 'Sub-contractor' or 'Out of Scope') – adds details of intended future deploy-ment: implementation of designs into actual software, actual business-processes, work-instructions, hardware, networks etc

- *Row 6*: ‘**Operations**’ (*Zachman*: implied but not described) – adds details of actual usage: specific instances of entities, processes etc, as created, modified, and acted on in real-time operations

The rows also represent different categories of responsibilities or stakeholders, such as senior management responsible for row-0 universals, or strategists at row-1 and -2, architects and solution-designers at row-3 and -4, and line managers and front-line staff at row-5 and -6. In effect, this is the same layering that we see in the management-hierarchy, and in the nesting of abstract services.

> Strictly speaking, the row-0 ‘Universals’ are more like another dimension or backplane, because *everything* in the enterprise needs to link back to them. Since we already have three dimensions – rows, columns and segments – it’s simpler to show it as an extra row at the top, and it’s easier for metamodel repositories to implement it that way, too. But it’s important to remember, though, that it *is* in effect another framework-dimension in its own right.

Below the core ‘Universals’, the framework splits horizontally into columns for six distinct categories of primitives, approximating to Zachman's what, how, where, who, when and why:

- *What*: **assets** of any kind – physical objects, data, links to people, morale, finances, etc
- *How*: **functions** – activities or services to create change, described independently from the agent (machine, software, person etc) that carries out that activity
- *Where*: **locations** – in physical space (geography etc), virtual space (IP nodes, http addresses etc), relational space (social networks etc), time and suchlike
- *Who*: **capabilities** clustered as **roles** or 'agents' – may be human, machine, software application, etc, and individual or collective
- *When*: **events** and relationships between those events – may be in time, or physical, virtual, human, business-rule trigger or other event
- *Why*: **decisions**, reasons, constraints and other tests which trigger or validate the condition for the 'reason' and the like, as in strategy, policy, business-requirements, business-rules, regulations etc.

In the lower layers of abstraction - closer to the real-world – we also need to split the columns themselves by context into distinct segments or sub-categories. Whilst these could be cut multiple ways, a typical set of segments would be:

- **physical**: tangible objects ('asset'), mechanical processes and functions, physical or temporal locations, physical events; also align to ***rule-based*** skills ('capability') and decisions
- **virtual**: intangible objects such as data ('asset'), software processes and functions, logical locations, data-driven events; also align to ***analytic*** skills ('capability') and decisions
- **relational**: links to people (as indirect 'asset'), manual processes and functions, social/relational locations, human events; also align to ***heuristic*** skills ('capability') and decisions
- **aspirational**: principles and values, brands and belonging, morale and self-belief ('asset'), value-webs and dependencies ('location'), business-rules ('event'); also align with ***principle-based*** skills ('capability') and decisions
- **abstract**: additional uncategorised segments such as financial ('asset', 'function'), energy ('asset'), time (as 'event'-trigger) etc

These segments are fundamentally different in scope and function:

- *Physical assets* are 'alienable' – if I give it to you, I no longer have it.

- *Virtual assets* are 'non-alienable' – if I give it to you, I also still have it.
- *Relational assets* exist between two entities (usually but not necessarily real people); each is one hundred percent the responsibility of *both* parties – if either party drops it, the relationship ceases to exist – and is actually harder to transfer to someone else than it is to create from scratch.
- *Aspirational assets* have some similarities with relational ones, except that the relationship is more a one-sided 'to' rather than a balanced 'between'.

The enterprise will need to be especially careful how it handles aspirational assets, for example, because they're closely linked to the sense of identity and self, and the passions embedded within them can be extremely destructive if not treated with respect.

> Hence woe betide any company that trashes a much-loved brand, for example, because the intensity of that aspirational commitment can unleash a wave of anger and retribution that resembles the inconsolable rage of grief… It's not 'rational' in any normal sense of the word, but it's definitely real, and does need to be treated as such.
>
> Closer to the opposite extreme, the loss of morale and drive that accompanies 'change-

> fatigue' has its roots in damage to aspirational assets: specifically, the loss of that sense of belonging, of being part of a known enterprise – an included member of a group who "coordinate their functions and share information in support of a common mission" and so on.
>
> This is crucially important in the architecture of the enterprise, because it's *the* key source of personal drive and commitment to the enterprise. If it's lost, the most we'll get from people is 'presenteeism' – their bodies may be present, but probably not much else. So whilst these can often seem subtle or strange, aspirational assets *matter*.

'Composites' of related entities and entity-types may exist within the same segment in a column, or in different segments of the same column. We can then link these composites across other columns to create other 'incomplete' design-patterns – the kind of structures we usually see in architecture models – or complete the composite across all of the columns for final implementation.

Understanding these asset-categories and the composites created from them helps to describe concerns such as storage and security. For example, a paper form is both a physical asset *and* a virtual asset: we need to manage it as a physical object, with all of the resultant issues around procurement, inventory, pre-use warehousing, modification and use, storage and disposal; yet we *also* need to manage it as a virtual asset, with all those concerns about data-quality, information-quality and so on. And we can describe some of the security-concerns by asset-category as follows:

- *physical security*: protect alienable asset against physical loss
- *virtual security*: protect non-alienable asset against loss via physical means (e.g. theft of laptop, damage to data-server), against inadvertent virtual loss (e.g. deletion, overwrite, out-of-date file-format) and against inadvertent or unauthorised replication
- *relational security*: protect relational asset (e.g. mutual trust) against damage or loss from either end of the relationship
- *aspirational security*: protect aspirational asset against loss at 'near-end' (e.g. arbitrary alteration of brand) and at 'far-end' (e.g. damage to reputation of the brand)

Security for abstract assets such as finance is usually a complex-composite derived from variations of the above.

The various 'pervasives' such as privacy, quality, ethics and the like typically emphasise protection of different combinations of asset-categories, for example:

- *security*: protects physical/virtual assets, also some relational/aspirational assets such as trust

- *privacy*: protects virtual/relational assets
- *health and safety*: protects physical/relational assets
- *ethics and corporate social-responsibility*: protects aspirational/relational assets
- *environment*: protects physical/aspirational assets
- *quality*: generic – protects the asset-categories that apply in the respective business context

All of these may intersect in a sometimes bewildering variety of ways – for example, safety-critical IT-systems may interweave concerns about security, safety, environment and other themes all within a single architectural package.

The balance between incompletion and completion of composites also enables architectural redesign, by anchoring trails of relationship between items or layers, to resolve concerns such as strategic analysis, failure-impact analysis and complex 'pain-points':

- **composites are usable when architecturally 'complete'**;
- **they are *re*-usable when architecturally '_in_-complete'**.

The notion of *'bindedness'* – the extent to which a specific composite or primitive should or must be included within a solution – can be used to convert a model into a reference-framework. The obvious example of bindedness is legal compliance, because if we don't comply, we're breaking the law. But it also applies to use of standards of any kind, and many other practical concerns such as whether a particular operating-system or software version or type of milling-machine or whatever should be used in which contexts and for what purposes; and what types of skills and experience would be needed so as to deliver a particular service or decision. Levels of bindedness for any item in a reference-model would typically include:

- *mandatory*: item must be used wherever applicable
- *recommended*: item should be used unless a preferred solution mandates an alternative
- *desirable*: use of the item would aid in consistency
- *suggested*: experience indicates the item may provide a better or more consistent solution than similar alternatives

> Note that these might vary not only according to context, but in many cases will also change over time. Once again, very few items are ever truly static in an enterprise architecture.

Given all this wide variety of information that constitutes the overall framework, you're going to need somewhere to put it, and some systematic means to keep track of it all. At the least, you're going to need:

- *architecture repository*, including models and metamodels, core references, governance documents and other artefacts
- *requirements repository*, documenting requirements, tests for confirmation of those requirements, and relationships between them
- *risk and opportunity registers*, detailing any risks and opportunities identified in architecture, and actions to be taken to address them
- *glossary and thesaurus*, providing standard definitions and cross-references between synonyms, homonyms, heteronyms and the like
- *dispensations register*, recording implementations that have been allowed to not comply with the architecture, the reasons for allowing non-compliance, and how to resolve it in future

Which leads us to another key point, about architecture toolsets. Most people start their architecture with whatever tools come to hand, which typically means standard office applications such as Excel, Visio and PowerPoint. These do work well enough for first experiments, but they don't scale, they don't handle versioning, and they can't manage the myriad of essential cross-connections and cross-references that start to emerge as soon as you begin any serious architecture analysis. So at some stage, and probably sooner rather than later, you'll have to face the facts: you're going to need a proper enterprise-architecture toolset.

> And yes, this may hurt, because all of them are expensive, and all of them have significant or even serious limitations: some of them have user-interfaces that are best described as 'user-hostile', and as yet none of them get close to delivering all of the functionality we need for whole-of-enterprise architecture. But there's no way round it: some kind of purpose-built toolset *is* essential for this type of work.
>
> There's a wide variety of tools out there: some emphasise visual modelling, others more the underlying structures and metamodels. As a consultant and adviser working in many different industries, I prefer the latter, but that's what matches my own needs, whereas your needs may well be different. So I'll avoid making any recommendations here, because the 'right' toolset is the one that works best for you in your own context.
>
> The TOGAF specification provides some useful suggestions – see Chapter 42, 'Tools for

> Architecture Development', in the TOGAF 9 book or website – and there's an excellent summary of selection-criteria on the Institute for Enterprise Architecture Developments website at www.enterprise-architecture.info/EA_-Tools.htm[7] - these should help to ease the pain of picking the right toolset for your needs.

In effect, the framework defines *meaning* within the enterprise and its architecture: and we do need at least some of it in place before we start work. But on its own, the framework does nothing: so the next item we need is a methodology for architecture development.

Practice – methods

Although there are a fair number of formal definitions around for enterprise-architecture, there are surprisingly few that describe methodology – what to *do* in architecture, how to do it, in what sequence, and so on. For most practitioners these days, it almost comes down to a single choice: the Architecture Development method (ADM) that's part of TOGAF, the Open Group's architecture framework.

Which is unfortunate, because whilst it's a good choice for the early stages of enterprise architecture – especially in information-based industries such as insurance and finance – it's essentially focused only on IT, and hence is *not* good

[7]http://www.enterprise-architecture.info/EA_Tools.htm

for later-maturity architecture, which does need to cover the whole of the enterprise, beyond just its IT. So we need to do some amendments here to make it usable in practice for the whole scope of enterprise architecture, in any industry and at every level of maturity.

> Sadly, this is still true of the latest version, TOGAF 9, which was released only a few weeks ago as I write this. Version 9 is much better-structured than the previous version, '8.1 Enterprise', but it still has the same fixed IT-centric scope, and most of the ancillary information assumes an IT-centred world. It's still useful, though, to read the TOGAF 9 specification, either in the published book or in the online version: see the *Resources* section below for the publication details.
>
> It doesn't matter much to us, anyway: the amendments described here will resolve all those constraints, and will work equally well with either version of TOGAF.

TOGAF's ADM consists of a cycle of eight phases – labelled A to H, and often referred to as 'crop circles'. These are organised in a circle around a central repository for requirements. The first four phases focus on architecture assessment: Architecture Vision for the overall cycle (Phase A); Business Architecture (Phase B) Information Systems

Architecture (Phase C), usually split into sub-phases for data- and applications-architecture; and Technology Architecture for IT-infrastructure (Phase D). Phases B to D each have explicit subsidiary steps for 'as-is', 'to-be' and gap-analysis. The second set of four phases deals with defining (Phase E), planning (Phase F) and implementing (Phase G) a 'roadmap' for change, and following through with a final 'lessons-learned' activity (Phase H).

These are all preceded by a 'Preliminary Phase', in which we would set up the architecture capability itself, its governance, and various core documents such as the Architecture Charter and Architecture Principles.

To amend it to work better beyond its inherent over-emphasis on low-level IT, the main changes we need are:

- stronger support for Agile-style development
- allow any scope – not just IT-architecture
- stronger and more explicit integration with governance
- stronger integration with information-repositories

So first, we clear the decks a bit in Phase A. In the original version 8.1, a typical complete architecture cycle would take many months at least. TOGAF 9 allows for more explicit subsidiary iterations within the cycle, but the overall cycle itself still takes as much as a couple of years. What we need instead is something that can still deliver real business value in as little as a couple of hours. To do this, we move much of the content of the existing Phase A back up into the Preliminary Phase, leaving Phase A free to concentrate on the specific details for the current iteration alone.

Next we need to change TOGAF's fixed scope. We can do that very simply, by setting the scope in our amended Phase A, and defined in terms of the respective layers, columns and segments in the framework. We then change the focus of Phase B to D: instead of the previous fixed scope for each phase, with separate 'as-is', 'to-be' and gap-analysis in each, we swap these round such that Phase B covers the 'as-is' for the selected scope, Phase C does the 'to-be', and Phase D the gap-analysis.

When that's done, we could summarise our revised version of the ADM as follows:

- **Phase A**: Define business-scope, business-purpose and future time-horizon(s) for the iteration; scope also identifies respective stakeholders and applicable governance for assessment and any probable implementation phases
- **Phase B**: Identify the baseline (what is already known in the architecture-repositories about the scope), then assess in more depth the 'as-is' context (adding content to the repositories as we do so)
- **Phase C**: Repeat Phase B for the one or more 'to-be' time-horizons specified in Phase A

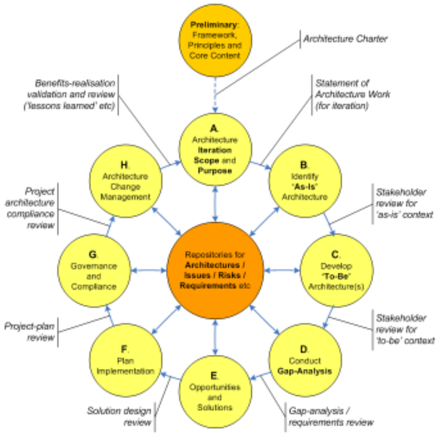
- **Phase D**: Do a gap-analysis for each 'as-is' and 'to-be' pair (from Phase B and C respectively), to identify requirements, constraints, risks, opportunities and suchlike for future change.
- **Phase E**: Review the results of Phase D to allocate priorities to requirements and identify appropriate means to implement the requisite changes ('solutions', in classic IT-speak); the applicable governance-rules shift from those of architecture to those of change-management during this phase
- **Phase F**: Establish a detailed plan for project-, programme- or portfolio-management to handle the changes 'from here to there' – in particular, dealing with the 'people' and 'preparation' aspects of change
- **Phase G**: Architecture assists change-governance with compliance, consistency and inter-project synergies during implementation of the planned business change
- **Phase H**: Return to architecture-governance to do a 'lessons-learned' review in relation to the respective business context, and identify any needs for further related architecture work.

As with the Zachman-like framework that underpins it, this restructure of the TOGAF ADM is described in more detail in *Bridging the Silos*, in the 'Methodology' chapters. There's

> also a two-page summary of the methodology at tetradianbooks.com/ebook/silos_real-ea-adm-ref.pdf[8] [PDF].

From this, *every* architecture iteration has the same overall structure. All that really changes is the scope, and the purpose for the iteration: once that's set up in Phase A, the rest of the activities follow on naturally from that. The 'Preliminary Phase' is actually an architecture iteration in which the scope is the architecture-capability itself. And technically speaking, Phase H is also an architecture iteration in its own right, where the scope is the current context of the architecture: but it's simplest to leave it where it is, if only to remind us to review the architecture itself on a regular basis.

[8]http://tetradianbooks.com/ebook/silos_real-ea-adm-ref.pdf


Preliminary: Framework, Principles and Core Content
Architecture Charter
Benefits-realisation validation and review ('lessons learned' etc)
Statement of Architecture Work (for iteration)
A. Architecture Iteration Scope and Purpose
H. Architecture Change Management
B. Identify 'As-Is' Architecture
Project architecture compliance review
Stakeholder review for 'as-is' context
G. Governance and Compliance
Repositories for Architectures / Issues / Risks / Requirements etc
C. Develop 'To-Be' Architecture(s)
Project-plan review
Stakeholder review for 'to-be' context
F. Plan Implementation
D. Conduct Gap-Analysis
E. Opportunities and Solutions
Solution design review
Gap-analysis / requirements review

Governance-artefacts define architecture-cycle's phase-boundaries

Crucially, anything that looks like content or activities for a later phase is deliberately held back until that phase. For example, we don't try to do implementation during architecture-assessment, because the applicable governance-rules are usually different in the respective phases. If we did try to do implementation too early, we'd be likely to do it wrong, or at the least annoy some important stakeholder who should have been involved!

This especially applies to any pre-packaged 'solutions': anything that looks like a 'solution' is explicitly shelved until Phase E at the earliest, so that we fully understand the real

> requirements in the context *before* we look at any purported 'solutions' to those requirements. The aim here is to reduce the incidence of *deus ex machina* delusions – such as the classic cart-before-the-horse mistake of 'IT-solution looking for business-problem' – that so often cause so much bitter contention with the wider business community.

The other advantage of this redesign is that it simplifies the link to governance. Each phase now ends with an explicit stakeholder-review: in effect, the reviews define the phase-boundaries. And the review itself provides a PRINCE2-style 'go/no-go' gateway at the end of each phase: so we don't assume – as classic TOGAF does – that every enterprise-architecture assessment must automatically lead to a major 'roadmap' for all-out change of the IT-systems. Often there's real business-value to be gained just from the assessment itself.

> This is especially true once we acknowledge Snowden's dictum that 'every intervention is a diagnostic, and every diagnostic is an intervention': the *process* of architecture itself – above all, the engagement in dialogue – is often the only direct 'intervention' we need.
>
> If you really want to cut costs, improve efficiency, enhance overall effectiveness, often

> the best way to do this is to get people to rethink their own way of working – not impose a 'new way of working' on them from outside. Enhanced awareness of the way in which everything links with everything else leads to small changes in action that can snowball by themselves into *natural* larger-scale change: and if we do it right, it hardly costs anything at all.

Another source of business-value is that, once we link the architecture information-repositories more strongly into the assessment methodology, even the smallest iteration will add a little more into the 'holograph', that body of knowledge about enterprise structure and purpose. This not only includes models and other architecture information, but requirements in general, updates to the shared enterprise glossary and thesaurus, and notes on new and revised risks and opportunities.

> That perhaps doesn't sound much, but even small updates to the thesaurus, for example, can help a lot, if sometimes in unexpected ways. Misunderstandings and miscommunications can cost a great deal, and not just in direct monetary terms: that thesaurus will *matter*.
>
> And opportunities can arise in some very unexpected places. The most intensive users of

> early mobile phones were not in the upper echelons of big-business – as the telcos had expected – but jobbing builders and small-traders, for whom communications on-site and on the move had a very high value indeed. And in those same early days, engineers would often exchange test-messages with each other via the technology's parallel control-channel, which usually worked even when there wasn't enough signal for the normal voice-channel to operate. The technique seemed useful, so, as a small experiment, one telco made the channel visible on their subscribers' handsets – and was taken completely by surprise when what's now known as 'texting' took off like wild-fire. That's why text-messages are restricted to 160 characters: the control-channel wasn't designed for texting at all. But it's a clever re-use of existing technology – leading to a very valuable business opportunity.

The other point here is that by linking review of the repositories into the architecture cycle, they should never go out of date – because they're re-assessed and updated every time we do any work in the respective space.

Historically, enterprise architecture frameworks have been great on the theory, but often not so great on the practice. So there's also a lot about methodology that we can learn from other disciplines, above all on the links between knowledge, governance and action. Two valuable sources in this are ITIL – especially Version 3, which is far less IT-specific than the previous version – and PMBOK, the 'Project Management Body of Knowledge'. Another well-known example is Christopher Alexander's work on patterns in building-architecture, which has been adapted to many other architectural domains. But there are plenty of other sources for good ideas: all we need do is keep our eyes open, and remember always to think wider than just the usual constraints of the IT-centric world.

Performance – metrics

What's the value of your architecture? And value to whom? These aren't trivial questions: if you can't answer them, and if you can't prove the value, it'll likely be your job that's on the line…

There are actually four different yet interdependent themes here:

- identifying what 'value' means within in the enterprise – in other words, not just money, but all the other 'pervasives' too;
- identifying the metrics and transforms needed to measure and monitor business-activity and business-performance in terms of each of those values;

- identifying and measuring the impact that the enterprise-architecture has on those measured values – which gives us the business-value of the architecture;
- monitoring and measuring the performance of the architecture practice – just like any other business capability.

None of these are as simple as they might at first seem – though measuring the business-value of the architecture is probably the hardest challenge, because some impacts are identifiable only in terms of what *didn't* happen, what *didn't* go wrong, and often the real returns can be measured only in the performance of the enterprise as a whole.

> Oddly enough, metrics and business-value are hardly addressed at all in the TOGAF specification. You'll find more detail, though, in the 'Completion' chapters in *Bridging the Silos*, whilst the role of the 'pervasives' is explored in depth in another companion book, *The Service Oriented Enterprise* – see the *Resources* section for more information on those.

One of the hardest parts – which is why we need to address it right from the start – will be weaning people away from an over-dependence on measuring performance only in monetary terms. Whether we think of it in the positive, as profit – in a commercial enterprise – or negative, as cost-cutting – such as in a government-department – money is, obviously, an interesting and important metric. But oddly, it's often not much *use* as a metric, because very few things are directly controlled *by* money. Technically speaking, it's more usually a complex derivative, an *outcome* of other factors rather than a driver in its own right: and what we most need to measure are the things we *can* control.

> A perhaps useful piece of historical trivia here: the word 'economy' literally translates as 'the management of the household', in fact up until the middle of the nineteenth century the word 'economist' was essentially synonymous with 'housewife'. Then a few folks came up with the metaphor of an enterprise or a whole nation as a 'household' on a larger scale – and hence 'the economy' as the management of that shared 'household'. Hence, in turn, all the present-day notions of 'economics'.
>
> But in doing so, they made a fundamental mistake: they measured that economy only in monetary terms, and ignored everything else. If you do that, the 'economy' metaphor no longer works: managing the household finances is rarely easy, but in many ways it's the *easy* part of managing a household, and it's certainly not the only one!

Balanced Scorecard is a good start, to get people thinking wider than money: but even that still places financial return as one of its four dimensions, placing it on an equal footing with 'Customer', 'Internal Business Processes' and 'Innovation and Learning'. This is misleading because, unlike the others, financial return is not a lead-indicator but a lag-indicator – it tells us where we've been, but not where we're going. To manage an enterprise – or for that matter the architecture of that enterprise – what we need are those lead-indicators.

Which is where the values of the enterprise – its 'pervasives' – come into the picture, because by definition they're the themes that are of value *to* the enterprise, and hence the things we most need to measure. Which is why we need to identify them as early as possible within our enterprise-architecture work – see section 'Vision, values, principles and purpose' in chapter *Step 1: Know your business.* As a very minimum, we need to measure against those five strands of effectiveness:

- Is it *efficient*? – does it make the best use of the available resources?
- Is it *reliable*? – can it be relied on the deliver the required results?
- Is it *elegant*? – does it support the human factors in the context?
- Is it *appropriate*? – does it align with the business values and purpose?

- Is it *integrated*? – does it help to link everything together?

Once we have those, and a real ‘balanced scorecard’ that’s tailored to the true values of the enterprise, we have a means to measure the value of the architecture.

We also need to monitor our own performance in doing enterprise architecture. Again, a focus on those five strands of effectiveness will be helpful here, but probably the best approach will be to use one of the ‘capability maturity models’ that are becoming more generally available for enterprise architecture.

> Two maturity-metrics that I’ve found useful are Marlies van Steenbergen’s ‘DyA Architecture Maturity Matrix’, from her book *Building an Enterprise Architecture Practice*, and the Meta Group’s ‘Architecture Maturity Audit’. Even though they’re aimed primarily at IT-architecture, they’re sufficiently detailed and fine-grained enough to give valuable pointers for any type of enterprise architecture – though in my experience the Meta Group’s version is the more useful of the two for assessing the early stages of architecture maturity.
>
> To keep things simple, though, we’ll stick with the minimalist maturity model from the TOGAF specification for the rest of this book,

> as its basic five-step frame fits better with our more generic needs here. It doesn't measure performance as such, but it does provide clear guidance as to what to do next at each step – which is, after all, the real purpose of performance-metrics.

The other key aspect of performance is about lessons learned from the previous 'perform_ing_' of the work. Enterprise architecture manages a body of knowledge about enterprise structure and purpose: what have we added to that body of knowledge during this cycle of work? To quote the US Army's 'After Action Review' process:

- What was supposed to happen?
- What actually happened?
- What was the source of each difference?
- What can we learn from this, to do differently next time?

Keeping track of 'the numbers' will help – no doubt about that. But it's questions like those above, extending the *narrative* and dialogue of the architecture, that will really help most to embed the architecture into the enterprise, and to demonstrate in direct, everyday practice the real value of our work.

Resources

- FEAF (Federal Enterprise Architecture Framework): see www.gao.gov/special.pubs/eaguide.pdf[9] [PDF]
- Preparation for enterprise IT-architecture: see Chapter 6 'Preliminary Phase' and Chapter 7 'Phase A: Architecture Vision' in Open Group, *TOGAF™ Version 9* (Van Haren, 2009)
- Architecture Development Method (ADM): see Parts II and III in online TOGAF 9 at www.opengroup.org/architecture/toga doc/arch/[10]
- DyA (Dynamic Architecture): see eng.dya.info/Home/[11]
- Effectiveness in whole-of-enterprise architecture: see Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008)
- ITIL (IT Infrastructure Library): see www.itil-officialsite.com [12]
- COBIT (Control Objectives in Information and related Technology): see www.isaca.org/cobit[13]
- PRINCE2 (Projects In Controlled Environments): see [www.ogc.gov.uk/methods_prince*2.asp](http://www.ogc.gov.uk/meth prince*2.asp)

[9]http://www.gao.gov/special.pubs/eaguide.pdf

[10]http://www.opengroup.org/architecture/togaf9-doc/arch/

[11]http://eng.dya.info/Home/

[12]http://www.itil-officialsite.com

[13]http://www.isaca.org/cobit

- Zachman Framework: see www.zifa.com[14]
- Framework and method for Agile architecture: see Tom Graves, *Bridging the Silos: enterprise architecture for IT-architects* (Tetradian, 2008)
- PMBOK (Project Management Body of Knowledge): see Project Management Institute at www.pmi.org[15] and Wikipedia summary at en.wikipedia.org/wiki/ProjectManagemer
- Patterns: see Christopher Alexander, *A Pattern Language: towns, buildings, construction* (Oxford University Press, 1977)
- Balanced Scorecard and its variants: see Wikipedia summary at en.wikipedia.org/wiki/Balanced_scorecard[17]
- Architecture maturity model: see Martin van den Berg and Marlies van Steenbergen, *Building an Enterprise Architecture Practice: tools, tips, best practices, ready-to-use insights* (Sogeti / Springer Verlag, 2006)
- Meta Group Architecture Maturity Audit: *Meta Group Practice*, (2000), Vol 4 No.4 (for Part 1) and No.5 (for Part 2)

[14]http://www.zifa.com
[15]http://www.pmi.org
[16]http://en.wikipedia.org/wiki/Project_Management_Body_of_Knowledge
[17]http://en.wikipedia.org/wiki/Balanced_scorecard

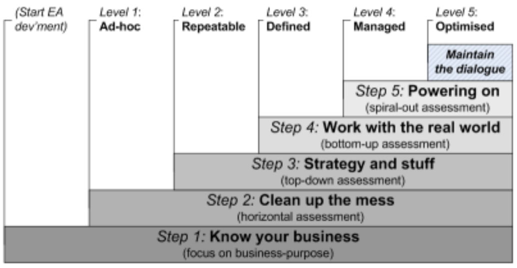
- After Action Review and other ‘lessons learned’ techniques: see Chris Collison and Geoff Parcell, *Learning to Fly: practical lessons from one of the world’s leading knowledge companies* (Capstone, 2001)

A matter of maturity

What we need to do at each step in enterprise-architecture depends to a large extent on the level of maturity that's been achieved to date. The TOGAF specification describes five distinct levels that it labels 'ad-hoc', 'repeatable', 'defined', 'managed' and 'optimised'. Those, though, are the maturity-levels we achieve *after* we've done the work. The typical steps for the work itself would be more as follows:

1. From nothing, to creating a stable base for 'ad-hoc' fixes – see chapter *Step 1: Know your business*
2. From 'ad-hoc', to structures that are efficient, reliable and repeatable – see chapter *Step 2: Clean up the mess*
3. From repeatable, to sufficiently well defined to respond to the changing needs of strategy – see chapter *Step 3: Strategy and stuff*
4. From defined, to managed well enough to respond in real-time to the confusions and crises coming up from the real world – see chapter *Step 4: Work with the real world*
5. From managed, to optimised well enough to tackle the organisation's more intractable 'pain-points' and 'wicked problems' – see chapter *Step 5: Powering on*

These layers build on each other, but the work of each layer never actually ends: we need to continue to revisit the respective context and scope for each, to review and amend as we climb upward.


(Start EA dev'ment)
Level 1: Ad-hoc
Level 2: Repeatable
Level 3: Defined
Level 4: Managed
Level 5: Optimised
Maintain the dialogue
Step 5: Powering on (spiral-out assessment)
Step 4: Work with the real world (bottom-up assessment)
Step 3: Strategy and stuff (top-down assessment)
Step 2: Clean up the mess (horizontal assessment)
Step 1: Know your business (focus on business-purpose)

Continuing updates for each layer underpin maturity growth

So if and when we get to the ‘Optimised’ maturity-level, there’ll always still be further work to do, in maintaining the dialogue that underpins the architecture – see What next?[18] But at least we know we’ll be doing it from strong, stable foundations – and contributing in every way towards real business value.

Resources

- Levels of architecture maturity in TOGAF: see chapter 51, ‘Architecture Maturity Models’ in The Open Group, *TOGAF*™ *Version 9* (Van Haren, 2009)

[18]../Text/Whatnext.xhtml#anchor40

- TOGAF (The Open Group Architecture Framework): see www.opengroup.org/architecture/togaf9-doc/arch/[19]

[19]http://www.opengroup.org/architecture/togaf9-doc/arch/

Step 1: Know your business

We can only implement a strategy successfully if we have *some* initial idea about what business we're in. IT-oriented 'enterprise architects' often try to skip this step in the rush to get down to 'the fun stuff' – the fine detail of computing technology. But it can be a lethal mistake, because we end up optimising the technology for a business that we don't even know – and hence are then all but forced into the trap of promoting pre-packaged IT-'solutions' *as* strategies, because it's the only part of the business we *do* know. So, to put it simply:

- **Whatever part of the business we're in, the business of the business must always come first.**

Which means that however much we might want to 'get down and get dirty' with the detail-level IT – which usually is the next stage, by the way – we *must* do this task first.

Purpose and strategy: The focus here is on the nature of the business – its vision, values and business purpose, the business milieu in which it operates and its role within that context, and its core business functions and processes.

Our aim here is to show *why* an architecture capability would be useful for the enterprise – and demonstrate that use by producing, at minimal cost, a simple set of architectural artefacts that have immediate practical value for the business.

> From a FEAF perspective, what we're creating here is a first cut at the BRM and PRM – the Business Reference Model and Performance Reference Model.
>
> From a TOGAF perspective, much of what happens here would be termed 'Business Architecture'. But note that we do much of it *before* we develop the Architecture Charter and the like: in fact much of the equivalent of TOGAF's Preliminary Phase happens near the end of this stage, rather than at the start. The reason is simple business-politics: yes, we must do all the Preliminary Phase work before we get down to what TOGAF thinks of as 'enterprise architecture' – but we won't even get that far unless we already have some business credibility behind us. Which means we need to do something that is immediately meaningful and valuable to everyone in the business – not just the IT-folks. Hence the slightly back-to-front way we'll do things here.
>
> We still do it under solid governance, though – stakeholder reviews and all that. Unless we do that, it isn't *architecture.*

People and governance: This is only a first-cut ‘proof of concept’, so we’ll need to keep things as simple as possible. Often it’ll be slipped in ‘under the counter’ as a kind of skunk-works project, with costs quietly buried under someone’s discretionary budget – though for credibility reasons it’d be best if that ‘someone’ is fairly senior to start with. In governance terms, that ‘someone’ would also be who we would report to during the work, though the end-results – the Function Model, the Architecture Charter and so on – should be distributed for comment as widely as possible.

> This is one of the few parts of enterprise-architecture that could be done entirely by external consultants or other ‘outsiders’. In fact it’s almost better to use outsiders here, especially in the early stages, because they would see the enterprise differently from any ‘insider’s assumptions, and would also have more licence to ask the essential ‘stupid questions’ such as “What *is* this business, anyway?”.

Planning and frameworks: We have a minor dilemma here, in that whilst we’re aiming to create the core overviews of the enterprise, the ‘holograph’ approach means we need one already existing before we can start, so as to derive business value. The answer is to use the Zachman-like framework described earlier in section ‘Planning –

frameworks’, in chapter *Preparing for architecture*, as it provides a core taxonomy to make sense of where and how everything fits together. Ultimately, everything we’ll do from now on will anchor back into that core framework: so we don’t just end up with an enterprise architecture, we start with one.

> You will also need a fairly complete architecture toolset: architecture repository and framework, requirements repository, risks and opportunities registers, and preferably the glossary and thesaurus as well. (You won’t need the dispensations register as yet, because we don’t get that far in this stage.)
>
> But whilst it does have to be fairly complete, it doesn’t need to be that sophisticated as yet. For the first pass through this work, you *can* get away with using standard office applications such as Excel, PowerPoint and Visio. This would also be good as a test-case for a trial version of a proper purpose-built toolset – but do first make sure that you *can* export from the toolset, otherwise you risk losing all of this work.

Practice and methods: Each section of work should follow the standard method and governance as described earlier in section ‘Practice – methods’, in chapter *Preparing* *for architecture.* The only variation would be that in some sections, such as developing the Function Model – see section 'Functions and services' below – it may be useful to do the 'to-be' (Phase C) before the 'as-is' (Phase B), to help stakeholders disengage from assumptions about the imagined 'inevitability' of current organisational structures and processes.

Although it's unlikely there would be much – if any – design and implementation for formal change-projects to do here, it will still be important to engage programme-management and the like in the overall process – if only to garner their suggestions and feedback on how the handovers of governance-responsibilities will need to work when we move later into the more active stages of enterprise architecture.

Performance, end-products and metrics: First time through, the overall work for this step should require no more than a few person-weeks of effort – just enough to establish the intended role and business value of the architecture capability. You'll probably need to take more time when you revisit this work at later stages of maturity, but that above should be sufficient to make a useful start here. Typical artefacts from this step would include:

- strategic description of the enterprise context – vision, values, purpose, market, legal and regulatory milieu, etc

- high-level descriptions of the needs within that context that are serviced by the organisation
- core content for the architecture framework – assets, functions, locations, capabilities, events and decisions
- core Function Model – 'the enterprise on a page' – summarising how the organisation serves those needs
- formal documentation of the authorised roles, responsibilities, activities, deliverables, funding and reporting-relationships for the enterprise architecture capability

This step would usually be run as an explicit short-term project, hence key metrics would typically include on-time, on-budget, and all required deliverables completed. Also some measure of customer-satisfaction would be useful – not only satisfaction of direct customers such as the project-sponsor, but more generally of the likely architecture stakeholders.

Vision, values, principles and purpose

At the highest level, 'the enterprise' is not the organisation, but the broader context in which the organisation operates, and which it shares with all other stakeholders in that enterprise. The ***purpose*** of this section is to identify and document the organisation's relationships with that broader enterprise.

> You'll also find this type of assessment useful as part of due-diligence for a proposed merger or acquisition. Assuming you've already done this for your own organisation, this exercise will provide you with the essential information for a gap-analysis on culture and background – the basis for a crucial go/no-go decision, because the degree of alignment is a known key criterion for future success or failure here.

On the ***people*** side, the skills you'll need are those for routine business-analysis. For a first pass you'll probably only need to meet up with a few strategists and other senior players, though for later iterations you'll need to extend that scope, eventually to every part and every level of the overall enterprise.

For ***planning***, you'll need access to typical business documentation such as the Annual Report and the corporate intranet. The architecture-entities you identify will usually be placed in the topmost rows of the framework.

The ***practice*** would be based on techniques such as visioning and the Business Motivation Model, with appropriate documentation.

As described above, ***performance*** would usually be measured in simple project-completion terms: on-time, on-budget and suchlike.

Vision, role, mission, goal

Use the 'vision, role, mission, goal' structure to summarise the overall ecosystem in which the organisation exists, and the role or roles it chooses to play within that ecosystem.

> Note that the terms 'vision' and 'mission' are used here in a slightly different way than in common business use – see the *Glossary* appendix for definitions. These usages help us to avoid the temptation to view the organisation *as* the enterprise – one of the most common yet most dangerous mistakes in business-visioning.
>
> For more detail on the 'vision, role, mission, goal' process, see the chapter 'Architecture on purpose' in *Real Enterprise Architecture: beyond IT to the whole enterprise*, and the 'Practice – Service Purpose' chapter in *The Service Oriented Enterprise: enterprise architecture and viable services.*

The vision is not a 'future state' for the organisation itself, but describes the common theme shared by *all* stakeholders in the enterprise. The typical phrasing of a vision would not be emotive in itself, but would invite or incite an emotional commitment to the shared 'cause': two good examples are "a sociable world" (brewers Lion Nathan) and "boundaryless information flow" (standards-body The Open Group). A mission here is more in the sense of a trade-mission, not a military-style 'sending' – in other words, an ongoing *capability* to serve a role within the enterprise, rather than an activity with a defined point of completion.

- What **vision** is common to all stakeholders in the enterprise? What single phrase *describes* the overall enterprise?
- What **role** or roles does the organisation play within the enterprise, to contribute towards the vision? What roles does it *not* play – hence leaving open for other stakeholders? Who are these other stakeholders, and what roles do *they* play within the overall enterprise? What relationships and transactions are implied by these different roles within the enterprise? How would you verify that each role – especially the organisation's own chosen role or roles – does support the enterprise vision?
- What **mission** or missions – ongoing services and capabilities – would be needed to support each role undertaken by the organisation in that enterprise? What metrics would you need to confirm that each

mission is 'on purpose' and *effective* in supporting the respective role?

- What short-, medium- and longer-term **goals** and objectives underpin each mission? What are the timescales and deliverables for each goal? By what means do you verify that each goal is achievable, and has been achieved? How do you verify that each goal does support the respective mission in an appropriate and effective way?

Once this basic high-level picture has been established, use a technique such as the Business Motivation Model to devolve the view downward into the detailed operations of the organisation.

> Beware, though, that the BMM does tend towards that dangerous self-referential notion of organisation as 'the enterprise' – particularly in its handling of higher-level terms such as 'vision' and 'mission'. Remember that the real enterprise here is *always* larger than the organisation: the detail-layers of the BMM work well enough, but take care at the top!

Store the results in the architecture-repository either in row-0 (for the vision and role, which should probably never change) or in the upper rows of the 'reason/decision' column (for missions and goals). Also note any additional information requirements, risks and opportunities that may arise, and save these to the respective repositories.

Values

Next, identify the required, espoused and actual **values** indicated for the organisation's relationships in the enterprise:

- What shared values are required from every player in the overall enterprise, in order for that enterprise to achieve success within the terms of its vision?
- What individual and collective values – either implicit or explicit – are required to support each of the relationships and transactions in the organisation's roles within the enterprise?
- What values does the organisation espouse, both in its relationships with others and its relationships within and to itself?
- What actual values does the organisation express in its actions and relations with others and within itself?
- Are there are any misalignments between required, espoused and actual values? If so, what impacts do these differences have, on the effectiveness of the shared enterprise, of the organisation in relation to and with the enterprise, and the organisation's internal function and relationships within itself?

Identify required values from an assessment of what 'effectiveness' would look like in relation to the enterprise and its vision: for example, fairness and trust will usually be required in almost any functional enterprise.

Identify espoused values from the organisation's Annual Report or other public sources, such as an 'About Us' or 'Our Values' section on a website, or publicity material provided to prospective employees.

Identify actual values from the behaviours or phrases actually used within the organisation; note especially how these may vary at different levels or in different parts of the organisation.

> The SEMPER diagnostic can be valuable here, because it's designed to identify the effective 'ability to do work' from the kind of language used to describe different aspects of the work-context. If the organisation claims that it values transparency, for example, yet people are using phrases such as "we're just mushrooms, fed sh*t and kept in the dark", it does kind of imply there's a significant mismatch between espoused and actual values! In some organisations, the scale of mismatch can be *huge* – with correspondingly huge impacts on the organisation's overall effectiveness, too…
>
> A SEMPER analysis can be downright scary at times, because of the ease with which it surfaces problematic issues such as those value-mismatches; but it does also describe what to

> do to resolve each of those problems. You'll find more detail on SEMPER in the book *SEMPER & SCORE*, and also on the sempermetrics.com[20] website.

Document the required and espoused values in the uppermost rows of the architecture framework, and also as core requirements in the requirements repository. Document each values-mismatch as a priority item in the risks register.

Principles

From values we move to **principles**, because the organisation's principles describe how its values should be expressed in practice. Values define the 'pervasives' for the enterprise, but in themselves are somewhat abstract; whereas principles are – or should be – concrete, actionable and verifiable.

One key complication is that, by their nature, principles will often compete or conflict: transparency versus privacy, for example, or innovation versus the safety of 'the known'. We need to document such clashes, and, wherever practicable, assign priority to one or other principle so as to simplify decision-making in the field.

> One useful principle, perhaps, is an acknowledgement that although every person in the

[20]http://sempermetrics.com

enterprise is personally responsible for resolving the balance of principles, no-one ever actually achieves it – not in the real world, anyway!

The appropriate test there would not be a demand for an impossible 'perfection', but more that all applicable principles were taken into account, and that all reasonable efforts were made to resolve any conflicts, under the constraints of the context.

Principles are not law as such, but in some ways are almost *above* the law – they're what law is drawn from, in fact. Law describes how things *ought* to work in theory; principles provide guidance as to how to make things work in practice. In effect, a law is a pre-packaged interpretation of principles: so whenever we meet a circumstance where 'the law' – in the enterprise sense, as the organisation's rulebook or whatever – doesn't make practical sense, we need to recognise the primacy of the underlying principles.

That's why the first principle in the TOGAF specification is about the primacy of principles: principles really do come first. But they in turn express the enterprise values; and the

> values themselves express the shared vision that defines what the enterprise *is*. That's why all this perhaps abstract-seeming stuff *matters*: because without it, the organisation has no practical purpose. Which is *not* a good idea...

Principles form hierarchies as they devolve down into the deeper detail. But ultimately every one of them needs to be anchored back into one or more values; and in turn every value needs to be expressed in explicit, measurable, actionable principles:

- What principles apply in the enterprise? In what ways are these principles expressed and documented?
- What value or values does each principle express? How would you confirm that those values are expressed by the principles?
- Are there any principles – explicit or implicit – that do *not* seem to be anchored in any espoused enterprise value? If so, what values are implied by such principles? Do any such 'shadow' values conflict with the espoused values of the organisation? If so, how, and which values 'win'?
- What principles express each value? Is every espoused value expressed within at least one hierarchy of principles which devolves all the way down to the

operations layer? If not, what principles would be required to express that value?

- In what ways are each principle applied, in theory, and in practice? What evidence exists that they *are* applied? – or not applied? If not applied in practice, what needs to be done to ensure that they *are* applied? What metrics would be needed to monitor and confirm this?
- What conflicts exists between principles? What guidance is provided to help people resolve such conflicts between principles in their own work?

For a first pass at this in an 'architecture demonstrator' project, you would probably only assess this at the most abstract level; but in later reviews you'll need to go much deeper, sometimes right down to the fine detail of system design and everyday operations. It's only when values are expressed as principles that they become meaningful *as* values: until then they're just an abstract 'nice idea'.

> The TOGAF 9 specification includes a very useful section – Chapter 23, 'Architecture Principles' – about principles and how to define and document them. As usual, we'll need to compensate for TOGAF's obsessive IT-centrism – such as its bizarre assertion that architecture principles are a subset of IT-principles – but otherwise their recommended format for defining principles works well in any context:

- *name*: represents the essence of the rule in a form that is easy to remember;
- *statement*: presents a succinct, unambiguous summary of the rule;
- *rationale*: anchors the principle back to the business reasons and business-benefits arising from the principle, and ultimately to the core value or values that it represents;
- *implications*: describes how the principle should be expressed in everyday actions or in influence on practical decisions.

"Essentially, principles drive behaviour", says the TOGAF specification. It also adds a checklist of keywords for validating well-described principles: *understandable*, *robust*, *complete*, *consistent* and *stable*; to these we should also add *measurable* and *verifiable*, so that we have some means to confirm that the principle has been applied n practice.

Recommended, anyway.

Principles represent the core reasons and decisions of the organisation, pervading the values throughout every layer and function, downward into the fine detail of systems designs and individual actions. Document them in the respective rows of the 'decisions' column of the framework and in the requirements repository, as linked trails of decomposition and derivation linked back to the root enterprise-value.

Business purpose

Finally, as a crosscheck, we come back to **purpose**. The previous parts of the assessment should have identified what the organisation believes its business purpose to be; we now need to check that against the reality. The difference can sometimes be painful, but nonetheless important to know…

Our guide for this is a phrase coined by the cyberneticist Stafford Beer: "the purpose of a system is what it does". In systems-theory terms, the organisation is a system in its own right, interacting with the larger ecosystem of the overall enterprise. The vision, values and principles define what the organisation's purpose *should* be, in theory; Beer's phrase – usually shown as its acronym 'POSIWID' – indicates its effective purpose in practice, and hence the *actual* principles, values and, probably, vision.

> The organisation states that health and safety are key values, but the reality is that there

> are high rates of accidents, illness and absenteeism. From POSIWID, we can be fairly certain that those themes are *not* valued in practice…
>
> The organisation's websites asserts the primacy of quality above everything else in its products. But POSIWID shows us that the managers believe the only thing that matters is 'the numbers' – and their bonuses reflect that fact, too. Notice what impact this has on product-quality – and on overall effectiveness…
>
> Values *matter*: yet we need to be clear which values *actually* apply in the enterprise, otherwise we have no means to identify when what we do really *is* 'on purpose'.

Any values-mismatch will lead to ineffectiveness: each is something we'll need to address as an organisation – and hence needs to be documented in the architecture gap-analysis.

- If "the purpose of the system is what it does", what *real* principles and values are implied in what happens in the organisation and in its relations with the overall enterprise?

- Comparing the results with the values and principles already documented, what conflicts can be identified? In each case, do the espoused values have priority, or the actual 'shadow' values? What is the impact of each values-mismatch on overall effectiveness?

Document each mismatch of values or principles as a high-priority entry in the risks-register.

The enterprise context

The ***purpose*** of this section is to establish the context and expectations of the overall enterprise in which the organisation operates. This includes concerns such as legal and regulatory constraints, applicable industry and other standards, and the key 'things', locations and events that comprise the enterprise context.

On the ***people*** side, the skills you'll need, as in the previous section, are those for routine business-analysis. You'll also need access to a handful of 'insiders' who know the nature of the business fairly well – which could be you, of course.

For ***planning***, most of the information you'll need would be provided by those 'insiders': you *could* find it by slogging your way through documents and intranets and industry reference-material, but asking the right people a few questions will be much quicker! The aim here is to populate more of the framework's upper rows, so you'll need some kind of architecture repository, as before.

The ***practice*** would be based on standard business-analysis techniques, documented as described below.

As before, ***performance*** would be measured in simple project-completion terms, reporting back to the project sponsor.

Compliance, constraints, standards, expectations

In the previous section we'll have identified the overall milieu – the 'vision' – and the role that the organisation plays within it, which identifies the effective requirements. In aligning itself to the vision, the organisation *chooses* concomitant constraints implied by its values and principles. Here we explore the constraints that come in to the organisation *from* the milieu, by dint of choosing to enact that role within the enterprise. These include any laws and other regulations that would apply; any required standards from the industry, or needed for interaction with customers, suppliers, partners and other players in the enterprise; and expectations from the general community about corporate social responsibility and the like.

These constraints will each have differing degrees of 'bindedness', and may vary in different jurisdictions, geographies, communities and so on:

- What laws and other regulations apply in this business context? To what extent and in what ways are they binding on the organisation? In what ways do these vary in differing jurisdictions and the like?
- What constraints – and opportunities – do these imply for the organisation and its business in the enterprise? What trade-offs do these imply against the business requirements?
- By what means would the business confirm its compliance with these constraints? What actions and information would be required? How would you monitor and measure compliance? What would be the consequences of failure to comply? What opportunities arise from the required compliance with these constraints?

Note that these can be a lot more complicated than they may at first seem: some regulations are binding across geographies, and some jurisdictions assert their reach into other contexts entirely. US regulations on technology export or money-laundering, for example, are deemed to apply throughout the entire supply-chain from source to end-user. And movement of staff may be restricted not only by local residency-rules, but also some very nasty booby-traps such as the way some countries extend their citizen-obligations to

> foreign-born children of former nationals. The definition of 'former national' can be dangerously blurry here, as one of our colleagues discovered the hard way in Greece after moving there on business: although he'd been born in Australia, and a full Australian citizen, the fact that one of his grandparents had been Greek meant he was also classed as a Greek citizen – and he was forcibly drafted into compulsory two-year military-service there. Such realities are genuine risks for the globalised enterprise: we need to be aware of them in the architecture.
>
> Such details may not matter too much on a first pass assessment, of course, but they certainly *do* matter when we explore these concerns in more depth in subsequent reviews.

Next we need to note the various standards that would apply in each context:

- What standards – quality-standards, technical standards, interface standards, language standards and suchlike – apply in each area of this business context? To what extent and in what ways are they binding on the respective areas of the organisation? In what ways do these vary in differing regions and the like?

- What constraints – and opportunities – do these imply for the organisation and its business in the enterprise? What trade-offs do these imply against the business requirements?
- By what means would the business confirm its compliance with the constraints of these standards? What actions and information would be required? How would you monitor and measure compliance? What would be the consequences of failure to comply? What opportunities arise from the required compliance with these constraints?

And there are what we might call 'good citizen' constraints, which may not have the force of law or formal standards behind them, but can still impose serious sanctions in terms of reputation or cooperation:

- What social expectations and social standards – ethics, environment, general 'good neighbourliness' and so on – apply in each area of this business context? To what extent and in what ways are they binding on the respective areas of the organisation? In what ways do these vary in differing regions and the like?
- What constraints – and opportunities – do these imply for the organisation and its business in the enterprise? What trade-offs do these imply against the business requirements?

- By what means would the business confirm its compliance with the constraints of these expectations and standards? What actions and information would be required? How would you monitor and measure compliance? What would be the consequences of failure to comply? What opportunities arise from the required compliance with these constraints?

> The SEMPER diagnostic might also be helpful here as an initial metric, identifying effective social reputation via language-cues in descriptions of the organisation from the broader community and other 'external' stakeholders.

Document the results in the respective parts of the architecture repository: the constraints themselves in the 'decisions' column of the framework, required metrics in the 'virtual asset' segment, risks and consequences in the risks-register, and so on.

Assets, locations, events

In this part of the work we aim to fill out more of the core parts of the framework, by identifying the assets, locations and events that are central to the organisation's business within the enterprise.

> There are a lot of questions in this part, many of which may seem very unfamiliar – even

> bizarre, perhaps – if your previous architecture experience has been only in an IT-centric context. Don't worry about it, though: all these questions and categorisations do matter, as we'll see later, but we don't have to do it all in one go! If any part seems too alien to make sense, do the best you can for now, and allow the questions to make more sense over time as this much larger picture of the enterprise starts to coalesce.
>
> Later on we'll need to go into all of this in much more depth, of course, but for a first-pass this review would need only to capture the most essential types of items – enough to create some kind of top-level anchor for subsequent iterations of the architecture cycle.

So first, assess the key **asset**-types:

- What types of *physical assets* – physical 'things' – are important to the organisation's business? What roles do these assets play in the business – for example, as input supplies, as output products, or used as consumables in business processes? What value does each type of 'thing' have for the business? How are these assets obtained, maintained, monitored, managed through their life-cycle, and disposed-of at the end of it?

- What types of *virtual assets* – data, information, knowledge – are important to the organisation's business? What roles do these assets play in the business – for example, as records, as metrics, as content for delivered services, as controls in business processes? What value does each type of virtual-asset have for the business? How are these assets created, obtained, maintained, monitored, managed through their life-cycle, and disposed-of at the end of it?
- What types of *relational assets* – relationships with other organisations and actual people – are important to the organisation's business? What roles do these assets play in the business – for example, as links with employees, suppliers, customers, shareholders, regulatory bodies, other stakeholders? How are these links used in business processes – such as through contracts and other agreements? What value does each type of relational-asset have for the business? By what means does the organisation identify when relational-assets need to be created, or have been changed, damaged or deleted, by the entity at the 'far end' of the link? How are these assets obtained, maintained, monitored and managed through their life-cycle?
- What types of *aspirational assets* – the personal sense of belonging, commitment and shared-purpose – are important to the organisation's business? In what ways do others connect to the business –

for example, morale and commitment of employees, customers' sense of 'belonging' via a brand, or community perception of reputation? To what does the organisation itself belong – for example, to a nation, to an industry, or to the shared enterprise as represented by its vision and role? What impacts do these assets have in business processes such as in HR, productivity, marketing? What value does each type of aspirational-asset have for the business? By what means does the organisation identify when aspirational-assets need to be created, or have been changed, damaged or deleted, by the entity at the 'far end' of the link? How are these assets obtained, maintained, monitored and managed through their life-cycle?

- What types of other *abstract assets* – abstract conceptual entities such as finance, credit and energy – are important to the organisation's business? What roles do these assets play in the business – for example, as access to resources for business processes, as measures of success, as relational factors in transactions? What value does each type of abstract-asset have for the business? How are these assets obtained, maintained, monitored and managed through their life-cycle?

Document each type in the upper rows of the respective segment of the 'assets' column in the framework.

> In business we would often talk about assets and liabilities as if they're different things. But in architectural terms they're actually the same: a 'liability' is an asset that has been assigned a negative value, or is a future promise to deliver that asset. So if you come across references to liabilities, document them as if they're the respective type of asset, but with a rider to indicate the negative valuation.

Finally, some asset-types will only make sense as composites: a paper form, for example, is a combination of physical asset (paper) and virtual-asset (information-record). Wherever practicable, we need to be able to split these into their base-categories, to enable down-to-the-roots redesigns in the more difficult architecture concerns such as disaster-recovery planning; but in some cases such decompositions may not make any sense, and we'll need to document that fact:

- What types of *composite assets* – combinations of any of the above 'atomic' asset-categories – are important to the organisation's business? What are their base asset-categories? In what ways is it possible – or not possible, in practice – to split the composite into its base-categories? What are the consequences of *not* being able to split the composite into its base-categories?

Document each of these as a composite that bridges the respective segments of the 'assets' column.

Next, assess the key types of **location** and their composites:

- What types of *physical locations* and their associated location-schemas – geographic, building-floor etc – are important to the organisation's business? What roles do these locations play in the business – for example, as retail contact-points, manufacturing locations, physical storage, resource sites? What value does each type of location have for the business? How are these locations identified, obtained, maintained, monitored, and managed through their life-cycle?

- What types of *virtual locations* and their associated location-schemas – networks, naming, web-addresses, contact-numbers etc – are important to the organisation's business? What roles do these locations play in the business – for example, as virtual contact-points, as nodes for information routes? What value does each type of virtual-location have for the business? How are these locations identified, created, obtained, maintained, monitored, managed through their life-cycle, and disposed-of at the end of it?

- What types of *relational locations* and their associated location-schemas – such as market-segments, nodes in reporting-relationship trees and social-networks

– are important to the organisation's business? What roles do these locations play in the business? How are these locations used in business processes? What value does each type of relational-location have for the business? By what means does the organisation identify when relational-locations need to be created, or have been damaged or deleted, by the entity at the 'far end' of the link? How are these locations identified, created, maintained, monitored and managed through their life-cycle?

- What types of *aspirational locations* and their associated location-schemas – in particular, the end-nodes of aspirational-assets – are important to the organisation's business? What value does each type of aspirational-location have for the business? How are these locations identified, maintained, monitored and managed through their life-cycle?

- What types of other *abstract locations* and their associated location-schemas – time and time-zones, for example – are important to the organisation's business? What roles do these locations play in the business – for example, as reference-points for measurement of performance? What value does each type of abstract-location have for the business? How are these locations identified, maintained, monitored and – where feasible – managed through their life-cycle?

- What types of *composite locations* – combinations of any of the above 'atomic' location-categories – are important to the organisation's business? What are their base location-categories? In what ways is it possible – or not possible, in practice – to split the composite into its base-categories? What are the consequences of *not* being able to split the composite into its base-categories?

Document each type in the upper rows of the respective segment of the framework 'locations' column, or as a composite bridging the respective segments of the 'locations' column.

And assess the key categories of **events** and their composites:

- What types of *physical events* are important to the organisation's business? What roles do these events play in the business, as input- or output-triggers for routine or exceptional business processes? What value does each type of physical event have for the business? How are these events identified, monitored and managed within an overall life-cycle?
- What types of *virtual events* – messages, signals, data-values – are important to the organisation's business? What roles do these events play in the business, as input- or output-triggers for routine or exceptional business processes? What value does

each type of virtual-event have for the business? How are these events identified, monitored and managed within an overall life-cycle?

- What types of *relational events* – arrivals, departures, contacts, other events in relationships with other organisations and actual people – are important to the organisation's business? What roles do these events play in the business, as input- or output-triggers for routine or exceptional business processes? What value does each type of relational-event have for the business? How are these events identified, monitored and managed within an overall life-cycle?

- What types of *aspirational events* – such as reputation- or public-relations events, or changes to brand – are important to the organisation's business? What roles do these events play in the business, as input- or output-triggers for routine or exceptional business processes? What value does each type of aspirational-event have for the business? How are these events identified, monitored and managed within an overall life-cycle?

- What types of other *abstract events* – such as cycles of time – are important to the organisation's business? What roles do these events play in the business, as input- or output-triggers for routine or exceptional business processes? What value does each type of abstract-event have for the business?

How are these events identified, monitored and managed within an overall life-cycle?

- What types of *composite events* – combinations of any of the above 'atomic' event-categories – are important to the organisation's business? What are their base event-categories? In what ways is it possible – or not possible, in practice – to split the composite into its base-categories? What are the consequences of *not* being able to split the composite into its base-categories?

Document each type in the upper rows of the respective segment of the framework 'events' column, or as a composite bridging the respective segments of the 'events' column.

Functions and services

The ***purpose*** of this section is to establish what the organisation does, and the skills and experience needed to do it.

On the ***people*** side, the skills and people-contacts you'll need will be much the same as for the previous section, such as those for routine business-analysis.

For ***planning***, most of the information you'll need would again come from those 'insiders'; much of it resides only in people's heads, and often you'll find you're the first person to write it all down. The aim here is to define core content for the upper rows of the remaining two framework columns, 'function' and 'capability'.

The ***practice*** would be based on standard business-analysis.

And ***performance*** would again be measured in project-completion terms, as specified by the project sponsor.

Services

For this section it's useful to take a 'service-oriented' view of the enterprise, and assert that *everything* in an enterprise delivers a service. As the ITIL v3 specification puts it, "Customers do not buy products: they buy the satisfaction of particular needs". And we satisfy those 'particular needs' through the services we provide. In that sense, products are proto-services that provide the end-customer with a means to deliver a self-service: for example, a vacuum-cleaner provides the service of cleaned floors.

This gives us a means to understand the mutual role of enterprise functions and capabilities, because a service is a structured combination of function and capability. One way to model at a high level this primacy of services is through a ***Results Logic Diagram***, which constructs a trail of derivations from the end-result of the organisation's activities – such as in the FEAF specification's term 'services to citizens' – back through the layers of internal services and their results, to the core functions of the organisation.

A mild warning: it's very easy to misuse a

> Results Logic Diagram to justify the existing structure of the organisation – which may be fair enough, but that's not really the point here!
>
> What we want is a gap-analysis from 'as-is' to 'to-be', to identify potential for useful change. But if we start the Results Logic from the 'as-is', by definition there would be no gap, and hence no real point – other than as a public-relations exercise, perhaps. Instead, first do it strictly as a 'to-be' analysis: imagine that, given a perfect world, what business-functions and services would you need, to support all of the steps in the 'results logic'? Once you have that, you can link across to the 'as-is', to develop that gap-analysis that you need.

To build the diagram, start from the enterprise vision as identified earlier:

- Given the overall priority – the vision – for the enterprise, what are the key *client-results* for the organisation's stakeholder-groups – its equivalent of 'services to citizens'?
- For each client-result, what are your respective *enterprise-results* – the measures or metrics by which the

organisation confirms that it has achieved its own outcomes *and* the client outcomes?

- For each enterprise-result, what is the tree of *intermediate-results* – the linked outcomes of subsidiary Missions?
- For each set of intermediate-results, what is the *premise* – the set of core assumptions defining the role of a service-group?
- For each premise, what is the *service-group* within the organisation that delivers that service or Mission?

Given that hierarchy of results, we can then start to model the matching hierarchy of services that would deliver those results. From there, we can identify common functions that underpin the services, and the differing capabilities that actually deliver them.

Functions

The terms ‘function’, ‘capability’ and ‘service’ are often blurred together, but perhaps the best way to understand ‘function’ is to think of it in mathematical terms: a function such as *a=func(x,y)* implies that something is returned – often in changed form – from the activity of the function. So to look for business functions, look for where something *happens* in the business – in particular, where something is changed. Services describe *what* it is that the business delivers; functions describe what it *does*.

Because of this, a ***Function Model*** is one of the most useful tools in the entire enterprise-architecture. We can use it as a single-page summary of the business *as* a whole; we can use it as a base-map for all manner of other cross-references, from project-touchpoints to costings to information-systems and process-flows; we've often seen managers use it as a way to show new recruits where their own work fits in with that of everyone else.

> This is the one architecture artefact that makes immediate sense to everyone: it *matters.*
>
> If you're looking for quick wins and instant credibility – which you usually will be, at this earliest stage of the game – then creating even a simple Function Model should be one of your highest priorities. You'll need also to have done at least some of the work above, to help you make sense of the information as an architect, but this is where other people will tend to sit up and start to take architecture seriously.
>
> For a first-cut you'll probably only need a two-tier version, with perhaps a few hints towards the more detailed third-tier – more on that in a moment. But you should be able to get together something useful and meaningful within even a couple of days'-worth of

trawling: so not only is it a valuable model, it doesn't cost much to do, either.

The model is a layered list of business functions, laid out as a visual summary of what the enterprise does. The aim here is to create a model that remains much the same as long as the organisation does that kind of work, so it needs to be independent of any current assumptions about business structure. So we start with the 'to-be' model, the description of the idealised enterprise, and then later work backwards to the 'as-is' to give us our gap-analysis.

There are no set rules about layout: the 'best' model is whatever best describes the business. Whichever way we do it, though, it's usually organised as a nested hierarchy, with three or four tiers of functions:

- ***Tier 1***: major categories of business functions – key aspects of what the organisation 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.

There's rarely enough space on a diagram for the tier-4 functions, so they're usually listed within a supporting text document.

> 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.

To identify candidate functions and activities, trawl through any available sources for information about points where business processes start and end, or wherever something is changed:

- 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
- the organisation's Annual Report: almost by definition, this is supposed to list every major category of business activity
- 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
- business data-models: look for the implied functions that would create, read, aggregate, update or delete the information-items

> The other obvious source for information is through conversations with appropriate staff – though beware that they're likely to want you to list every one of their tasks as a top-level business function!

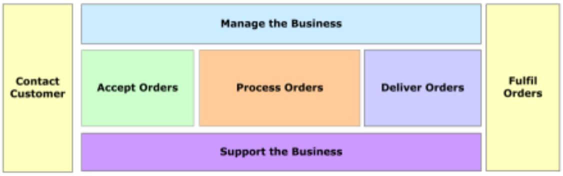
Every business function *does* something, so each function-label on the model should include a verb. For tier-1 and tier-2 functions you can get away with using abstract verbs such as 'Provide' or 'Manage', but for tier-3 or tier-4 especially, you need to use more proper descriptive verbs such as 'Receive', 'Assess' or 'Monitor'.

For ***tier-1 functions***, ask:

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

Aim to define some six to twelve tier-1 functions. These will usually be evident in the structure of the enterprise:

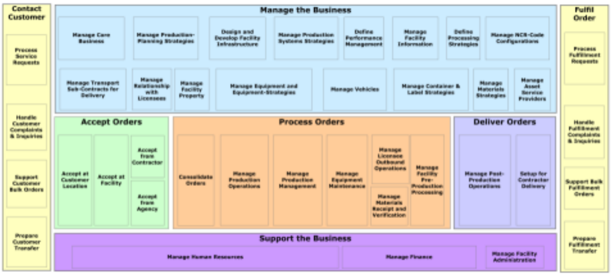
for example, every organisation will have a set of contact-points for customers and other stakeholders, a set of core business-processes, a layer for strategy and management, and a set of business-support functions such as HR and finance.



Functions: tier-1 example

Functions at tier-3 and below are relatively easy to identify in the trawl through documents and the like. But ***tier-2 functions*** are often less obvious at first, and we'll need to derive them from natural clusterings of tier-3 functions, such as implied by higher-level entries in the org-chart, or groups of functions that reappear together in different locations. So 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?


Contact Customer
Process Service Requests
Handle Customer Complaints & Inquiries
Support Customer Bulk Orders
Prepare Customer Transfer
Manage the Business
Manage Core Business
Manage Production-Planning Strategies
Design and Develop Facility Infrastructure
Manage Production Systems Strategies
Define Performance Management
Manage Facility Information
Define Processing Strategies
Manage NCR-Code Configurations
Manage Transport Sub-Contracts for Delivery
Manage Relationship with Licensees
Manage Facility Property
Manage Equipment and Equipment-Strategies
Manage Vehicles
Manage Container & Label Strategies
Manage Materials Strategies
Manage Asset Service Providers
Accept Orders
Accept at Customer Location
Accept at Facility
Accept from Contractor
Accept from Agency
Process Orders
Consolidate Orders
Manage Production Operations
Manage Production Management
Manage Equipment Maintenance
Manage Licensee Outbound Operations
Manage Materials Receipt and Verification
Manage Facility Pre-Production Processing
Deliver Orders
Manage Post-Production Operations
Setup for Contractor Delivery
Support the Business
Manage Human Resources
Manage Finance
Manage Facility Administration
Fulfil Order
Process Fulfillment Requests
Handle Fulfillment Complaints & Inquiries
Support Bulk Fulfillment Orders
Prepare Fulfillment Transfer

Functions: tier-2 example

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.

Every business function will use assets; take place in locations; be linked by events; and be impacted by laws, rules, regulations, standards, constraints and other business reasons or decisions. So these in turn form a useful set of cross-checks for each function:

- What *assets* does the function use, create change, update, delete, destroy? What category of asset, or combination of asset-categories, is involved in each case – physical, virtual, relational, aspirational, abstract? In what way does the function change the respective asset? In that sense, what category of function, or combination of function-categories, is involved in each case – physical, virtual, relational, aspirational or abstract?

- What *locations* are related to the function? What category of location, or combination of location-categories, is involved in each case – physical, virtual, relational, aspirational, abstract?
- What *events* trigger or are triggered by the function? What category of event, or combination of event-categories, is involved in each case – physical, virtual, relational, aspirational, abstract?
- What *reasons* or decisions apply to or impact on the function? What category of reason, or combination of reason-categories, is involved in each case – rule-based, analytic, heuristic guideline, or principle?

There's more detail on this, and on how to tackle the other two tiers, in the 'Practice – Services and Functions' chapter in the companion volume *The Service Oriented Enterprise.*

And there's also a matching Visio stencil and template for building a Function Model at tetradianbooks.com/services-model[21] .

[21]http://tetradianbooks.com/services-model

Again, keep it simple – at this stage this is only a first pass, not a full detailed assessment. Gather whatever level of information seems useful, and document it as links between entities in the architecture repository.

Capabilities

Functions need to be linked to capabilities in order to deliver services. A function describes a required change, but on its own has no way to enact that change; and on its own a capability has no function – literally so. Yet we do need to assess each separately, because different combinations of capability and function deliver different services.

> The specification for the Archimate architecture-notation standard includes a useful illustration of this in its insurance-industry example.
>
> - At the higher level, the client sees a single insurance-claims service, implemented in several different ways – different detail-combinations of function and capability, but with the same *overall* business-function – such as a web-interface, a call-centre, a retail storefront, a personal visit by a claims-adjuster, and so on.

- At the back-office, there are two different services with, again, similar business-functions: one handles claims for below $1000, the other function manages any larger claim. The capabilities used to implement each would be different, because different competencies apply: the lower-value claims can be handled via a rule-based approach, implemented by IT, or by a trainee, perhaps; the higher-value claims require higher levels of skills and experience, hence probably a human process perhaps backed up by an IT-based decision-support system.

Same overall function, same overall business service, but with different business-rules, different asset-types, different events or even different locations, will often imply a different combination of capabilities.

This points to the last of the framework columns, and the last of the content that we need to populate in this first pass. To do this, we need to note that whilst Zachman describes capability as ‘who’, it’d be more accurate to describe it as the real ‘how’ – the competencies, rather than the activities – of a business-service.

We’ve left this till last because it’s often the most difficult of the lot. The reason is that capabilities inherently merge within themselves *two* sets of categories:

- the set used mainly for *assets*, but also for *functions*, *locations* and *events*: physical, virtual, relational, aspirational, abstract
- the set used mainly for *reasons* and decisions: rule-based, analytic, heuristic, principle-based

The former set indicate how and to what the capability is applied; the latter set represent the required skill-levels:

- *rule-based*: no real skill required, can be implemented via training, or built-in within software or machine function
- *analytic*: requires analytic competence and usually some practical experience; may be built into software, but at significant cost
- *heuristic*: requires genuine skill and practical experience for context-specific interpretation; autonomous software systems possibly but at high cost and high complexity; IT is usually more effective in a decision-support rather than decision-making role
- *principle-based*: requires high degree of skill and ability to cope with inherent uncertainty; not yet feasible to build any IT-based system with this type of capability

As mentioned earlier, there is some natural alignment between these two sets of categories, but not enough that they could fully substitute for each other. So in effect we need to map capabilities using a matrix between the two sets: a clerk might do only simple rule-based decisions on virtual data, which could well be handled better by a software-driven process, but a maintenance engineer or machinist might require a high degree of skill with physical objects, for example, and be very hard to replace by IT alone.

> A moment's calculation would show that there are at least twenty cells in that matrix: too many to make it worthwhile to list all the possible assessment-questions here. So for this column, adapt and combine the previous questions on assets, locations and events) with an assessment of skill-levels as above, to arrive at appropriate questions for each cell.
>
> The skill-level assessments will also indicate which capabilities can be implemented by IT, with relative ease, or more expensively, or only with inordinate difficulty, or not at all. With luck, this should dissuade all but the most insanely over-enthusiastic IT-advocates from attempting to build 'solutions' in contexts for which by definition they cannot feasibly work – which should help to defuse some of the anger and angst so often locked up in the bitter relationship between business and IT!

From this, build a capability-map for the enterprise:

- What capabilities are implied as required by the Results Logic Diagram? How would you categorise each capability, in terms of asset-type acted upon and required skill-level?
- What capabilities are implied as required by the Function Model? How would you categorise each capability, in terms of asset-type acted upon and required skill-level?
- What capabilities are implied by each key asset-type, location and event identified in the previous assessments here? What is the required skill-level in each case?
- What capabilities are implied by each key reason, decision, constraint, standard or suchlike identified in the previous assessments here? What is the required skill-level for appropriate decision-making in each case? What asset-types would be involved in each case? What events or functions would call for this capability?
- Does the capability need to vary in different locations? If so, what location-category is implied in each case?

Document the results of these assessments in the upper rows of the 'capability' column in the framework, together with any cross-links as required.

This then completes the population of the base 'holograph' in the framework – the pre-built architecture that we'll need in place before we can do any work under proper architecture governance. The final actions for this step are to define all the governance processes and artefacts that we'll need for subsequent architecture cycles, and the methods, tools and techniques by which we will share and communicate the results of that work.

Architecture governance

The ***purpose*** of this section is to set up the formal governance processes for the enterprise-architecture capability.

> I've placed this part at the end of this step, rather than at the beginning, because up till this point there won't have been that much need for *architecture* governance – the work will have been done under normal *project* governance instead. But once we've proven the value of the notion of architecture – which we should have done by now, especially with the Function Model – we'll have to tackle the need for governance of the architecture itself, in its own terms.
>
> When you revisit this stage – which you should do on a regular basis, probably at least once

> or twice a year – you would of course already have all of the governance documents in place. For a revisit, it's probably best to review them *first*, before doing any of the other work of this stage. But for the first pass, we won't need to do it until here.

Note that, from the business perspective, part of that governance is formed by the processes through which you engage others in the architecture 'conversation'.

On the ***people*** side, this is where we move out from desk-based business-analysis and start to engage with people in general. The range of people you'll need to work with will depend on the chosen scope – a smaller pilot, at the start, but eventually out to the entire enterprise – but people-skills in general will begin to come much more to the fore here, rather than the analytic skills that have been the main requirement so far.

For ***planning***, you'll need some kind of 'communication plan' as to how you will engage with the various stakeholders, and a formal process or checklist for defining the required governance.

The ***practice*** would be based on that communication plan, and the 'governance for governance' process.

> If you're familiar with TOGAF, much of what happens here is a simplified version of TOGAF's Preliminary Phase, combined with the

> parts of its Phase A 'Architecture Vision' that deal with overall setup for architecture rather than the details for a single iteration.
>
> One important difference is around scope. TOGAF assumes that enterprise architecture will always be a major effort, encompassing the entire enterprise – or all of its IT, anyway. But that isn't what happens in practice: if nothing else, you won't get the funding until you've proven the business value. Instead, pick a small area of the business as a pilot, and do your initial governance-planning and the like for that area alone. Once you've proven the value of the work, you'll need to expand the scope – but at the start, keep it simple, and keep it small.

And whilst ***performance*** would again be measured in project-completion terms, as specified by the project sponsor, perhaps a more important metric would be the indications of acceptance and take-up of the architecture by the broader business community – of which the Function Model would probably be the first example.

Creating engagement

For architecture to be useful, people have to be willing to *use* it. And they'll only be able to use it if they know it exists – which is why it's essential for architects to communicate what they do. Hence the importance of a 'communications plan' and the like – and from there, of tools such as wikis and intranet websites to put that plan into practice.

> Chapter 36 'Architecture Deliverables' in the TOGAF 9 specification includes an all-too-brief description of the role and content of an architecture Communications Plan. As that summary states, mapping out the stakeholders and their communication needs is something we must do right at the start, in Phase A of the cycle – but we then need to *use* that plan, and keep the dialogue going beyond the nominal scope of any single architecture cycle.
>
> The plan is only the start-point of engagement – and that engagement *matters*. Make it happen!

Communication and engagement is the other side of the architect role. The catch is that it's mostly about *people*, not abstract analysis – and that demands a completely different skillset from that we've used so far. For many enterprise architects, dealing with all of the interpersonal politics and other 'people-stuff' is *hard* – but all of our previous work will be in vain unless we *do* tackle it. All those analyses and models and the like may be the backbone of the work, but the real architecture – the place where the architecture is literally 'real-ised' – is always through people. The dialogue *is* the architecture.

> By the way, it's wise here to *expect* that people will tell you "you've got it all wrong", or worse, with disparaging comments about your competence – or lack of it – thrown in for good measure. Painful though this may be, it's a natural consequence of the processes by which people learn in a collective space: meaning *emerges* from a collective conversation about the 'unknown', "I don't know what I want, but I'll know it when I see it". Hence we need something – almost anything, really – to initiate and guide that conversation. Which means that at the start of the conversation, the usual response is "I don't know what I want, but I know that isn't it" – or, in short, "you're wrong".
>
> So don't worry about it, and above all don't over-defend your work: *the architecture is the dialogue*, not just the end-result. Ask their advice, ask what you could do better. That's how engagement happens: and when others become passionate about the architecture, 'owning' what they've co-created with you, you'll know you've succeeded.

Yes, it can be intensely frustrating – insanely, hair-tearingly, mind-bendingly frustrating – when other people 'just don't get it' in relation to the architecture. But that's *our* problem, not theirs: if we want others to 'get it', it's up to *us* to provide conditions under which they *can* 'get it', where they *can* see the benefits of working with others in the enterprise in an architectural way. Some typical reasons why they don't 'get it' would include:

- "this architecture stuff doesn't make sense" – so *we* need to find ways in which it does make sense for them
- "it doesn't apply to our context" – so *we* need to show where it applies, and why
- "it's just theory" – so *we* need to show how it works in practice
- "no-one bothered to ask us" – so *we* need to show where it *does* take their experience into account
- "it's working now, why do have to change it?" – so *we* need to justify, in their terms, the requirements behind the change
- "we don't have time for this stuff!" – so *we* need to show why they don't have time to *not* do 'this stuff'
- "why should we bother to help anyone else?" – so *we* need to show why it's in their own interest to do so

We can't make the architecture happen on our own: an enterprise is about *shared* goals, *shared* vision. So we need to understand that whilst we may have nominal responsibility for the architecture, we don't *possess* it: it belongs to everyone in the enterprise. The architecture will only happen when people feel that they 'own' it too: not imposed on them from above, but something in which they themselves are co-authors, co-creators. That's what engagement is about; that's what it's *for*.

This is one of the reasons why the Function Model is so important: it describes the enterprise in a way that includes *everyone*. Another reason is that it provides a basis for *story* – a story of the enterprise as a whole, as an ecosystem. Each business-process is the thread of a story; each use-case and transaction is a story; with the Function Model we can show how these stories literally weave their way across the organisation and the enterprise, touching different business functions and services and capabilities, using the enterprise assets in different ways and in different locations, triggering other business events for different business reasons. The architecture *is* a story: so to help others make sense of the architecture, tell it *as* a story – a narrative which includes them *in* the story.

> Storytelling is a skill in itself, and one whose business applications and business value are only now beginning to be better understood. One resource we've found useful for this, and for practices on 'narrative knowledge' in gen-

> eral, is the Australian consultancy Anecdote: see their website at www.anecdote.com[22] .

Typical techniques for engagement include:

- publishing models and other artefacts – typically via the architecture-toolset
- running your own intranet-website, including wikis and other facilities for feedback – again, some architecture-toolsets will support this requirement 'out of the box'
- seminars and 'public' presentations across the enterprise
- workshops for engagement with operations staff, particularly those with direct involvement in front-line business processes
- maintaining a 'watching-brief' relationship with senior staff, strategists, project-managers and others involved in change-management
- building working-relationships with those responsible for other 'pervasive' themes such as security, privacy, quality, health and safety, ethics and environment

It's unlikely that you would use each of these approaches in a first pass of the architecture, but it's worthwhile

[22]http://www.anecdote.com

considering all of the options from the start, as you'll certainly need them later on.

> In the 'Completions – Closing the loop' chapter in the companion volume *Bridging the Silos* there's more detail on tactics for linking with knowledge-management, change-management, quality-management, communities of practice and other probable allies within the enterprise to help us with these aspects of engagement.

Architecture is a *dialogue*, not a monologue: so perhaps most of all, listening is more important than talking. People are likely to listen to what we have to say only if they feel they too have been heard.

From this, two-way communication creates engagement with people; and engagement in turn creates governance. More precisely, it creates the kind of governance in which people *want* to be involved, because the work has meaning to them. And that's what we're aiming for here: enterprise architecture only becomes the architecture *of* the enterprise when the enterprise in general is engaged in every aspect of its creation and use.

Creating the architecture capability

For the architecture development so far, we've run it as a stand-alone project, or embedded as part of another project.

We've reached the first maturity-level, and we've presented the results, as a kind of proto-architecture. Everyone – we hope! – is happy with what we've done: we've proven that architecture is indeed of value to the enterprise.

But that's it: we're done. We can't do it this way again: it works well enough for a first-pass, but it's not sustainable – especially in terms of its governance. To take it further, we need to set it up as a proper enterprise capability, with proper governance and so on.

> What follows will be sufficient to set you up for the next step in architecture maturity, but it doesn't stop there, of course. Each time you change the formal scope of your architecture, or get ready to move up to another maturity-level, you'll need to revisit this work, to review and update the architecture capability and its governance.

Depending on where and at what level you want to move on to next, there can be quite a bit of work to do at this point – almost a mini-project in its own right, in fact. But whilst there's fair bit to do – including a sizeable amount of paperwork, no doubt – it's all straightforward, and for the most part already documented in the TOGAF specification and other architecture descriptions: you'll need to tailor it to your own context, but that's about all that's required. We can summarise it as follows:

- you'll need sufficient *funding*, equipment, work-space and other *resources*
- you'll need the right *people*
- you'll need an appropriate *toolset* and repositories and registers in which to record and maintain that 'body of knowledge about enterprise structure and purpose'
- you'll need formal *governance*, including formal *authority* to engage others in the work
- you'll need to define *governance-artefacts* to be created, updated and used within the architecture cycle itself

It's not really feasible here to describe how to obtain the right ***funding*** and ***resources***: that'll depend on the nature of the organisation, its organisational culture and structure, on the scope to be covered by the architecture capability, and a whole host of other context-specific factors. All we can say for certain here is that it'll have to be done, and probably done first. But having done all the previous work of this step, you should now be able to *prove* the potential business-value of architecture – which means it should be a lot easier than trying to argue the business-case from scratch.

Finding the right *people* and bringing them on board will, again, be somewhat context-dependent. But by now – such as from the comments earlier, in the section 'People –

governance' in chapter *Preparing for architecture* – you'll have a better idea of the range of skillsets that'll be needed for the next stage: so again this should be easier than starting to build the right team from scratch.

> Chapter 52 'Architecture Skills Framework' in the TOGAF9 specification summarises the skillsets needed for IT-architecture. It doesn't extend much beyond that scope, but it does at least suggest the probable skills and levels of experience needed at each maturity-level for true whole-of-enterprise architecture. Recommended.

The same applies to ***toolsets*** and repositories: we explored those issues in the section 'Planning – frameworks' in chapter *Preparing for architecture*. You should review those facilities and capabilities there as part of this activity – for example, you'll soon be needing a purpose-built architecture toolset if you're not already using one, and you'll need to include a dispensations-register as part of the repositories for all further architecture work. But it's all straightforward enough: it just needs to be done – and paid for, which may be the hardest part!

The main work here will be around ***governance*** – in particular, developing and documenting all the requisite procedures – and ***governance-artefacts*** used within the architecture-cycle.

In the companion volume *Bridging the Silos*, the chapter 'Completion – Architecture-artefacts' provides summaries of the purpose and content for most of the documents used in the modified Architecture Development Method described earlier, in chapter *Preparing for architecture*. In the lists below, documents described or referenced in *Bridging the Silos* are shown with a number-sign (#).

There's also a whole section in the TOGAF 9 specification – the Architecture Capability Framework – that deals with governance and related documents, whilst its Chapter 36, 'Architecture Deliverables', provides brief summaries of the purpose and content of a whole range of typical architecture-artefacts. These are fairly comprehensive, but unfortunately many of the descriptions are scrambled by TOGAF's obsessive IT-centrism – for example, its insistence that architecture-governance is a subset of IT-governance – so we need to do a certain amount of translation before we can use them in real whole-of-enterprise architecture. (The specification is further scrambled in that it also includes descriptions of several documents that relate to *implementation*-governance rather than *architecture*-governance – which don't belong there at all.) In the lists below, documents described or referenced in the TOGAF specification are marked with an asterisk (*).

Typical governance-procedures and related documents for overall architecture governance include:

- *Architecture Governance* # * and *Architecture Board* * – defines the governance processes for the architecture itself
- *Architecture Charter* # * – identifies and defines the formal authority and responsibilities for architecture as a business capability or business unit
- *Architecture Principles* # * – identifies and describes the principles used to govern architectural decisions, with applicable higher-level principles included by reference
- *Architecture Standards* # * – identifies and describes the formal and other standards used to guide architecture decisions and designs, with external standards usually included by reference

Formal governance procedures and work-instructions will also be needed for management of the architecture-repository, requirements repository, risks and opportunities register, issues register, dispensations register and glossary and thesaurus. Since it's likely, though, that these will also be shared outside of architecture, their governance will probably need to be beyond the authority of architecture itself.

Typical architecture-cycle governance-artefacts include:

- *Request for Architecture Work* # * – describes the business-questions and context to be addressed in an architecture-cycle
- *Statement of Architecture Work* # * – maintains a record of all architecture work and decisions in the current architecture-cycle
- *Architecture Roadmap* * – overview of proposed changes arising from a large-scale architecture-assessment in ‘classic’ enterprise-architecture
- *Project-plan* or *Migration-plan* # * – overview and/or detailed description of a proposed project or portfolio of projects
- *Architecture Compliance Statement* # * and associated checklists * (for IT only) – asserts the extent to which a proposed implementation complies with the specified architecture, and describes reasons for any non-compliance
- *Architecture Description Statement* # – simplified Architecture Compliance Statement used in ‘hands-off’ architecture (see section ‘Hands-off architecture’ in chapter *What’s next?*)

- *Architecture Position Statement* # – describes architect's recommended response to non-compliance in an Architecture Compliance Statement or Architecture Description Statement
- *Architecture Dispensation Statement* # – describes architect's reasons for permitting a non-compliant implementation, and recommendations for future review and resolution
- *Phase__-completion reports* # – describes the activities and results of an architecture-cycle Phase, and includes stakeholder sign-off for the respective Phase

Governance will also be needed to manage some other concerns such as security and visibility of architecture models, descriptions, frameworks, roadmaps definitions and other products from the architecture work.

You may need other governance processes and artefacts in your own specific context, but these at least should serve as a start.

Resources

- TOGAF (The Open Group Architecture Framework): see www.opengroup.org/architecture/togaf9-doc/arch/[23]

[23]http://www.opengroup.org/architecture/togaf9-doc/arch/

- Adapting TOGAF for whole-of-enterprise scope: see Tom Graves, *Bridging the Silos: enterprise architecture for IT-architects* (Tetradian, 2008)
- Business Motivation Model: see businessrulesgroup.org/bmm.shtml [24]
- Vision, role, mission, goal: see Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008)
- Visioning and purpose in service-oriented architecture: see Tom Graves, *The Service Oriented Enterprise: enterprise architecture and viable services* (Tetradian, 2009)
- SEMPER diagnostic and metric: see Tom Graves, *SEMPER and SCORE: enhancing enterprise effectiveness* (Tetradian, 2008)
- Online version of SEMPER diagnostic: see www.sempermetrics.com[25]
- ITIL (IT Infrastructure Library) and primacy of services: see www.itil-officialsite.com[26]
- Archimate specification: see *Archimate Practically* (Archimate Foundation, 2007) and www.archimate.org[27]

[24]http://businessrulesgroup.org/bmm.shtml
[25]http://www.sempermetrics.com
[26]http://www.itil-officialsite.com
[27]http://www.archimate.org

- Business storytelling and narrative knowledge: see www.anecdote.com[28]

[28] http://www.anecdote.com

Appendix: Glossary

This summarises some of the terms and acronyms we've come across in the book.

ADM: acronym for Architectural Design Method, the methodology used in *TOGAF* to guide development of *enterprise architecture*

ArchiMate: a visual language used to model *enterprise architectures*, developed by Netherlands consortium Telematics

chaos domain: in the sensemaking framework, domain of inherent uncertainty and unpredictability; decisions are guided by *principles* and *values*; represented in the business context by unique market-of-one customisation and by non-repeatable maintenance issues; also useful when deliberately invoked in creativity, in *narrative* and *dialogue*, and in *foresight_techniques such as _scenario* construction

complex domain: in the sensemaking framework, domain of *emergent* properties and non-linear relationships between factors; decisions are derived from heuristics and guidelines; unlike *chaos*, which is inherently uncertain, may often create an illusion of predictability, especially where linear analysis is applied within a short-term, narrow set of assumptions

complicated domain: in the sensemaking framework, domain of complicated yet identifiable cause-effect relationships; decisions are derived from contextual analysis

DyA: acronym for Dynamic Architecture, an *enterprise-architecture* framework developed by Netherlands consultancy Sogeti

effective: ‘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*

efficient: ‘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*

emergence: context within which cause-effect patterns can be identified only retrospectively, and in which analytic techniques are usually unreliable and misleading

enterprise architecture: a systematic process to model and guide *integration* and *optimisation* of the information-technology of an enterprise or (at higher maturity-levels) the entire enterprise

FEAF: acronym for Federal Enterprise Architecture Framework, a framework and methodology developed for *enterprise architecture* by the US government

goal: a specific objective to be achieved by a specified point in time; emphasis on the *physical* or *behavioural dimension* of *purpose*, contrasted with *mission*, *role* and *vision*

mission: a desired capability or state to be achieved, usually within a specified timeframe, and to be maintained indefinitely once achieved; emphasis on the relational and, to a lesser extent, the *virtual dimensions* of *purpose*, contrasted with *goal*, *role* and *vision*

narrative: personalised and often emotive expression or interpretation of knowledge, as history, anecdote or story

optimisation: 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

principle: a conceptual commitment or model, the *conceptual-dimension* equivalent of *value*

purpose: 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*

recursion: patterns of relationship or interaction repeat or are 'self-similar' at different scales; permits simplification of otherwise complex processes

role: a declared focus or *strategic* position within the 'world' described by a *vision*; emphasis on the *conceptual* and, to a lesser extent, the relational dimensions of *purpose*, contrasted with *goal*, *mission* and *vision*

scenario: an imagined future context, developed for the purpose of understanding both the present context and options for action in the future context

simple domain: in the sensemaking framework, domain of certainty and known cause-effect relationships; decisions are predefined by laws, rules and regulations

strategy: 'big picture' view of an action-plan for an organisation to implement a *purpose*, usually emphasising its *vision*, *role* and *mission* components; contrasted with the *tactics* required to execute the plan

tactics: detailed *missions*, *goals* and other step-by-step activities to execute a *strategy*, or some segment of an overall strategy

TOGAF: acronym for The Open Group Architecture Framework, an IT-oriented framework and methodology for *enterprise architecture* developed collectively by members of the Open Group consortium

value: an emotional commitment – 'that which is valued', either individually or (in an enterprise context) collectively)

vision: description of a desired 'world', always far greater than any individual or organisation; described in the present tense, yet is never 'achieved'; emphasis on the *aspirational dimension* of *purpose*, contrasted with *goal*, *mission* and *role*

visioning: generic term for the process of identifying, developing and documenting *vision* and *values*, leading towards *strategy* and *tactics*

Zachman framework: a systematic structure for categorisation of models within an IT-oriented *enterprise architecture*, developed by John Zachman