

Whole-of-enterprise architecture

Extending enterprise-architecture beyond IT

Tom Graves : *Tetradian Consulting*
February 2007

© 2007 Tetradian Consulting

Tetradian Consulting
The Coach House
Balkerne Close
Colchester CO1 1NZ
England
[+44] (0) 781 560 6624
info@tetradian.com
www.tetradian.com

Extending enterprise-architecture

- **Audience**
 - enterprise architects, strategists, planners, process architects, technical architects
- **Objective**
 - identify limits of current enterprise-architecture
 - establish criteria for extended architecture
- **Agenda**
 - describe existing architecture frameworks
 - identify limitations and problem-areas
 - identify requirements for extended architecture
 - describe suggested architecture framework

The current enterprise-architecture frameworks are valuable for reducing IT costs and complexity. But their over-emphasis on IT, almost to the exclusion of everything else, can become a hindrance when we need to extend those architecture principles to the whole of the enterprise.

So the purpose of this slidepack is to identify the limitations of existing enterprise-architecture frameworks, and to establish criteria for whole-of-enterprise architecture.

It's aimed at people whose work revolves around balancing 'big picture' issues with day-to-day practice – roles such as enterprise-architects, process-architects, strategists and planners – but it should also be useful for senior business managers.

Business need for enterprise-architecture

“Increasing the scope of Enterprise Architecture to encompass more disciplines increases the benefits to be gained:”*

- **EA = Technical Architecture:** reduce IT complexity and costs
- **EA = Enterprise-Wide IT Architecture (EWITA):** support collaboration among different parts of the enterprise
- **EA = EWITA + Business Architecture (BA):** increase enterprise agility and alignment with business strategy

Business now needs a new level of architecture maturity:

- **EA = integration across entire enterprise:** increase adaptability, resilience, management of opportunity / risk; increase synergies between processes and partners

* Bredemeyer et al., “Enterprise Architecture as Business Capabilities Architecture”, <http://www.bredemeyer.com>, slide 10



3

As Dana Bredemeyer explains in his influential article, enterprise architecture grew out of a business need to manage the growing cost and complexity of IT systems.

As EA maturity developed, it provided new insights into re-purpose and re-use, and in how to link systems together in new ways – for example, to provide a real-time view of stock levels on a web-based e-commerce system. More recently the scope has widened again, with a belated awareness that IT developments must be linked much more strongly to business needs and business strategy.

The challenge now is to extend this integration to all aspects of the business and its processes – and even across complex multi-partner enterprises.

Enterprise architecture frameworks

Some existing types of architecture framework:

- **Grid-type**
 - Zachman framework [US]
 - Dynamic Architecture framework (DyA) [Netherlands]
- **IT-hierarchy**
 - Federal Enterprise Architecture Framework (FEAF) [US]
 - The Open Group Architecture Framework (TOGAF) [US/UK]
- **Hybrid process/IT**
 - Architecture of Integrated Systems (ARIS) [Germany]
 - ArchiMate [Netherlands]

There are quite a few enterprise-architecture frameworks around these days. Some have a kind of broad ‘fill in the boxes’ approach; some work their way down the technical layers of IT; and others start from more of a process background, linking across from there to IT. These are some of the better-known examples of each type.

Grid frameworks: Zachman [1]

	DATA	How	FUNCTION	How	NETWORK	Where	PEOPLE	Who	TIME	When	MOTIVATION	Why
SCOPE (CONCEPTUAL)	Set of things important to the business	Set of processes the business performs	Set of locations in which the business operates	Set of resources important to the business	Set of products important to the business	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
Planner	GP HW - Elements of Business Logic	Function - Elements of Business Process	Place - Maps Business Location	People - Maps Business Resources	Time - Maps Business Goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
ENTERPRISE MODEL (CONCEPTUAL)	Set of business logic	Set of business process	Set of business location	Set of business resources	Set of business goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
Owner	Set - Business Logic Set - Business Relationship	Proc - Business Process IO - Business Resource	Place - Business Location Set - Business Location	People - Business Resources Set - Business Resource	Time - Business Goals Set - Business Goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
SYSTEM MODEL (LOGICAL)	Set of data model	Set of business process	Set of business location	Set of business resources	Set of business goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
Designer	Set - Data Model Set - Data Relationship	Proc - Business Process IO - Business Resource	Place - Business Location Set - Business Location	People - Business Resources Set - Business Resource	Time - Business Goals Set - Business Goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
TECHNOLOGY MODEL (PHYSICAL)	Set of physical data model	Set of business process	Set of business location	Set of business resources	Set of business goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
Builder	Set - Physical Data Model Set - Physical Data Relationship	Proc - Business Process IO - Business Resource	Place - Business Location Set - Business Location	People - Business Resources Set - Business Resource	Time - Business Goals Set - Business Goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
DETAILED REPRESENTATIONS (PHYSICAL CONTENT)	Set of data model	Set of business process	Set of business location	Set of business resources	Set of business goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
Sub-Contractor	Set - Data Model Set - Data Relationship	Proc - Business Process IO - Business Resource	Place - Business Location Set - Business Location	People - Business Resources Set - Business Resource	Time - Business Goals Set - Business Goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues
FUNCTIONING ENTERPRISE	Set of business logic	Set of business process	Set of business location	Set of business resources	Set of business goals	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues	Set of business goals that the business pursues

Covers 'what', 'how', 'where', 'who', 'when', 'why'

Source: <http://www.zifa.com>

The Zachman framework is the 'granddaddy' of the field, first released almost twenty years ago. It's a straightforward grid, with five layers from most abstract ('planner') to most concrete ('sub-contractor'), and six columns of data, function, network, people, time and motivation, representing the questions 'what?', 'how?', 'where?', 'who?', 'when?' and 'why?' respectively.

It's close to being the standard reference for the field: every other system maps to it, and uses it as a check-list for completeness.

Grid frameworks: Zachman [2]

- **Strengths**

- systematic: covers entire scope of IT concerns
- layered: from abstract ('planner') to concrete field-level
- standard: used as reference-base by most other frameworks

- **Concerns**

- is only a reference-framework - no defined methodology
- is IT-centric
 - all examples are IT-based
 - no reference to process, people, 'things' etc except in IT context
 - no recognition of non-IT-based knowledge
- not well-suited to complex multi-enterprise contexts
 - assumes simple single-organisation hierarchy of interest

What's not so useful is that it's only a reference-framework. We can't say "We're using the Zachman methodology", because there isn't one: all there is is the grid, which in practice is just a check-list of possible models. The danger – which we often see in practice – is that organisations try to 'fill in the blanks', populating every possible box in the grid with an appropriate model: it's an excellent academic exercise, of course, but far too expensive in every sense for the realities of business.

Another problem, which we'll also find in all existing frameworks, is that not only is there an over-emphasis on technical minutiae, but information-technology is treated as the only source of business knowledge – neglecting the human derivation of meaning on which organisations ultimately depend. More on this later.

And like most other frameworks, Zachman places the organisation as the centre of the world – not in the world. This may seem subtle, but it makes it much harder to map multi-enterprise processes and exchanges – the kind of complex world that large organisations now need to manage.

Grid frameworks: DyA [1]

	Business objectives							
	Business architecture			Information architecture		Technical architecture		
	Prod/ service	Process	Orga- nization	Data	Appli- cation	Middle- ware	Plat- form	Net- work
General principles								
Policy directives								
Models								

DyA grid framework

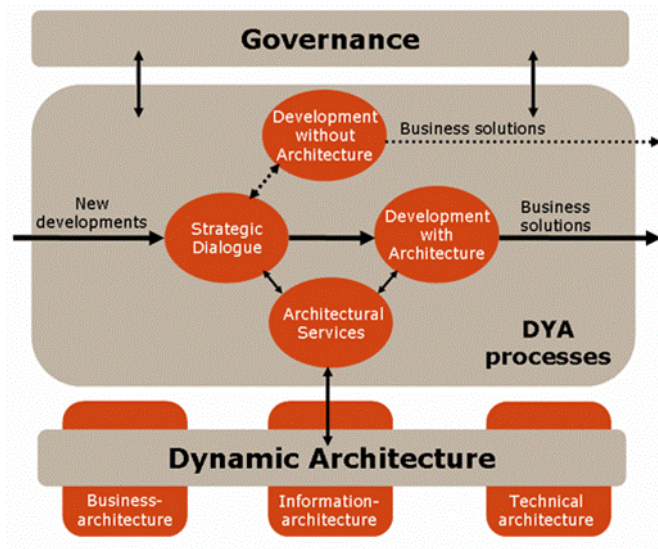
Source: http://eng.dya.info/Home/dya/what_is_dya/architecture_framework.jsp

Dynamic Architecture, or DyA, is another grid-framework, from Dutch consultancy Sogeti, part of the CAPGemini group.

Like many of the European frameworks, this one is aware of a wider world than solely IT, but still places most of its emphasis on the technical minutiae – though oddly only at the abstract level of models. It also covers additional levels above those of Zachman, namely general principles and policies.

But like Zachman, it's still something of a fill-in-the-boxes exercise – though their methodology does emphasise the need for 'just enough, just in time' architecture.

Grid frameworks: DyA [2]



Visual summary of DyA methodology

Again unlike Zachman, there is a proper methodology associated with DyA, although it applies mostly at a rather abstract level.

One very good feature, not seen in most other methodologies, is the systematic management of ‘Development without architecture’ – in other words the projects and systems that must, for various operational and other reasons, be permitted to be non-compliant with the architecture.

Grid frameworks: DyA [3]

- **Strengths**

- covers all high-level IT
- grid-based but includes well-defined methodology
- governance includes management of 'Development without architecture' (i.e. necessarily non-compliant systems)
- 'just enough, just in time' approach to architecture devmt

- **Concerns**

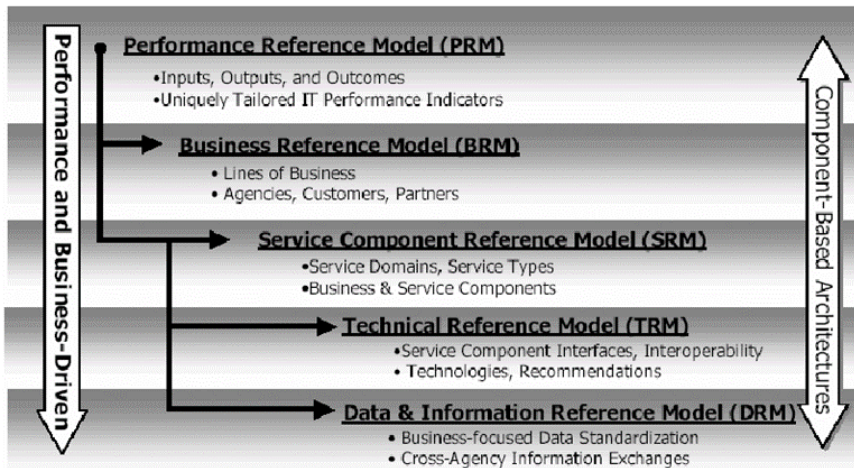
- methodology is proprietary (Sogeti/CAPGemini)
- is IT-centric
 - all examples are IT-based
 - no reference to process, people, 'things' etc except in IT context
 - no recognition of non-IT-based knowledge
- does not address low-level implementation

The most serious limitation of DyA is that it appears to be proprietary to CAPGemini. Some parts of the methodology – such as a very useful enterprise-architecture maturity metric and supporting spreadsheets – are publicly available from Sogeti's website, but the core of the methodology is not.

And despite its brief mention of products and process under the 'business architecture' heading, it is, like Zachman, very definitely IT-centric – but doesn't go into much detail even on that.

So a useful 'upward and sideways' extension of Zachman, but for most purposes that's about it.

IT-hierarchy frameworks: FEAF [1]



FEAF Reference Model hierarchy

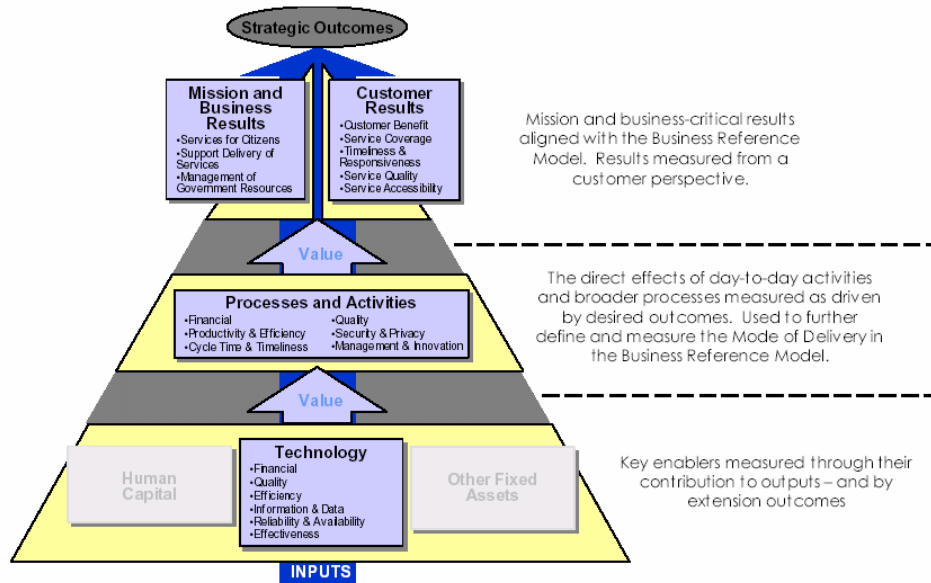
Source: <http://www.cio.gov> and <http://www.whitehouse.gov/omb/egov/a-2-EAModelsNEW2.html>

As is perhaps inevitable in a government model, the US Federal Enterprise Architecture Framework is organised in sets of hierarchies. The framework is open, detailed – perhaps too detailed in places! – and fully published on the US government's CIO.gov website.

The slide here shows the hierarchy of 'reference models': performance metrics, business structures, services, technical structure and, oddly, data-structures beneath technical.

Note the strange sideways jump between business and services, and again from services to technical and data...

IT-hierarchy frameworks: FEAF [2]



FEAF: derivation of business value

...because the ‘derivation of value’ hierarchy shows us what’s missing, and the reason for that sideways jump. Information-technology – here shown as ‘Technology’ – is only one of three legs of the model. The other two legs – the greyed-out boxes for ‘Human Capital’ and ‘Other Fixed Assets’ – are placeholders for future work to link these other domains into a true whole-of-enterprise architecture framework.

IT-hierarchy frameworks: FEAF [3]

- **Strengths**

- systematic structure of interrelated models
- fully-described open methodology
- strongly linked to business drivers

- **Concerns**

- current version is exclusively IT-centric
 - all examples are IT-based
 - no reference to process, people, 'things' etc except in IT context
 - no recognition of non-IT-based knowledge
 - (future versions should expand on 'Human Capital' and 'Other Fixed Assets' sections of the structure)
- aimed primarily at government, not business-enterprises
 - tendency towards bureaucracy-heavy implementation

For now, though, FEAF suffers from the same over-emphasis on IT as in the other frameworks – which is unfortunate, but at least there is an acknowledgement that there is a world beyond IT.

The other key concern with FEAF is that it's so darned big: literally thousands of pages, much of which may not be relevant outside of a US government context. And like early iterations of ISO 9000, it's all too easy for it to become a monstrous paper-trail of bureaucracy, consuming vastly more effort than could be recovered by improved effectiveness.

IT-hierarchy frameworks: TOGAF [1]

Defines hierarchy of four ‘architectures’:

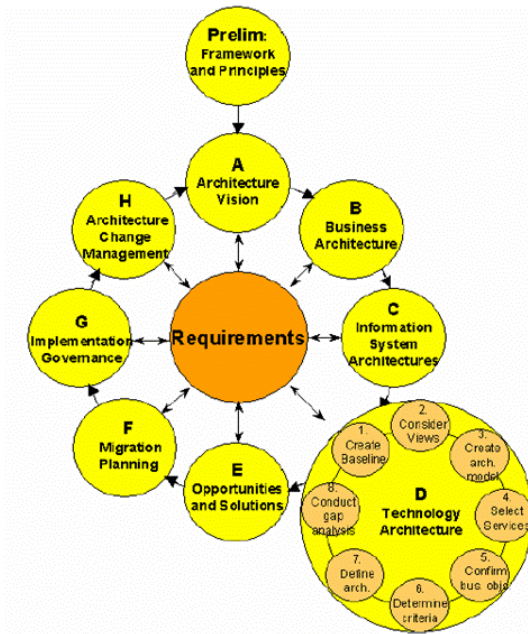
- **Business architecture**
 - business drivers, processes, core requirements
- **Data architecture**
 - structure of ‘information important to business’
- **Applications architecture**
 - consistency of management of data
- **Technology architecture**
 - consistent structure of information-systems technology

Source: <http://www.opengroup.org/architecture/togaf8-doc/arch/toc.html>

TOGAF, The Open Group Architecture Framework, could be described as the ‘open source’ equivalent of FEAF: it’s not just a government exercise, anyone can join in with its creation. Currently in its eighth iteration, it uses the same kind of hierarchy approach as FEAF – though here data-architecture gets a higher ranking.

As a document, it’s also a lot more manageable: less than four hundred pages, more than half of which is the description of the Architectural Design Method, or ADM methodology.

IT-hierarchy frameworks: TOGAF [2]



Visual summary of Architectural Design Method (ADM)

- the core of the TOGAF methodology

The ADM is a well-structured iterative framework, with requirements at the core – which, among other things, indicates the need for a proper version-controlled requirements repository as a key part of any architectural process. After each stage, the repository is updated – which may in turn call for a brief review of a previous stage, until an overall picture is developed.

The beauty of this approach is that it's well suited for what DyA calls 'just enough, just in time': a usable skeleton architecture can be developed very rapidly, with the detail filled in more slowly over time, as needed.

But note that cursory reference to 'business architecture', near the start of the cycle; and, by contrast, the enormous bulge of 'technical architecture'...

IT-hierarchy frameworks: TOGAF [3]

- **Strengths**

- fully-described open methodology (ADM)
- linked to business drivers - requirements as centre of ADM

- **Concerns**

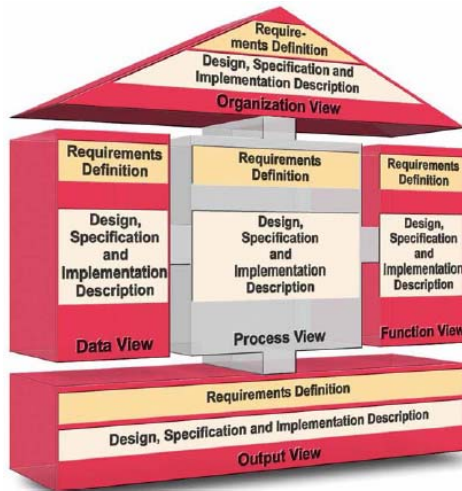
- current version is excessively IT-centric
 - all examples are IT-based
 - no reference to process, people, 'things' etc except in IT context
 - no recognition of non-IT-based knowledge
- business-architecture is poorly described
 - is essentially 'anything not-IT'
 - (major work is in progress to improve TOGAF integration with business)
- over-emphasis on low-level technology-architecture
 - exemplified in 'bulge' at stage D of ADM

...because TOGAF is not so much IT-centric as obsessed by it. There's no balance at all: in essence, 'business architecture' is 'anything not-IT', all but forgotten in a headlong rush to drown in the details of technical minutiae – almost exactly what an enterprise-level architecture aims to avoid, in fact.

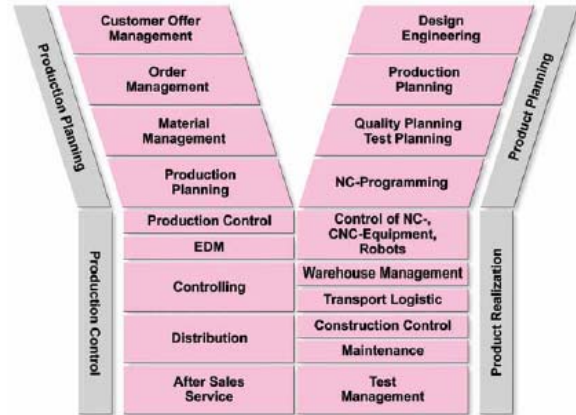
And useful though it is, in practice the ADM is little more than a skeleton of guidelines – it does take a lot of work to flesh it out for any real-world architecture.

To be fair, the TOGAF community is addressing these issues: for example, the next release, TOGAF 9, should include much more emphasis on business integration. But at present, although the framework is excellent for the earlier IT-centric levels of enterprise-architecture maturity, it's actually more of a hindrance than a help when we need to move beyond those limitations.

Hybrid frameworks: ARIS [1]



ARIS: IT architecture



ARIS: process architecture

Source: http://www.ids-scheer.com/international/english/products/aris_design_platform/94053

By contrast to their US counterparts, European enterprise-architecture frameworks have tended to start from product and process, with IT as the afterthought. ARIS is perhaps the best example: its original product / production model on the right, and the more recent IT model on the left.

Like TOGAF, requirements are an essential part of each zone of the model, with processes and services as the centre that holds everything together.

Hybrid frameworks: ARIS [2]

- **Strengths**

- framework is strongly process-aware
 - ARIS framework originally developed for modelling of product development and product realisation

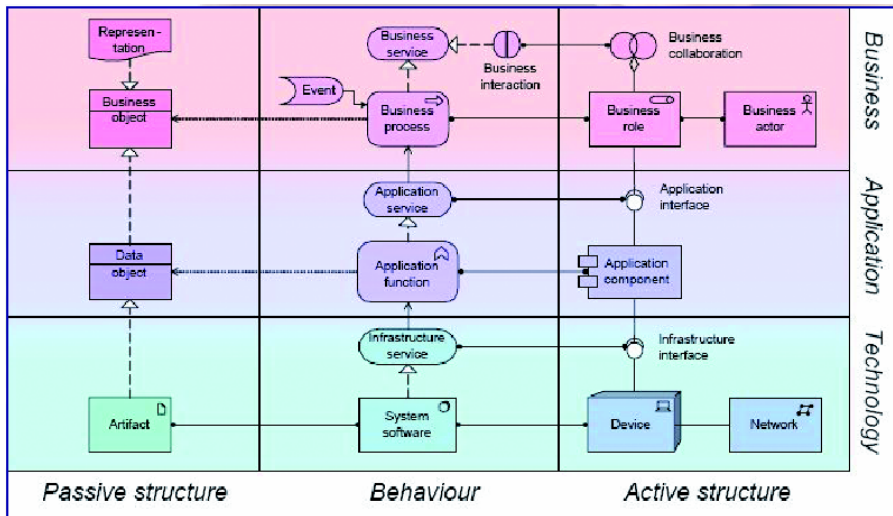
- **Concerns**

- methodology is proprietary (IDS Scheer)
- proprietary model supported only by ARIS toolset
- 'enterprise-architecture' portion is based on TOGAF, with same limitations
 - all examples are IT-based
 - no reference to people, 'things' etc except in IT context
 - no recognition of non-IT-based knowledge
- integration with process is only via ARIS toolset

But strangely, and annoyingly, there doesn't seem to be much integration between process and IT: and the IT base-model is essentially the same as TOGAF, with much the same limitations around people, 'things' and narrative knowledge.

Unlike TOGAF, though, the whole thing is 'closed', proprietary: the models and methodology are available only through the commercial ARIS toolset – a serious limitation in itself.

Hybrid frameworks: ArchiMate [1]



Source: <http://www.telin.nl/index.cfm?language=en&id=252&context=253>

Finally, to ArchiMate, another European ‘process-aware’ framework, like DyA developed in the Netherlands. ‘Passive structure’ is in essence data and its representation, a kind of expanded equivalent of TOGAF’s ‘Data architecture’; ‘active structure’ shows the equivalent of TOGAF’s business, applications and technical architectures; whilst ‘behaviour’ is the differing types of process and service at each level.

Unlike ARIS, it’s open, in the public domain, already used in several architecture toolsets such as BizzDesign. And its simple, straightforward structure does map cleanly to Zachman...

Hybrid frameworks: ArchiMate [2]

- **Strengths**

- open, non-proprietary
- framework is process-aware
- consistent, integrated modelling of IT and process

- **Concerns**

- no explicit methodology
 - is more an enterprise-architecture language than a framework
- ‘enterprise-architecture’ portion is based on TOGAF, with same limitations
 - all examples are IT-based
 - no reference to people, ‘things’ etc except in IT context
 - no recognition of non-IT-based knowledge

...the catch is that’s about all the structure there is, because ArchiMate is not so much a methodology as a modelling language, a means to portray architectural design. In that sense it’s rather like UML, the Unified Modelling Language so often used these days in software development.

And its ‘enterprise architecture’ portion is, again, derived from the TOGAF structure, leaving us with the same integration problems as before: everything’s IT-centric, with little or no allowance for physical ‘things’, for people as people, or for non-IT-based knowledge.

Supporting models and standards

- **IT governance**
 - AS 8015, CobiT, M_o_R, etc
- **Information management**
 - ITIL, BiSL, eTOM, ISPL, ASL, etc
- **Quality management**
 - TQM, ISO 9000, TickIT, ISO 27001, ISO/IEC 20000, etc
- **Quality improvement**
 - ITS-CMM, Six Sigma, eSCM-SP, IT Balanced Scorecard, etc
- **Project management**
 - PRINCE2, MSP, PMBoK, IPMA Competence Baseline, etc

But when we have an architecture-framework that does do what we need, we'll then need a plethora of tools and techniques to bring that integration down into concrete practice. Here at least we are well-served: there are standards and best-practice frameworks for pretty much everything, from IT governance to information management, quality management, quality improvement and project management.

All we're missing, then is that high-level framework for enterprise-architecture – in other words, for a real enterprise-wide architecture.

Limitations of existing frameworks

All existing architecture-frameworks have strengths, but share some common serious limitations:

- **IT-centric**
 - tend to bundle non-IT into ill-defined ‘business architecture’
- **non-dynamic**
 - tend to partition world into fixed states: ‘as-is’ vs ‘to-be’
- **poor recognition of people as people**
 - no mechanism to address uniqueness, no recognition of skills
- **limited or no recognition of physical ‘things’**
 - process-models track data, but not physical objects
- **no recognition of non-IT-based knowledge**
 - narrative as source of business meaning

As a quick summary, these are the limitations that we face with existing ‘enterprise-architecture’ frameworks.

Without exception, they’re too IT-centric. Even process-aware frameworks such as ARIS are poor at understanding people as people, or at modelling flows of physical ‘things’. And none of the frameworks seem to have any concept of the non-IT-based narrative knowledge that is the primary source of business meaning.

Even more problematic, the methodologies typically specify just two fixed states for the architecture: the present, or ‘as-is’, and the desirable future, the ‘to-be’. What this convenient partitioning fails to address is that reality is dynamic: by the time we finally get to the ‘to-be’ state, the world has already moved on. We can perhaps get away with this for a simple technical architecture, but not for a full-scale enterprise-architecture.

Limitations of languages, toolsets

Supporting modelling-languages and enterprise-architecture toolsets reflect those limitations:

- **modelling languages**
 - **BPML** (process): no ‘things’; people only as swimlanes
 - **UML** (process/data): no ‘things’; people only as actors
 - **BMM** (business purpose): organisation-centric, ‘vision’ inappropriately defined, no support for business meaning
- **architecture toolsets**
 - **Popkin/Telelogic System Architect**: good support of TOGAF, FEAF etc, but interface unusable by non-specialists
 - **ARIS** and **BizzDesign**: integrated support for process, but nothing to manage individual variance

Equally frustrating is that many of the modelling languages and toolsets reflect the same limitations.

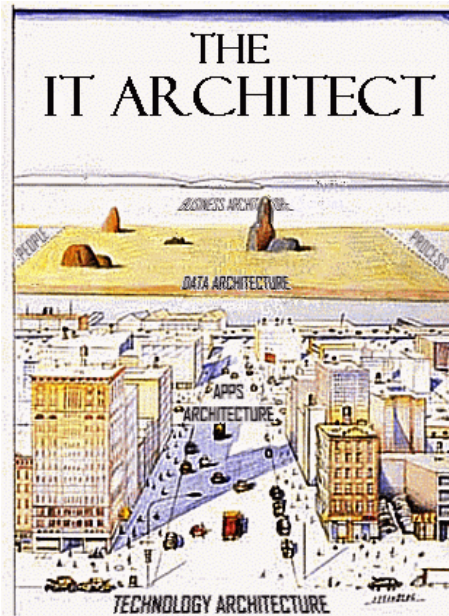
BPML, the Business Process Modelling Language supported as standard by most toolsets, has no means to model physical ‘things: which may be fine for an IT-automated process, perhaps but not for modelling the equivalent manual process that’s required for business continuity when the computer system goes down.

UML is stuck in the same IT-only world: no ‘things’, no modelling of people as people.

And the Business Motivation Model – the supposed standard for motivation modelling, up in the top-right corner of Zachman – is loaded with similar limitations, one such being its inability to model a ‘vision’ that can be shared across today’s complex multi-role, multi-partner enterprises.

The current toolsets for enterprise-architecture, such as System Architect, ARIS and BizzDesign, end up enforcing these restrictions. Their clumsy user-interfaces and frequent lack of internal rigour don’t help in this, either.

IT-centric architecture: too narrow a view?



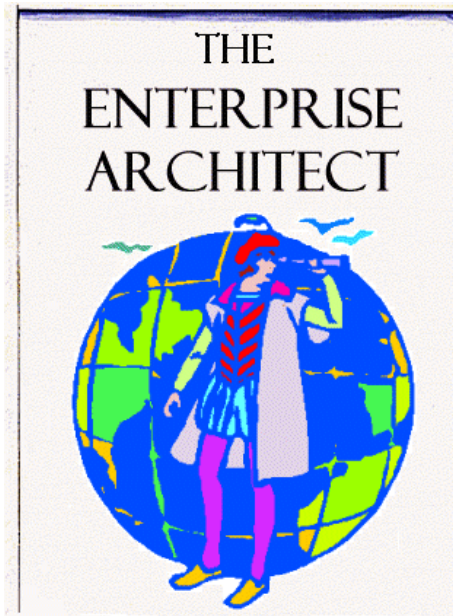
Like the old *New Yorker* cover, an IT-centric ‘enterprise-architecture’ tends to see its world as flat, with low-level technology at the centre of it all...

...and everything else of decreasing importance the further away it is from that centre...

...which is not what business wants or needs.

In effect, the current ‘enterprise-architecture’ frameworks, languages and toolsets all portray the world as a kind of ‘flatland’, centred on the low-level technical concerns of IT...

A broader view of the enterprise



By contrast,

an integrated, business-oriented enterprise architecture needs to see the 'world' as a whole, like a globe, with no real centre as such...

...and must be willing to explore and map any aspect of the broader enterprise, to create a greater understanding of that whole.

...whereas what business needs is a much more rounded view.

Seeing the world from every direction, every dimension.

Not a flatland, but a globe.

And that is what our enterprise-architecture frameworks need to support.

Core principles for ‘a broader view’

Some core principles for an extended whole-of-enterprise architecture:

1. **Holistic** - architecture of the whole as whole
2. Emphasis on overall **effectiveness**
3. **Business-driven** - ‘on purpose’
4. Support for agile response, **agile development**
5. **Governance** via responsibility-based ‘ownership’
6. Centrality of **requirements management**

Implement via a **service-oriented architecture**

So here are what I’d suggest are some core principles for that broader, rounded, more ‘global’ view of the enterprise.

Let’s look at each of these in a little more detail.

1. Architecture of the whole [1]

Four dimensions to the structure of the enterprise:

- **Business-direction dimension**
 - Business drivers/goals, strategy/tactics, performance, etc
- **People dimension**
 - skill-sets, teamwork, social networks, rostering, etc
- **Knowledge dimension**
 - information-technology, tacit knowledge, business meaning
- **Physical dimension**
 - machinery, warehousing/stock, logistics, lead-times, etc

and the **integration** of these into a coherent whole

The existing frameworks tend to bundle everything that's not IT into a general grab-bag labelled 'business architecture'. Before we do anything, we need to tease these threads apart into something more usable at an enterprise-wide scope.

At an abstract level, it's useful to describe the overall enterprise in terms of four distinct dimensions: business-direction, people, knowledge and physical 'things'. What we'd think of as 'services' and processes sit in the interactions between these dimensions. Seen this way, it's clear that IT is really only one part, one minor subset, of the overall 'knowledge' dimension – which helps to bring a sense of perspective on the limitations of an IT-centric 'enterprise architecture'.

A key point here is that this interaction is dynamic, not static. What makes it work is a kind of hidden 'fifth dimension', the systematic integration of the four dimensions into a coherent whole.

1. Architecture of the whole [2]

These four dimensions form the ‘four corners of the globe’...

...IT Architecture *and* Business Architecture, together, and more...

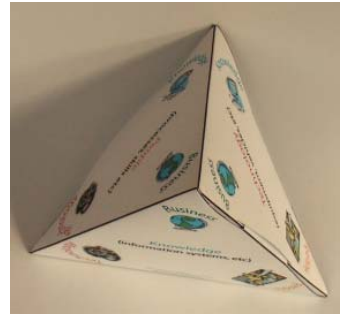


...rotating constantly
between views...

...to maintain that sense of the whole...

...seen from any dimension, any direction, as needed.

More information: <http://tetradian.com/name>



A tetrahedron provides us with a simple, yet tangible, model of these four dimensions, the ‘four corners of the globe’ from the business perspective.

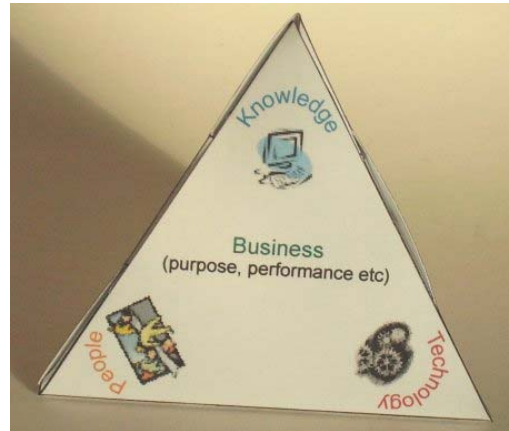
By rotating our attention constantly between these views, we gain and maintain a sense of the whole, to keep it working as a whole. This is what enterprise-architecture really needs to create for us.

(You’ll find more information about this ‘tetradian’ on the Tetradian web-site, at <http://tetradian.com/name> .)

1. Architecture of the whole [3]

Typically, each area sees up to three dimensions at one time:

- an **Operations** area sees only People, Machines, IT/Knowledge (*as on right*)
- an **IT** area sees only IT/Knowledge, Machines and Business
- an **HR** area sees only People, Business, perhaps IT/Knowledge



The business system is comprised of **all four dimensions**, always; the architecture must model this whole, as a whole.

Each area of the business has different emphases on these dimensions – and also the dimensions they tend to ignore. For example, Operations don't have time to think much about business-directions; IT are perhaps notorious for forgetting about people, whilst HR perhaps think of little else.

Yet we can't expect most business-people to keep all four dimensions in mind at once, especially under the “Do it now!” pressures of day-to-day business: it's way too much to ask. So it's our responsibility, as enterprise architects, to keep that balance for the enterprise as a whole – maintaining all four dimensions in our models, always.

2. Emphasis on effectiveness

- **Is it Efficient?**
 - maximises use of resources, minimises wastage of resources
- **Is it Reliable?**
 - predictable, consistent, self-correcting, supports 'single source of truth'
- **Is it Elegant?**
 - clarity, simplicity, consistency, self-adjusting for human factors
- **Is it Appropriate?**
 - supports and maximises support for business purpose
- **Is it Integrated?**
 - creates, supports and maximises synergy across all systems

Aim is to ensure systems fit in with everything else

More information: <http://tetradian.com/strategy#score>



29

Many business managers are all but obsessed by efficiency; yet efficiency is only one part of what we really need, which is effectiveness.

There's no point in making something more 'efficient' if it isn't reliable. And if it isn't elegant – in both the scientific and human sense – it won't work well: simplicity, clarity and consistency really do matter when maintaining distributed systems over long periods of time. There's also no point in doing something unless we know it's 'on purpose' – which means we first need to know what the purpose for that 'something' is. And we need to ensure that these not only work together, but tie in well with everything else: without integration, increased efficiencies in one place invariably lead to greater losses elsewhere in the system.

These may seem obvious: but they can be all too easy to miss unless we test for them, systematically, repeatedly, in every part of our enterprise-architecture.

(More information about this on the Tetradian website at <http://tetradian.com/strategy#score>)

3. 'On purpose' - business-driven

The anchor for a business-architecture, and for any quality-system, is business vision and business purpose

- **Clarity on business purpose**
 - importance of an emotive 'vision' to support motivation
- **Need for audit-trail of purpose for all activities**
 - suggested structure of vision / role / mission / goal
- **Purpose is dynamic**
 - systematic foresight tools to track strategic 'weak signals'
- **Requirements are dynamic**
 - centrality of requirements identification and tracking

More information: <http://tetradian.com/strategy#vrmg>



30

Existing architecture-frameworks do increasingly call for some kind of linkage to business drivers and business goals, but don't seem to give any guidelines as to how to do this. In practice, it needs to be one of the core concerns of the 'business direction' dimension in the extended architecture.

A key problem is confusion around the term 'vision'. It's not a marketing exercise: true, it needs to be emotive, yet it must also be 'larger' than the organisation itself, to create space for customers, partners and so on. A systematic 'motivation audit-trail' of vision / role / mission / goal seems to provide the best approach. (More about this on the Tetradian website at <http://tetradian.com/strategy#vrmg>)

Although the vision must be stable, business purpose also needs to be dynamic, responding to changes in the environment. To track the strategic 'weak signals' of future change, a systematic use of foresight theory and practice would seem important here – likewise a systematic approach to requirements management, of which more later.

4. Agility for dynamic change

- **Dynamic design - not state-based ‘as-is’ vs ‘to-be’**
 - there can be no certain ‘future state’, only dynamic change
- **‘Just enough, just in time’ architecture**
 - populate the overall model one piece at a time, as needed
- **Use iterative development for agility**
 - DSDM methodology: features are variable, budget is fixed
- **Prepared for surprise, not against surprise**
 - responsive / proactive model supports experimentation
- **Resilient, self-adaptive architecture**
 - architecture requires compliance but also adapts

A key flaw in many existing architecture frameworks is that they assume a final, unchanging ‘to-be’ architecture – like a finished building. But even a building changes slowly over time, and in any case enterprise-level architecture is more like the design of a ever-changing city. We need to design for change, not against it.

Classic frameworks such as FEAF and Zachman tend towards the Waterfall software model: monolithic, cumbersome, expensive and so slow they’re out of date before they start. By contrast, the DyA framework promotes the idea of ‘just enough, just in time’ architecture – the same iterative approach as Agile DSDM development, which turns conventional software methodology on its head by making the list of features variable to fit fixed limits of time and budget.

In short, we our architecture to be resilient, self-adaptive, prepared for surprise, open to experimentation with new technologies – and sufficient support all of this in a managed way, across the whole enterprise. A hard ask: but it does help to be clear that it is what we need, rather than holding everyone back at the Waterfall.

5. Responsibility-based governance

- **Responsible ‘owners’ for all key items**
 - business data-entity, business process, business rule, etc
- **Responsible by choice, not by role**
 - forced ‘responsibility’ often leads to failure
- **Encourage governance through responsibility**
 - emphasise ‘carrot’ of shared advantages of compliance, rather than ‘stick’ of reduced funding or other penalties
- **Lead change through self-responsible ‘champions’**
 - local champions are committed to making things happen, but may resent attempts at control from the centre

A key concern for enterprise-architecture is the need for governance: there’s no point in creating a design if no-one will follow it.

But the ‘big stick’ approach rarely works: and architects don’t have the time or the clout to police every development in the enterprise. What does work is a responsibility-based approach: if people ‘own’ a project, a data-set, a business-rule, they’re more likely to take care of it. A key part of the architect’s role, then, is to show people that it’s in their interest to be responsible on behalf of others as well as for their own immediate group.

Best tactic: find the local ‘champions’ of agile innovation in each area, and make sure they have the support they need. Help them to keep aligned to the architecture, and to share ideas and experiences with others, but otherwise keep out of their way – they’ll balk if they see anything that seems like control from the centre!

6. Centrality of requirements

As in TOGAF's ADM, requirements must be at the centre of architecture-development:

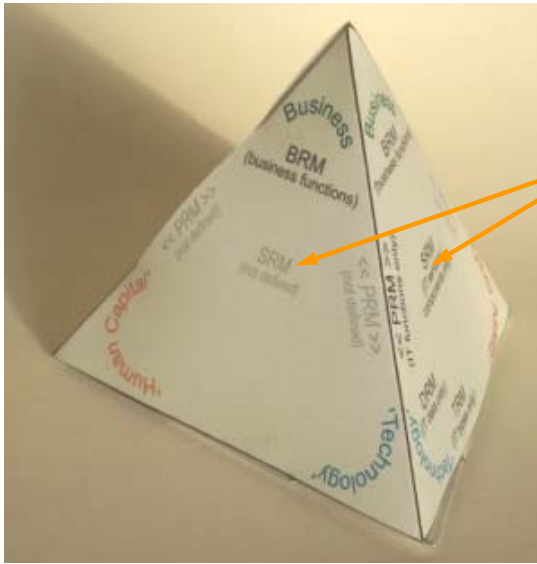
- **Requirements are dynamic**
 - versioning, release-management etc are essential
- **Requirements are defined by people**
 - each requirement needs an 'owner'
- **Requirements may conflict**
 - map all conflicts and their resolutions
- **Requirements vary in priority**
 - use MoSCoW priorities: Must have, Should have, Could have, can Wait for later release

As in the TOGAF ADM, requirements are the architecture's core. Everything from the organisation's vision to the appearance of a single item on a screen is a requirement – and all of these are part of the architecture.

The hard part is that they're also changing, all the time. (The only requirement that shouldn't change, ever, is the vision.) They're dynamic; they belong to different areas of the enterprise; they conflict. And in an Agile environment, requirements become more important, not less, because different concerns come to the fore with each iteration.

Release-management, ownership, version-control, conflict-resolution: these are all part of requirements-management, and hence for the architecture itself.

Service-oriented architecture (SOA)



As in FEAF's architecture, services are the key - the balance-point around which everything else revolves

- service as 'system'
 - service-contracts
 - SLAs (service level agreements)
 - KPIs (key performance indicators)
 - KSCs (key success criteria)

* Adapted from FEAF (Federal Enterprise Architecture Framework), <http://www.cio.gov>

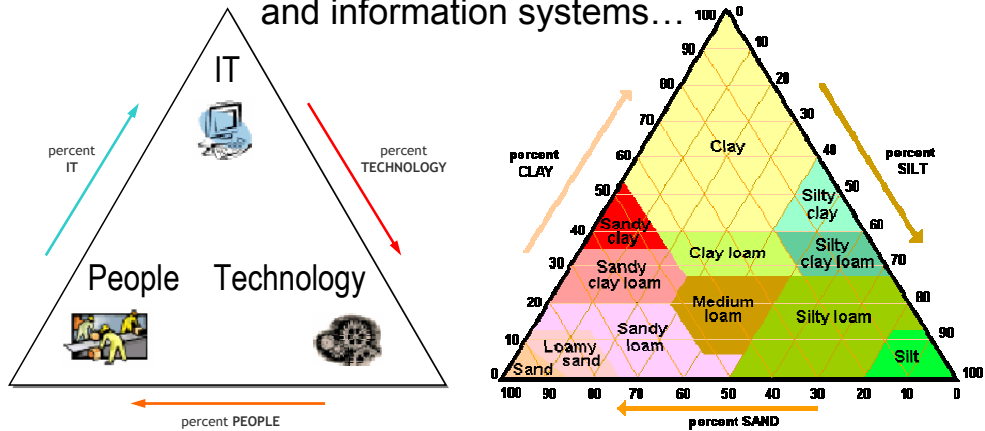
‘Service-oriented architecture’ is something of a buzz-word in IT circles at present: by describing relationships between systems in a consistent way, as ‘services’, IT complexity can be greatly reduced.

Yet the same applies elsewhere in the enterprise: everything is a ‘service’. In effect, every business-process is a self-contained ‘system’ that provides a service; and each of these services needs clear ‘service-contracts’ with its ‘providers’ and ‘consumers’, with its own explicit Service Level Agreements, Key Performance Indicators, Key Success Criteria and the rest.

In this sense, services and service-oriented architecture are the key to creating simplicity across the whole enterprise – and all of the business benefits that that would bring.

The structure of a service

A service is comprised of manual processes, physical technology and information systems...



...in any appropriate combination, much like different soil-types.

Different combinations might be used in different contexts – Hub versus Local Centre – but still deliver the same service.

More information: <http://tetradian.com/entarch#service>

It's important to understand services first in this abstract way, because it tells us the nature of each service, and what service-contracts are required with its 'providers' and 'consumers', independent of how the service is implemented.

We then can choose what mix of IT, people-processes and machine-technology to use in each implementation – in much the same way that each type of soil in a garden is made up of a different mix of clay, sand and silt.

Each different mix will give the service different SLAs, and different internal processes; but the overall service – the external service-contracts, the KPIs and KSCs – should remain the same. Among other advantages, this simplifies planning for overload and for disaster-recovery: we can change the implementation of the service - the mix of IT, people and technology – without changing the service itself.

(More on this on the Tetradian website at <http://tetradian.com/entarch#service>)

Summary

- Existing frameworks and tools are excellent for IT-centric ‘enterprise-architecture’
- Those frameworks and tools are not well-suited for use beyond that scope
- An **integrated approach** to enterprise-architecture is essential for further synergies across organisations
- The ‘**four dimensions**’ model provides a simple starter-framework for an integrated architecture
- An expanded notion of ‘**service-oriented architecture**’ is a core requirement for an integrated operations model

So let's summarise what we've seen in this brief excursion through present enterprise-architecture.

The current frameworks and tools work well with an IT-centric architecture; but they don't work well for anything else. Dumping everything that's not-IT into a blurry, indistinct grab-bag marked ‘business-architecture’ won't be acceptable for much longer.

What we need to do now is expand out that grab-bag into its proper dimensions, and be clear about their characteristics and the interactions between them, together with the dynamic nature of business requirements and business integration. The ‘four dimensions’ model gives us the simplest possible framework for a high-level model of this business ‘globe’.

And an expanded concept of abstract ‘services’, independent of their implementations, would provide the key means to apply that high-level model into purposeful, profitable, business practice.

[ends]