Architecture as system
Enterprise-architecture, service-architecture and the Viable System Model

Tom Graves : *Tetradian Consulting*
January 2007

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

Architecture as system

- **Audience**
  - enterprise architects, service architects, operations strategists, process analysts, IT-systems analysts
- **Objective**
  - increase business take-up of enterprise architecture
  - demonstrate tools to review service-architecture
- **Agenda**
  - extend enterprise-architecture beyond IT-systems
  - extend service-architecture across all service types
  - use 'systems-theory' tools to improve services
  - business benefits of extended architecture

TETRADIAN
the futures of business

Although it's still a relatively new discipline even in large organisations, enterprise-architecture continues to gain momentum as a means to manage cost and complexity, particularly in IT-systems.

One of the most significant issues in enterprise-architecture today is how to extend the current frameworks beyond IT-systems alone, and make it more usable and useful for the enterprise as a whole.

So this presentation explores two related methods to address this issue: extending the concept of service-oriented architecture to include all types of business services; and using systems-theory techniques to validate service designs.

Enterprise architecture maturity levels

"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

A systems view of architecture extends this further:

- **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

TETRADIAN
the futures of business

As Dana Bredemeyer explains in his well-known 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 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.

EA needs a broader scope than IT-systems

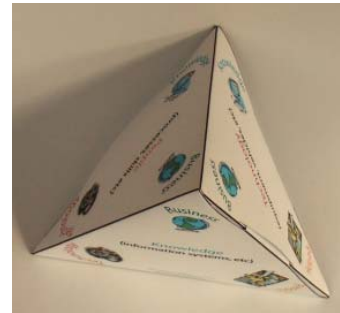We need an *integrated* view of business systems...

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

...rotating constantly between views...

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

...including the *business* dimension, symbolised by

TETRADIAN
the futures of business

To do this, we need to extend enterprise-architecture far beyond IT alone.

We need it to extend to all of the organisation's 'four corners of the globe': business knowledge (of which IT is only one part); the people and processes of the business; machines and other physical technology; and business management and strategy.

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

Service-oriented architecture (SOA)
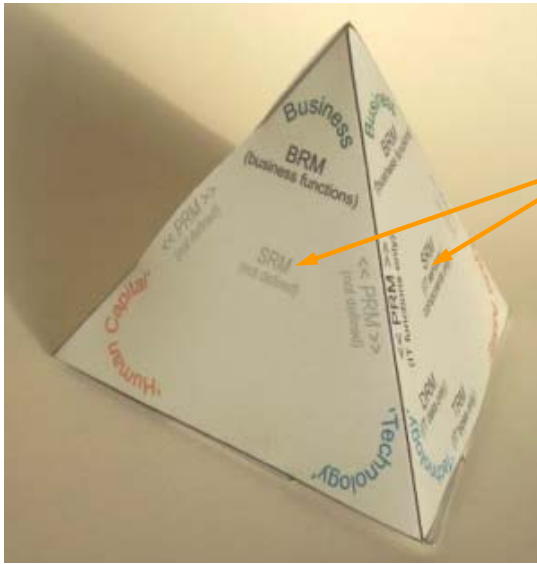
As with IT architecture, services are the key – the balance-point around which everything else revolves

- 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

TETRADIAN
the futures of business

'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: <u>everything</u> 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 SLAs, KPIs, KSCs 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 also a 'business system': it comprises "manual process / activities, physical equipment and information systems"…
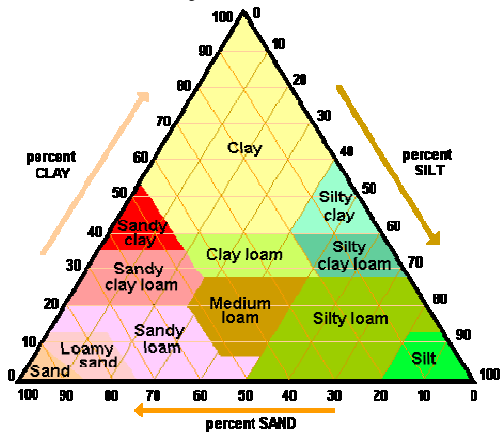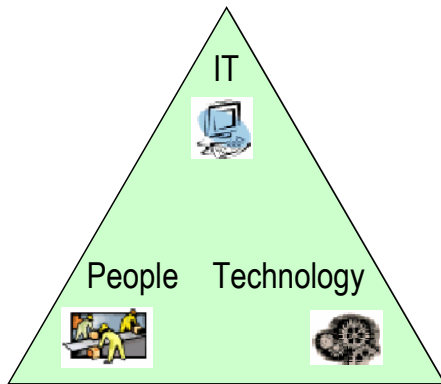
…in any appropriate combination, much like different soil-types.

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

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.

Services and infrastructure

We need to distinguish between key categories of services:

- **task services** create end-to-end processes *along* value-chains
  - *banking examples*: receive application, credit check, disburse funds
  - *logistics examples*: lodgement, receive in dock, transport, sorting
- **infrastructure services** provide support *across* other services
  - *business examples*: performance monitoring, strategy, scheduling
  - *IT examples*: networks, applications, SOE, help-desk, phone mgmt
  - *people examples*: time and attendance, recruiting, rostering
  - *asset examples*: building mgmt, power, equipment maintenance

Each type of service may deliver via any combination of IT/knowledge, machine-technology, people or business assets, components and processes

TETRADIAN
the futures of business

We can split services into two main categories, which I describe as task services and infrastructure services.

Task-services link together into 'value-chains' – the end-to-end processes which determine what the organisation does. As a result, these are usually regarded as 'core tasks' – visible, and highly valued. Most of an organisation's process-engineering effort will go into streamlining these services.

Infrastructure services provide support for these 'core tasks', linking across the value-chains. Even though they don't provide value directly, as task-services do, the organisation cannot function without them.

As before, any of these services could, in principle, be delivered via any mix of IT, people or technology.

Services and infrastructure
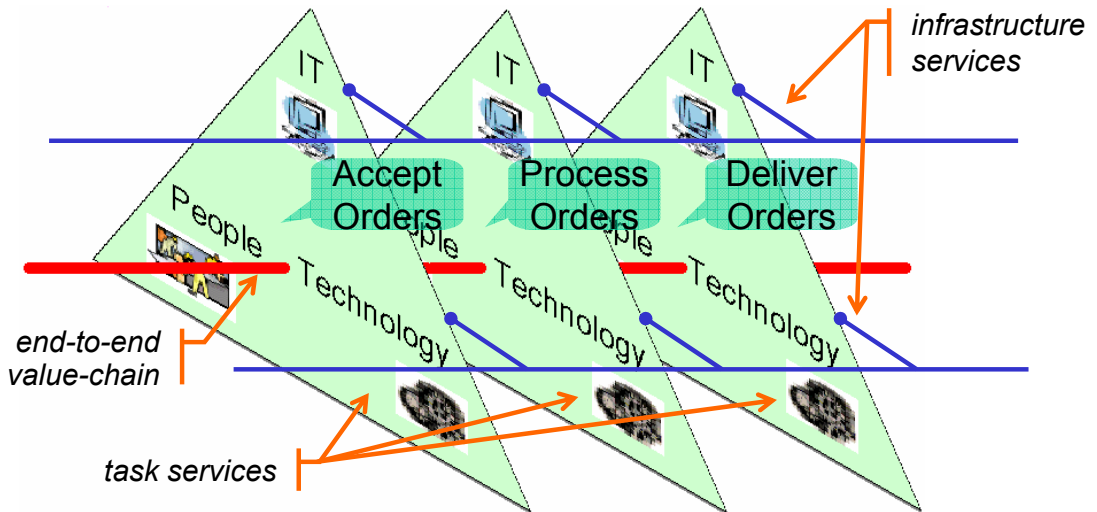
Task services are linked *along* value-chains

Infrastructure services link *across* value-chains – also in some cases across other infrastructure services

To describe this more visually, imagine a set of services – each implemented via their own mix of IT, people-processes and technology - linked along a value-chain.

In this example, the value-chain might be 'Accept Orders', 'Process Orders', and 'Deliver Orders' – as in a retail or logistics context.

These are the task services of the value-chain.

The infrastructure services link across the value-chains, providing support to the task-services. Some services – strategy, for example, or recruitment, or networks – provide infrastructure-services to task-services and to other infrastructure-services alike.

Services and the Cost Model

- **Task-services are relatively easy to cost**
  - often correspond with Activities on the Functional Model
- **Infrastructure-services are often (much) harder to cost**
  - few infrastructure-services may be visible on the Functional Model
  - costs tend to be absorbed in and concealed by task-services
- **Support-services for infrastructure may be even less visible**
  - is especially true for abstract 'services' such as corrective-action, knowledge-sharing, vision/values maintenance, 'connector' roles
- **A key objective for a service-oriented architecture is to 'surface' *all* the hidden infrastructure – and its costs**
  - direct cost of the service, if fully supported and integrated
  - opportunity-cost of an absent or under-supported service

TETRADIAN
the futures of business

This distinction between task-services and infrastructure-services becomes important in costing and cost-management.

Task-services visibly provide value: their costs and benefits are relatively easy to identify.

But because infrastructure-services contribute only indirectly to value, their costs and benefits can be much harder to identify. As a result, they might be dismissed as invisible, irrelevant, 'a cost to the business'. The danger is that they then become too-easy targets for budget-cuts and over-zealous 'process re-engineering' - yet the task-services cannot operate without them.

This is especially true for the more abstract services of 'infrastructure to infrastructure', such as quality-management. Or management itself, for that matter.

A key objective of a service-oriented architecture, and one of its key advantages, is to 'surface' the functions and costs of all the hidden infrastructure – making it easier to measure and manage.

The systems trade-off within services

Once its purpose is clear, a service may use *any* combination of people-processes, machine-processes and knowledge-processes...

...the key concern is *effectiveness*.

IT

People   Technology
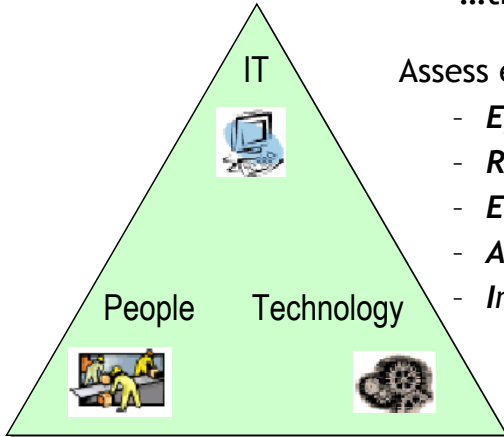
Assess effectiveness with the **EREAI** checklist:
- *Efficient* (conceptual / knowledge domain)
- *Reliable* (practical / process domain)
- *Elegant* (human/ergonomic domain)
- *Appropriate* (purpose/business domain)
- *Integrated* (systems domain)

Use systems-theory frameworks to assess and improve effectiveness across the whole.

TETRADIAN
the futures of business

Once we grasp this – that everything is a service, with differing details of implementation and use – we can then focus on the key concern of effectiveness.

The point here is that efficiency is only one component of effectiveness. What we might call 'elegance', for example, is essential to issues such as simplicity and self-adapting ergonomics – making it easier for people to understand and move between different processes and services. And integration is key to ensuring that gains in one area do not cause greater inefficiencies elsewhere.

Services cluster together into systems and systems-of-systems. To understand these, we must turn to the formal discipline of systems-theory.

Business system as 'viable system'

To broaden the scope, think *organisation* as *organism*

In systems-theory, each 'viable system' / service must be able:

- to *do* its tasks (a 'doing system')
- to *sense* (and report on) its internal and external environment
- to *remember* (a repository of knowledge about its past)
- to *coordinate* its activities with other systems
- to *plan* its activities (strategy and tactics, often with others)
- to *adapt* to and, where possible, improve its own environment

In principle, it should also have a sense of its own *purpose*.

This is *recursive*, a multi-faceted hierarchy: each layer is similar to, yet differs from, the layers 'above' and 'below'.

TETRADIAN
the futures of business

Although it can seem abstract at first, systems-theory is about understanding the whole <u>as</u> whole. One of the best ways to do this is to think of an organisation not as a collection of discrete parts, but as a kind of living organism.

All of the systems and services in a living organism mesh together in a coordinated way: muscles provide moving-services, blood-vessels provide energy-services, nerves provide coordination-services, and so on.

Each service has some local specialisation, in its task, but they each also share many of the same characteristics – sub-systems or sub-services, if you like - as listed here.

Frequently we'll find these sub-systems arranged in hierarchies – exactly as in any large organisation.

Services and the Viable System Model

In Stafford Beer's 'Viable System Model'*, each system / service (unit at a given layer) contains a set of specialised sub-systems

- **5** – *policy / purpose* (green)

- **4** – *'outside / future'* [+ strategy] (yellow)

- **3** – *'inside / now'* [staff management] (red)

- **3*** – *sporadic audit / review* (pale blue)

- **2** – sub-process *coordination* (mid blue)

- **1** – process *operations* (lilac)

These interact with each other to act on and with the external world (the amoebic 'blob' to the left of the diagram).
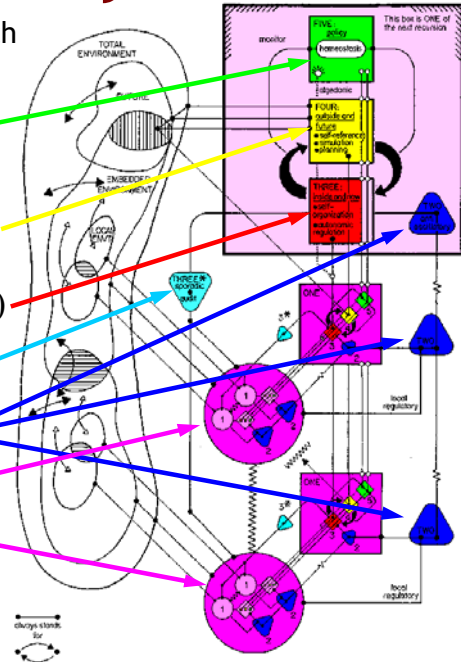
* *Models for Change: The Viable System Model*, http://www.staff.mcs.uts.edu.au/~jim/bpt/vsm.html

Staying at the abstract level for a moment, Stafford Beer's 'Viable System Model' is a classic system-theory framework, proven for more than forty years in real organisations – up to the coordination of an entire country, in one case.

The components of the 'viable system' are shown in the coloured triangles, circles and squares on the right of the diagram. The amoebic blob to the left represents the environment in which the system operates.

This model is especially useful for service-oriented architecture because it places less emphasis on the 'doing' part of the system – the lilac circle on the diagram, Beer's 'system-1' – and more on management and coordination – in other words, our hidden infrastructure-services.

VSM recursion (hierarchy of services)

The model is *recursive*: each layer contains the next, to whatever depth required.

Each cluster of 'management' services (system-5, -4, -3, also -2) may support several 'operations' clusters (system-1).

Each system-1 cluster may also contain 'management' services for the next level below in the service-hierarchy.

System-2 (coordination) and especially system-3* (sporadic audit) are partly outside – i.e. in part they are always 'infrastructure services'.

Note: VSM shows mechanisms for service-choreography (e.g. 'system 2' and 'system 4'), but not choreography itself
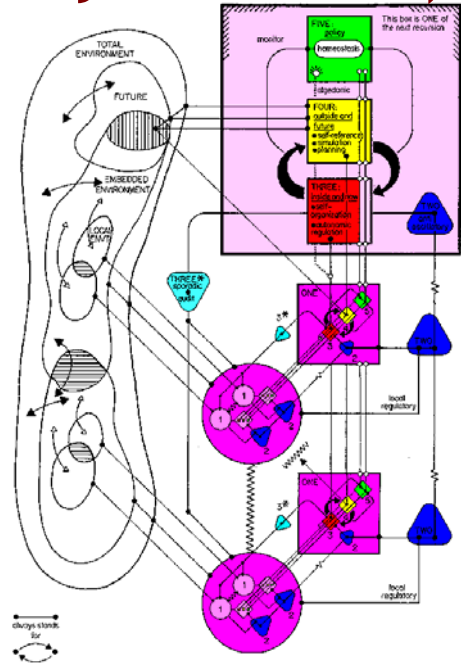
A bit more here about how the Viable System Model represents that hierarchy.

The larger pale-lilac square containing the three smaller squares ('management' systems 5, 4 and 3) and the triangle ('coordination' system-2) repeats at the next level, as the smaller lilac 'system-1' squares linked to the 'operations' part of each system-1. These smaller squares, you'll notice, each contain their own systems 5, 4 and 3, for the next level down, and so on.

Note that coordination (system-2) and, especially, 'sporadic audit' (system-3*) are necessarily outside of this hierarchy: they link directly to everything, without mediation by anything else.

VSM interactions for self-adaptation

Interactions *between* these sub-systems support improved processes and/or self-adaptation to a changing environment

- **X** – *exception-management* for short-term ('1' ↔ '3', '1' ↔ '4')

- **C** – *corrective action* (review of '3*' / 'X' ↔ '3' / '4', also driver for 'P')

- **M** – *issue-tracking / issue-management* (usually triggered by 'X', '2' and/or '3')

- **P** – *process-improvement* (interaction up and down between any '1'... ↔ ... '5')
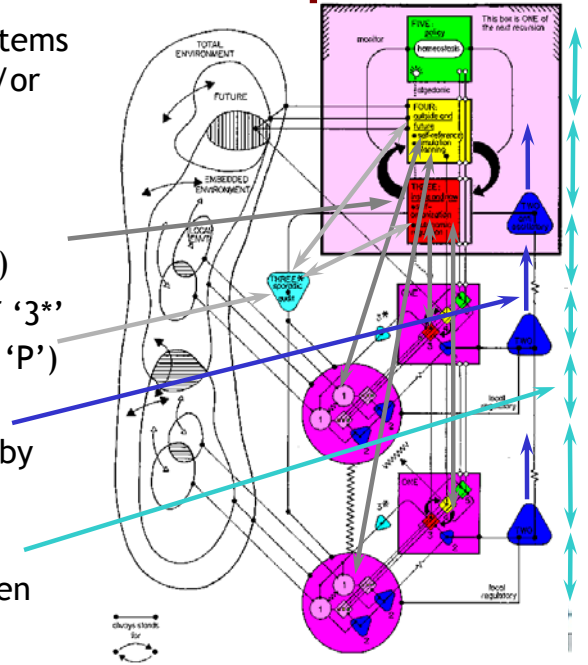
The Viable System Model also helps us understand how quality-system processes work, through interactions <u>between</u> the various 'systems' of the model.

The 'systems' shown here aren't part of the original model: in fact, they arose from our work on a quality-management project in a large logistics organisation. But they do help to illustrate the role these subtle yet essential infrastructure-services play within large organisations.

Use VSM for gap-analysis on services

VSM 'systems' provide a useful checklist to evaluate services

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

TETRADIAN
the futures of business

Coming down from the abstract level, the Viable System Model descriptions of 'systems' are immediately useful in service-oriented architecture, as a checklist to verify completeness in service design.

If any of the lettered 'systems' (X, C, M, P) are absent, the service cannot run well.

If any of the numbered 'systems' (from 5 down to 1) are absent, the service cannot operate at all – not for long, at any rate, and not as part of a 'living organisation'.

It really <u>is</u> as simple as that.

Integration: quality-systems, costing

We can use Viable System Model 'systems' as filters to review a Functional Model or Business Systems Model

- – *Functional Model*: hierarchy of services/activities by end-to-end function
- – *Business Systems Model*: clustering of related services/activities that share the same data and functions across business units

Gaps would point to unrecorded functions / services, untraceable costs, and/or lost opportunities for improvement

- – map services/activities within Function/System to VSM 'systems'
- – identify Functions/Systems without coverage for respective VSM 'system'

Next slide shows a real review in a logistics company, with a Business Systems model of 20 business-systems

- – shows count of Business Systems per VSM 'system'
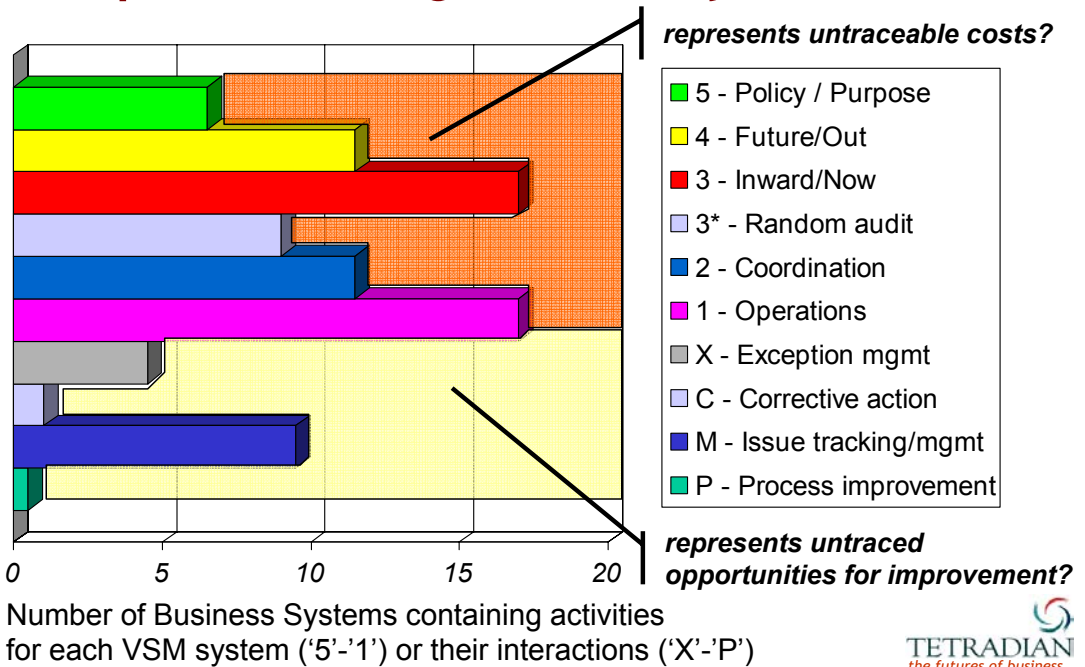
TETRADIAN
the futures of business

Armed with that knowledge, we can also use the same principles to review models right at the top of the architecture hierarchy, such as a Functional Model or Business Systems Model.

We know from the previous slides that every Viable System Model 'system' must be present in every system or service. Hence each must also be represented in some form at the appropriate level of hierarchy in a Functional Model or Business Systems Model. If it isn't, the costs for the services are untraceable – usually embedded incorrectly in the costs assigned to other services.

In addition, if the lettered 'systems' aren't represented, there's no apparent way to improve the service's performance.

These are serious problems for a service-oriented architecture.

Blueprint coverage of VSM systems

represents untraceable costs?

Legend:
- 5 - Policy / Purpose
- 4 - Future/Out
- 3 - Inward/Now
- 3* - Random audit
- 2 - Coordination
- 1 - Operations
- X - Exception mgmt
- C - Corrective action
- M - Issue tracking/mgmt
- P - Process improvement

represents untraced opportunities for improvement?

Number of Business Systems containing activities for each VSM system ('5'-'1') or their interactions ('X'-'P')

TETRADIAN
the futures of business

17

This is an example of a real review of this kind, in a large logistics organisation.

Their Business Systems Model was comprised of twenty high-level 'business systems', or clusters of related services. From a cross-map between the detailed layers of the model and the Viable System Model 'systems', none of the business-systems described the complete set of services required for a 'viable system'. Overall, barely half of the possible mappings were present in the model. In other words, barely half of the organisation's costs could be traced and assigned to the appropriate service.

Even worse, only a fifth of the possible mappings for quality-system services were described at all on the Business Systems Model – a serious issue reflected in the organisation's real-life problems of day-to-day quality-control.

Since that review, accurate cost-tracing and a major emphasis on corrective-action processes have become core priorities for that organisation.

Summary

- **Architecture extends from IT to whole of enterprise**
  - 'four corners': people, process, knowledge, business
  - improve business engagement in enterprise architecture
- **Services are the key to enterprise architecture**
  - all functions, tasks, activities can be understood as services
  - consistent service-modelling improves agility, resilience
  - higher-level service-architecture improves risk-management
- **Systems-theory frameworks clarify service-models**
  - Viable System Model as consistent description of services
  - provides checklists / gap-analysis for service completeness
  - assists in cost-tracking, identification of opportunity-costs
- **Reduce complexity, cost of business systems** TETRADIAN
  the futures of business

So, to summarise, we need to understand enterprise architecture as applying to <u>all</u> aspects of the enterprise – not just the IT-systems subset. A key benefit from this is that it also helps to bridge the all-too-common divide between business and IT.

Service-oriented architecture, and services in general, are the key to this broader role for enterprise architecture. A consistent approach to service models, in part independent from their tangible implementation, also brings great benefits to the organisation.

And abstract tools such as systems-theory can be of direct benefit in providing simplicity and consistency in service design, and in validation of service completeness.

This brings us back, in turn, to the real purpose of enterprise architecture: to reduce the cost and complexity of business systems, and to provide the enterprise with the agility and adaptability that it needs in the present day.