On Systems Architects and Systems Architecting: some thoughts on explaining and improving the art and science of systems architecting

Hillary G Sillitto
Thales UK Land & Joint Systems
1 Linthouse Road,
Glasgow G51 4BZ,
Scotland, UK
hillary.sillitto@incose.org

Abstract. ISO's process to adopt the IEEE 1471 standard on architecture descriptions has revealed of the order of 130 standards concerning or relating to architectures and architecting. Within the "enterprise architecture" theme alone, different people use the term to refer to mean quite different things: the "architecture of the enterprise as a system"; the enterprise context for the enterprise IT system; or the architecture of the enterprise IT system itself. A focus on tools and methods has led to confusion between the creative activity of "architecting", by which I mean making or exposing the key strategic decisions about the purpose, organisation, behaviour and critical design features of the system, and the analytical and descriptive activity of architecture modelling, which supports and captures the results of architecting. The INCOSE UK Architecture Working Group established a "belief systems" methodology to explore and seek to reconcile the many conflicting views on architecting This paper expands on the views presented by the author at the IS08 Architecture panel session in an effort to identify and communicate a better understanding of the fundamental skills, principles, philosophy and approach underpinning effective systems architecting. It seeks to improve their integration by focusing on "purpose, context and process" of architecting with the perspective that "hard systems exist inside soft systems", and to show that "lean pull" allows architecting to focus on the intended use of its products rather than adherence to process standard or frameworks.

Introduction

The field of systems architecting was strongly shaped by Rechtin's classic text "Systems Architecting – creating and building complex systems" (Rechtin, 1991) - whose title succinctly defines the value proposition of systems architecting. It is clear from observation and personal practice that there are quite distinct systems architect roles in the customer problem space, in framing and shaping the system solution, and in the product domain. Many of the architect's skills map directly into non-engineering domains such as organisational design.

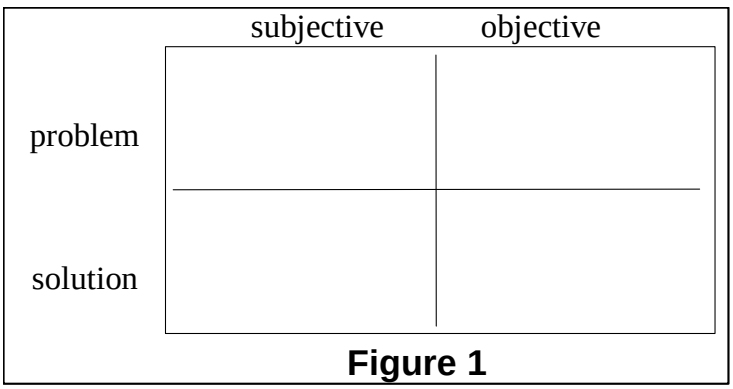

Figure 1

The central issue for systems architecting is coping with complexity. It is useful to think of four kinds of complexity, in a Boston Square 2x2 matrix (Figure 1). On one axis, there is complexity to do with the problem, and with the solution. On the other axis, there is subjective complexity – which means that people don't understand it and can't get their heads round it – and objective complexity –

which means that the problem situation or the solution has an intrinsic and measurable degree of complexity. So we use architectural methods both to give people usable mental models of complex situations and systems, and to measure and get control over their inherent complexity. These are different purposes, and they need different approaches and styles of presentation – which however need to be tied together by a common model, otherwise we are just doing Powerpoint engineering. Architecture helps us both to understand the problem and deliver the solution.

Most people would agree that we architect for a purpose, not just to produce expensive wallpaper. But it is difficult, the terminology is overloaded, different people use the same words to mean different things, styles of presentation are not always matched to the stakeholders' understanding, and our customers are confused! This confusion has led to a proliferation of snake oil in the market place. Tools, recipes and frameworks dominate the debate rather than ideas. This leads inevitably to further confusion, the antithesis of the purpose of architecture.

Architecture is not an end in its own right, merely a means to enable the creation of successful systems. We need to inject "lean thinking" and "customer pull" to make sure architects and modellers provide fit for purpose outputs to decision makers and system developers, users and operators. This implies a need for a consistent and widely understood architectural "**language** and **grammar**", such as UML, SysML, and the various widely used achitectural frameworks[1]. We need to make things clear and "obvious" to stakeholders, and at the same time keep track of the real complexity that has to be managed.

This is much easier if you have a good architectural **design**. By this I mean a clean structure that makes sense at high level and provides a framework for a robust and resilient solution that is agile and adaptable. This is more difficult because we never start from a clean sheet. The traditional project lifecycle model for systems engineering assumes that the system lifecycle has a beginning, a middle, and an end. But in systems of systems, and enterprise-level systems engineering, each project is simply a controlled change to a continuing – and hopefully improving – enterprise capability, and to the system baseline that delivers it (Kemp et al, 2008). This is a fundamental shift in perspective, and requires new ways of thinking, and new approaches to management – for both of which systems architecture techniques are key enablers.

The paper is informed by the author's experience and observations in a number of contexts and domains and seeks to identify common factors that illustrate the essential skills of systems architecting, and some traps to avoid. We conclude by presenting some "patterns of architecting" which advance the argument that architecting is essentially similar across domains.

A fundamental premise is that the various architecture frameworks (MODAF, DODAF, Popkin etc) are not the essence of architecting, merely tools language and syntax for the architect to use when appropriate. A focus on specific mandated "views" has been necessary to embed the "language" of architecting but has had the unfortunate and widely remarked side-effect of creating a focus on the specific work-products rather than the underlying purpose.

The paper tries to unwrap these issues by considering the purpose, context and process of systems architecting, applying the insight that "hard systems exist inside soft systems" (Blockley and Godfrey, 2000) leading to the idea that architecting is an intermediate or interfacing layer

[1] These, with their roots in software and information systems, address functional, behavioural and information exchange aspects of architecture well, other aspects including form and fit less so.

between the soft domain of systems thinking and the hard domain of systems engineering management and design.

Scope and definitions

This paper is concerned with all aspects of systems architecture. Any linguistic bias is unintentional and should not be interpreted as limiting the discussion to a particular domain. Terms are generally used in the spirit of their natural language definitions. In particular:

- **Functionality** – what a system does

- **Behaviour** - The way in which someone or something functions or acts or reacts to a stimulus, or responds to situations. For example [mgt] the way a person reacts in a particular role in a process or [stru] the way a beam bends in a structure or (systems) stimulus-response characteristics including outputs, state transitions and time constants.

- **Capability** - the ability to do something. Systems *have capability* in given states, modes and operational environments. For example: **System capability** – function and performance of a system under defined operational conditions; **Operational capability** – ability to provide a useful service that provides value to stakeholders; **Military capability** – a type of operational capability, the ability to create a desired military effect in a given operational context.

Purposes for architecting

Architecting has a number of purposes and practices depending on context, many of which were explored at the Special Architectures Workshop (INCOSE International Workshop, Jan 2008). These include:

- Explore the problem space to establish an understanding of the problem
- Establish a framework within which stakeholder consensus can be developed
- Define the context and relationships within which analyses and trade-off decisions are to be made
- Provide analysis for decision support
- Establish the strategic organisational framework for a project or product line
- Control design by setting out and communicating the strategic “system wide design decisions” that establish coherence across a project
- Guide testing, acceptance and transition to operations by clarifying the relationship between purpose, usage, design, performance and behaviour.
- Provide mental models that allow operators and users to understand how to configure and use the system.

An understanding of these purposes allows architecting effort to be focused to maximise value and avoid the “expensive wallpaper” syndrome.

The context for architecting

Since the essence of architecting is managing complexity it follows that architects work in complex situations and complex organisations. These may be projects, businesses, government agencies, or cross-organisation “enterprises” created to pursue and execute large projects and/or to create and operate large systems. So architects need to communicate with many stakeholders. Different stakeholders will require a different style of engagement.

A discursion on Architectural "language": Abstraction and effective communication

Good architects are capable of dealing with a wide range of levels of abstraction and moving freely between them. They need to be aware that the people with whom they are trying to communicate may not share this facility. They also need to be aware that different people think in different modalities. Neuro-Linguistic programming identifies a number of primary modalities of thought including verbal, visual and kinaesthetic. As a caricature, we can think of engineers as being visual, civil servants as being verbal, and military people as being kinaesthetic. Architects working in a defence acquisition environment have the challenge of finding ways to present their work that work with all three of these populations and convey the same message to all three. This was brought home strongly to the author when he was told "if you want [a senior military individual] to understand your diagram, make sure it has lots of arrows on it". When relaying this to a colleague he said "ah, you mean like the opening credits in Dad's Army (a popular British TV comedy programme set in a military context in World War 2 – the opening credits use moving arrows to illustrate the evolving military situation in Europe in 1939-40).

It seems that people who look at an architectural work product form an overall impression of the message almost subconsciously in their preferred modality, and subsequently absorb messages in the other modalities that are consistent with their initial impressions. This means that if a diagram uses inconsistent "grammar rules" in the visual and verbal modalities, or in the "flow" implied by the diagram, it will convey different and inconsistent messages to people with different modalities.

The architect needs to ensure that his chosen methods of communication convey the same message to people with different NLP modality preferences, and to people in different cultures. This means he needs to review the products in detail with the intended users to make sure the work products are fit for purpose and convey the intended messages consistently and accurately.

The challenge and process of architecting, and the nature of complexity

I believe that the architect's modus operandi is to divine the fundamental underlying structures, patterns and rule and within a complex problem space, to establish clearly to the purpose and context of the proposed intervention (often the creation and deployment of the system of interest); and to establish the fundamental structures, patterns and rules that will establish coherence in the solution – the project, the process and the product. These roles reflect the two fundamental architecture belief systems identified by the UK Architecture Working group (Wilkinson and Bryant): "reverse architecting" to understand what exists, and "forward architecting" to define the desired future state. The essence of this "Janus-like" role is mastering and then managing complexity.

Figure 1 introduced the concept of problem and solution space, and subjective versus objective complexity. The literature on complexity shows little meeting of minds on the subject. In "an introduction to systems science", Warfield focuses on the subjective aspects and introduces the concept of "spreadthink". This is essentially a measure of the range of "belief systems" or "world views". This thinking is grounded in the concepts of soft systems and based on the assumption that complexity is a social phenomenon. Sheard and others have presented (e.g. Sheard, 2005) concepts of complexity rooted in "hard systems science" based on the mathematical concepts of chaos and complexity theory. This kind of complexity is measurable and quantifiable. This

concept of “measurable complexity” is real, and is demonstrated for example in the theory and measurement of partial coherence in electro-magnetic wave propagation – the classic reference for this in the field of optics is (Born and Wolf).

Figure 2 shows that in the belief system represented by Figure 1, the purpose of architectural description is to reduce subjective complexity and remove the perception of “unknown unknowns”; while the purpose of architectural decisions is to manage objective complexity and maximise system effectiveness, integrity and flexibility.

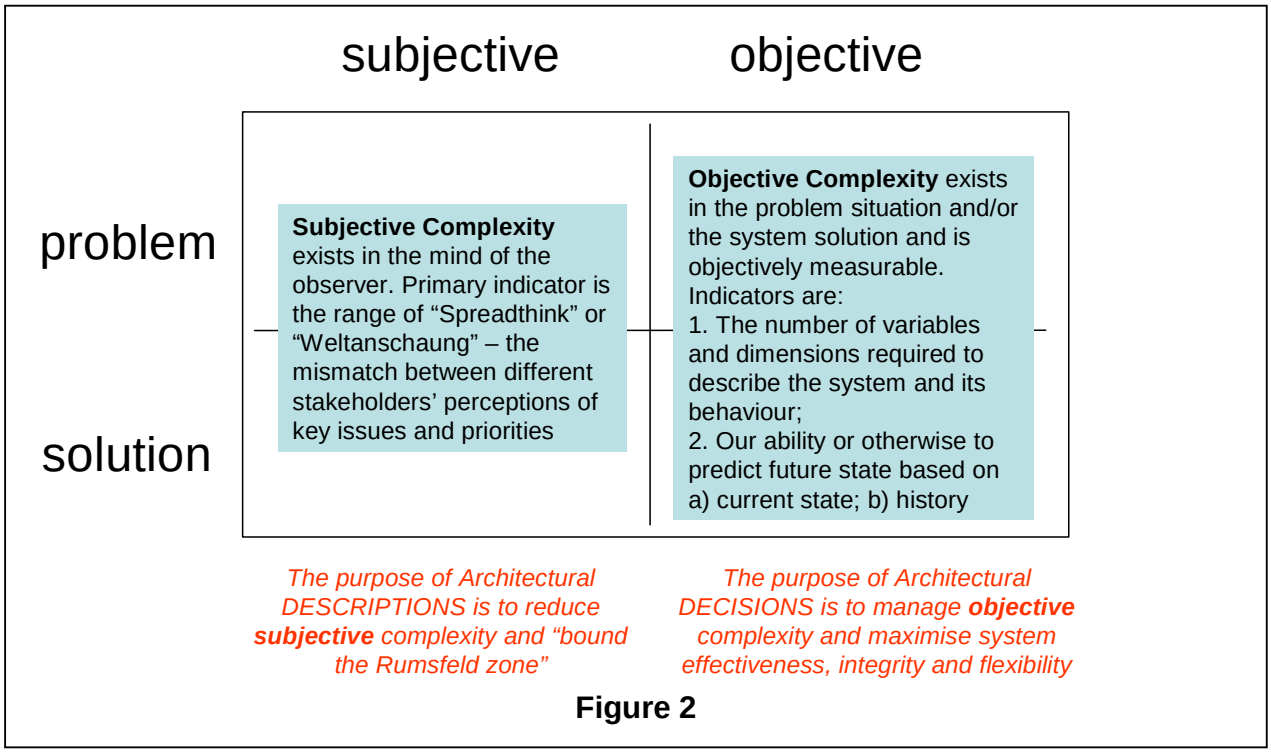

Figure 2

This guides us to the conclusion that the architect needs to work in a number of areas and maintain coherence between them, for a number of different but related purposes.

In the subjective domain his task is to produce and secure agreement on shared mental models of purpose, context and the major characteristics and structure of the solution – to establish the “shared vision” of the system, and a clear understanding of the uncertainties and un-knowable aspects of the problem.

In the “objective-problem” quadrant the architect needs to establish a rigorous definition of the intended use and constraints – including the business process the system supports, and the physics of the domain, as well as the behaviour of interfacing and inter-operating systems. In the “objective – solution” quadrant we need to describe and model the characteristics and dynamic behaviour of the solution as a “system black box” and show how these characteristics and behaviours are created by the characteristics and dynamic behaviour of the components of the system and their interactions with each other and the environment. The objective models of problem and solution space allow the architect to analyse and optimise system level emergent properties in the intended environment within the understood constraints: and then to “control the design” (including the organisation and the process) and define the approach to integration and acceptance of the solution.

To the simple and powerful mantra “Architecture controls design, design controls implementation” it seems necessary to add two further roles. The first is to provide the framework for integration and acceptance – a clear message from (Rechtin, 1993). The second is more difficult and subtler.

In the modern world it is not sufficient to “predict and provide”. Complex systems need to have “goal-seeking behaviour” if they are to achieve their purpose and provide enduring value in a changing world. The system needs to be architected along with the organisation that will operate it, the processes it will use, the measures it will use to manage the processes, and the incentivisation of the people who will operate and derive value from it.

Architecture frameworks and products: problem or solution

This section reviews experience to date with architectural frameworks, including the DoD and MOD Architecture Frameworks, MODAF, DODAF, and suggests that we have some way to go in peoples and team maturity, understanding and implementation of process, and tool support before the envisaged benefits of current architectural frameworks can be fully realised.

Architecture frameworks are used to organise the output of architecting. They provide a “language and grammar” for presenting the results of architecting in a way that they can be readily understood and compared. Their role is illustrated by the following diagram, which was produced during the Special Architecture Workshop at INCOSE’s International Workshop 2008.

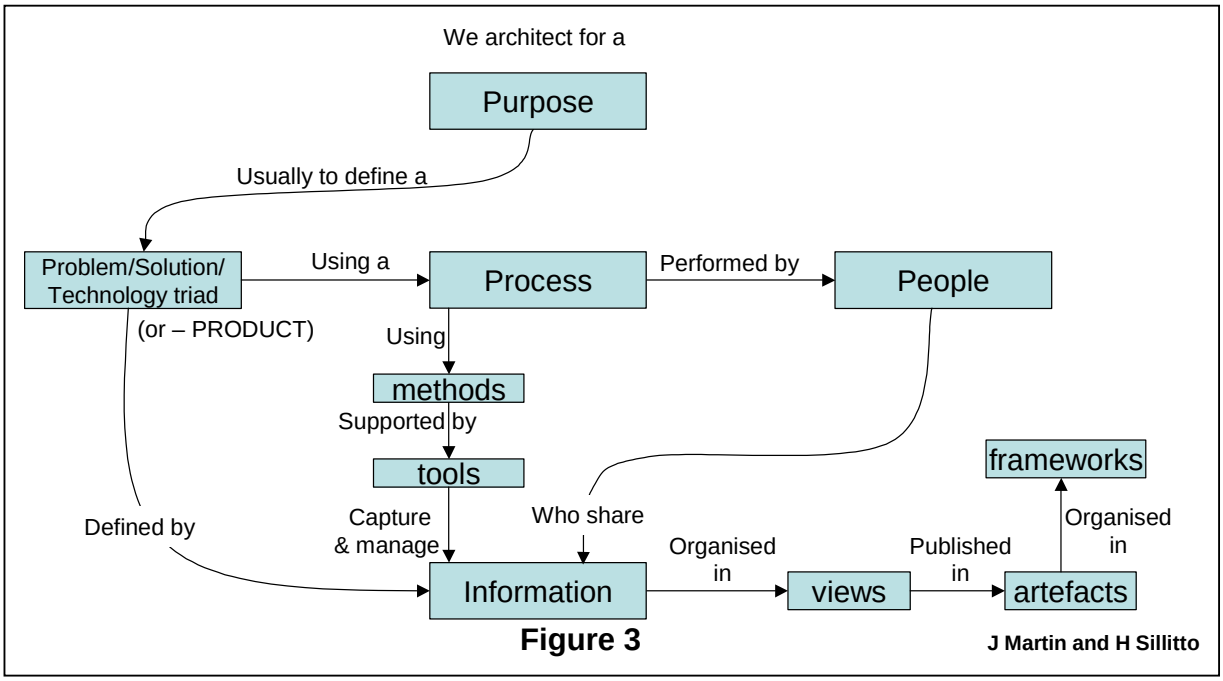

Figure 3

The benefit and purpose of Architectural Frameworks can be summarised as “reducing perceived complexity through separation of concerns”. DoDAF and MODAF were introduced by government agencies responsible for system of systems design, management and assurance, seeking to improve coherence across their area of system-of-systems responsibility and facilitate the management of interoperability and delivery of system-of-system capabilities. While there is

clear evidence that the approach can work and provide enormous benefits, it takes a long time to embed the “language” and during this bedding-in period there is a tendency for the benefits to be diluted by an assurance-driven “tick in the box” mentality. Some projects, teams and individuals “get it” and others do not. The value a project gets from doing work on architectural frameworks depends on its maturity of approach.

At an initial level of maturity, use of these frameworks focuses on individual views, which are used essentially to get stakeholder consensus; there is no guarantee that the views are mutually consistent, correct or viable. At a higher level of maturity, the views are derived from an integrated model which guarantees to the extent that its scope permits that the different views are mutually consistent with each other and with the underlying belief system embedded in the architecture model. This is illustrated in Figure 4, originally published in (MOD, 2007).

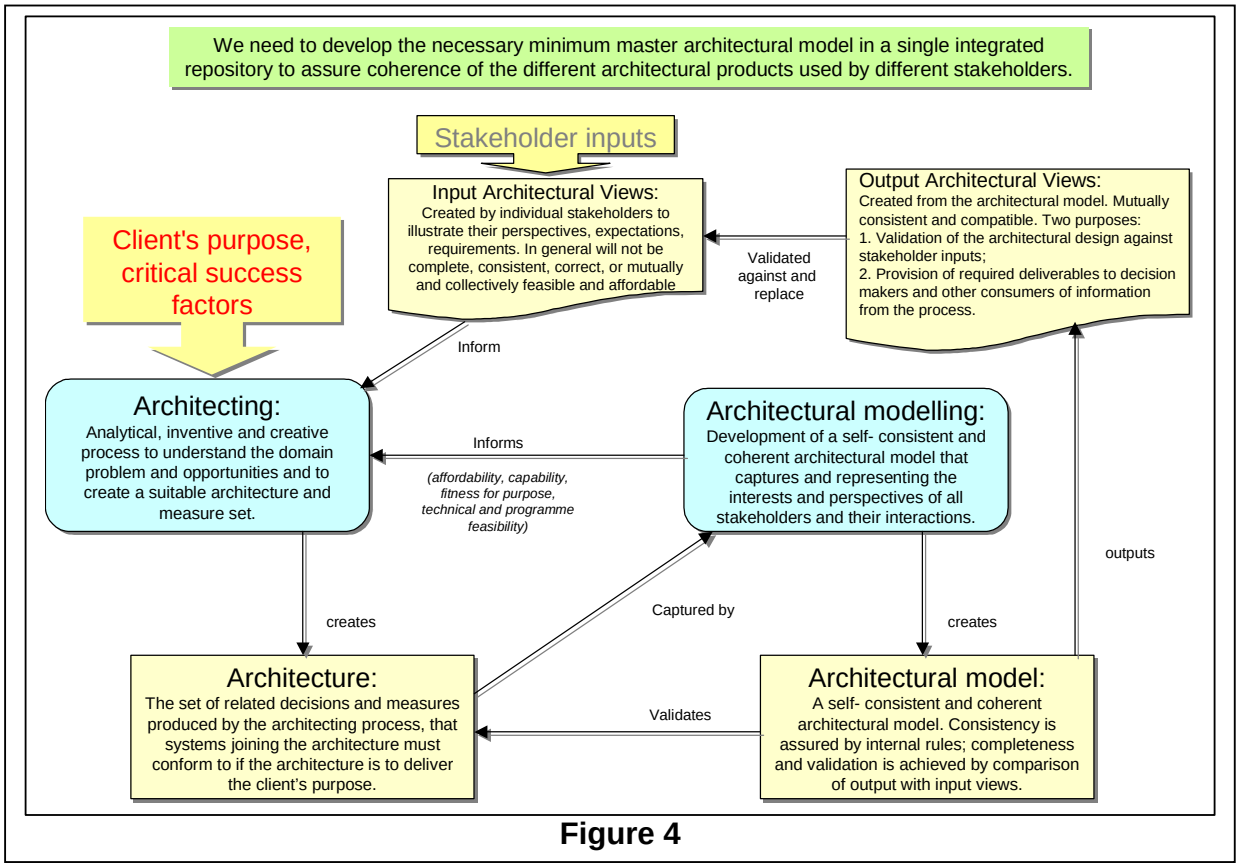

Figure 4

This leads to interesting stakeholder management issues. Some stakeholders may insist that their view of the problem is not open to debate, leaving the architecture modelling team in a “Catch 22” situation. If their original input is returned to the stakeholders unchanged, they perceive that the architecture analysis and modelling process has added no value. If it is changed they reject the changes.

Agencies responsible for managing and assuring acquisition and operation of Systems of Systems have a difficult judgement to make about how they mandate use of architecture frameworks. Over-mandation in an immature organisation will lead to a great deal of nugatory effort. Under-mandation means that key information will not be readily available for those who

need it and that project teams will continue to fail to deliver against system of systems requirements.

There has been a tendency to focus architecting effort in the early design and definition stages, so there is a lot of work on “as-required” architectures. This gets people thinking about the context and purpose of their system in the wider SoS context, and structuring requirements based on architectural insights, both of which are clear benefits. However it is very difficult to ensure that architecture representations keep step with the evolving design. This limits the extent to which detailed representations of systems in architecture repositories are likely to be current or correct. Since requirements are often traded out during development, and system features may not be implemented in the way envisaged by the architecture modeller, contractors and acquirers need to re-validate the information held in architecture repositories before they make critical design decisions. It is only worth building up large architecture model repositories if the models can be maintained and their configuration, currency and accuracy guaranteed.

This means that modellers need to resist the temptation to “model what they can model”, and should model only what is agreed, stable and useful. Some current tools produce the various MODAF/ DODAF views independently with nothing to constrain the views to be consistent. This allows fast work but does not guarantee integrity of the overall model. Many tools use a single integrated meta-model to ensure consistency. These tend to encourage modellers to build the views “bottom-up”, and may lead the modeller to build into the model assumptions that are not yet stable, in order to make it sufficiently complete to create the views. This can mean the model has to be substantially re-factored to accommodate relatively minor changes in design or context, leading to “process waste”, high maintenance costs, and a reluctance to invest the effort required to keep the models current and correct. It appears that tools designed to support architectural frameworks are not yet “fit for purpose” in terms of their ability to support and facilitate the natural flow of the architecting process without imposing constraints on the architect’s ability to develop his ideas using an agile blend of top-down and bottom-up thinking while supporting progressive coupling and consistency checks.

The usage of the architectural information and models, and the value of that usage, needs to be understood from the start, so that the work can be properly sized and funded based on business benefit; and incentives need to be devised to ensure that all parties maintain their part of the architecture. Current joint industry/government work in the UK is seeking to understand the role of MODAF within an open systems enterprise model. In the evolution towards this end state, the level of mandation should be sufficient to encourage mature teams, and those that are quick to learn, to do enough architectural modelling to get value out of the process for themselves; but conversely the mandation should be sufficiently “lightweight” that it does not put a significant burden on immature teams. The practice of teams hiring an architectural modeller for a few weeks to produce the views required for an assurance gate does not lead to team ownership of the architecture product or approach, and should probably be discouraged.

The desirable end-state of the use of architecture frameworks is that architecture framework representations are at a high level of abstraction, kept in step with a closed loop design and implementation process incorporating model driven design and test harnesses. The “as required” models of systems in architecture repositories need to be progressively replaced or supplemented by “as contracted”, “as designed”, “as tested” and “as accepted”. This means that architecture frameworks need to move from the assurance space into the design and management space; the work done in evolving the models over time should wherever possible be additive – adding detail

as it is firmed up. This will bring into architecting the lean principles that originated in manufacturing and are found to be very effective in “harder” design domains.

Meanwhile, few current tools adequately support “real” system architects because they do not mirror the way they think.

Two fundamental issues: operational capability is a wicked problem; and complex systems are created not top down but by aggregation of simpler systems

Operational Capability is a wicked problem

The following discussion is framed in the context of military capability but it applies equally to any enterprise–level objective that depends on purposeful system operation in a dynamic and evolving “open world”. Military capability is made up of a number of constituent elements, referred to in the UK as “Defence Lines of Development” (DLODs). These are doctrine (or, more generally, “process”), people, organisation, training, equipment, information, logistics, and infrastructure. Study of military practice, e.g. (Smith), leads to the conclusion that “*Military Capability – the ability to achieve a desired effect - is an emergent property of force structure, under command, deployed into a scenario and environment for a purpose.*”

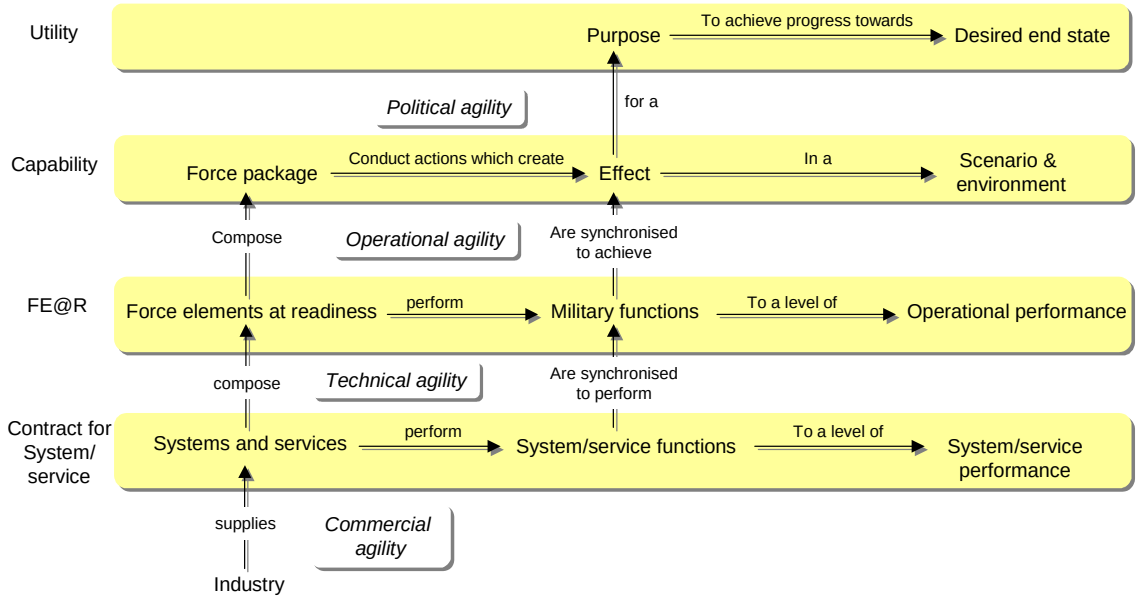

Figure 5: constructing military capability from its components

The necessary components of operational capability cannot be logically derived from decomposition of the capability need; rather, an iterative approach is required to synthesise an acceptable solution, with feedback to help us know when we are going in the right direction. There is no single “right answer”. Complexity is increased by the inherent unpredictability of the problem domain and the diverse views, backgrounds and attitudes of the stakeholders.

Therefore capability level systems engineering must manage the cognitive and “emotional” complexity across the stakeholder community, seek to quantify “what good looks like” in terms of a small number of agreed measures of merit, use these measures of merit to manage complexity in the “physical” world, and establish a robust and reconfigurable array of force

elements that can be configured by the operational commanders to fulfil their mission. This approach is made clear by Figure 5 (extracted from MOD Systems Engineering handbook draft D March 2007), which shows the problem as a series of layers with loose coupling between each.

This illustrates two fundamental issues for systems architects:

1. Finding a structure that matches the problem in hand
2. Understanding how to manipulate available elements to create desired emergent properties

Arbitrarily complex systems can be constructed from simple elements

We have already seen that systems requiring the attention of architects are complex. This suggests that architects need to be familiar with the concepts of complexity as it applies to systems engineering in their domain.

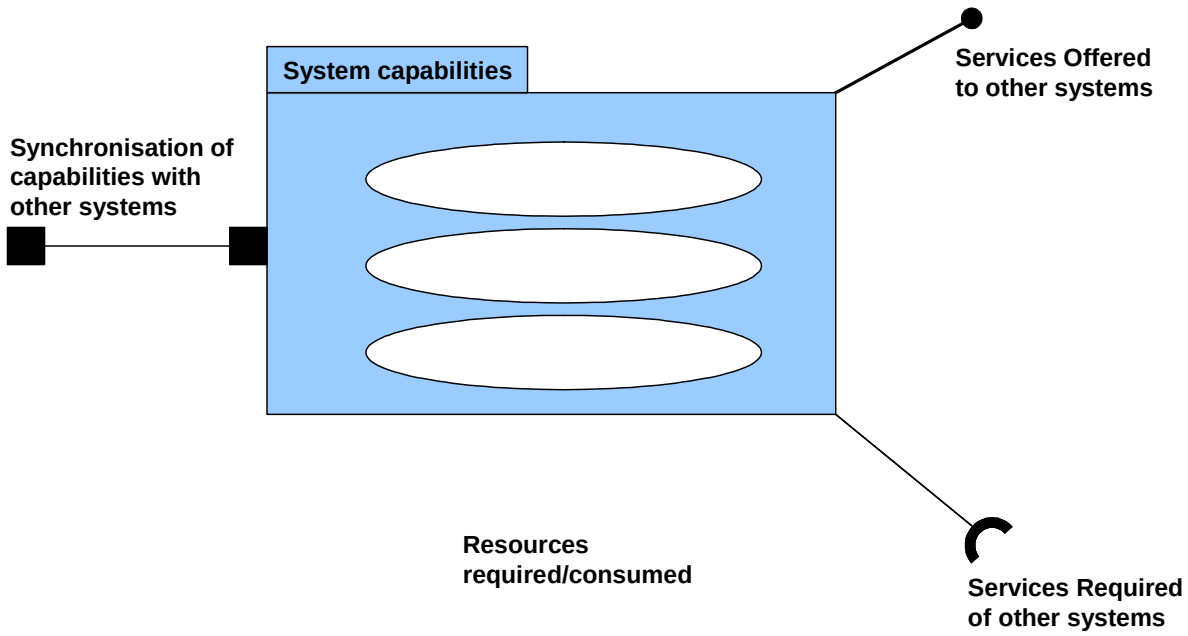

Figure 6: an element of a complex system

We know from the study of complexity that complex behaviour can be created by the interactions between apparently quite simple elements; and that arbitrarily complex systems can be synthesised by aggregation of such elements. Such complex systems are built bottom-up, but their emergent behaviour is difficult to predict bottom-up. The art of complex system design is to synthesise the required system properties through integration of dependable and fully characterised elements; and to maintain the desired properties during operation by embedding feedback and goal-seeking behaviour into the overall system design.

Borrowing ideas from service orientation, and also from the “building block” concept of (EIA 632) we can represent a generic component by the concepts in Figure 6. The element can provide and accept services, synchronise its behaviour, and requires resources.

Perhaps the same applies to architects.

Architecting is similar in different domains

There are different kinds of architects, as illustrated in Figure 7.

It seems that architecting within any domain has fundamental similarities. The differences are to do with the domain knowledge required to be effective in the domain, and the order in which major decisions are made and constraints applied due to the design and risk drivers in different domains. Figure 7 illustrates a common “architectural pattern” applied to different domains in which there are more or less clearly defined architectural practices.

We conclude in another paper offered for this conference (Sillitto et al) that the fundamental characteristic of an effective systems person is “understanding systems”. Thus a primary characteristic of an architect is the understanding of how systems behave, in general and in his domain.

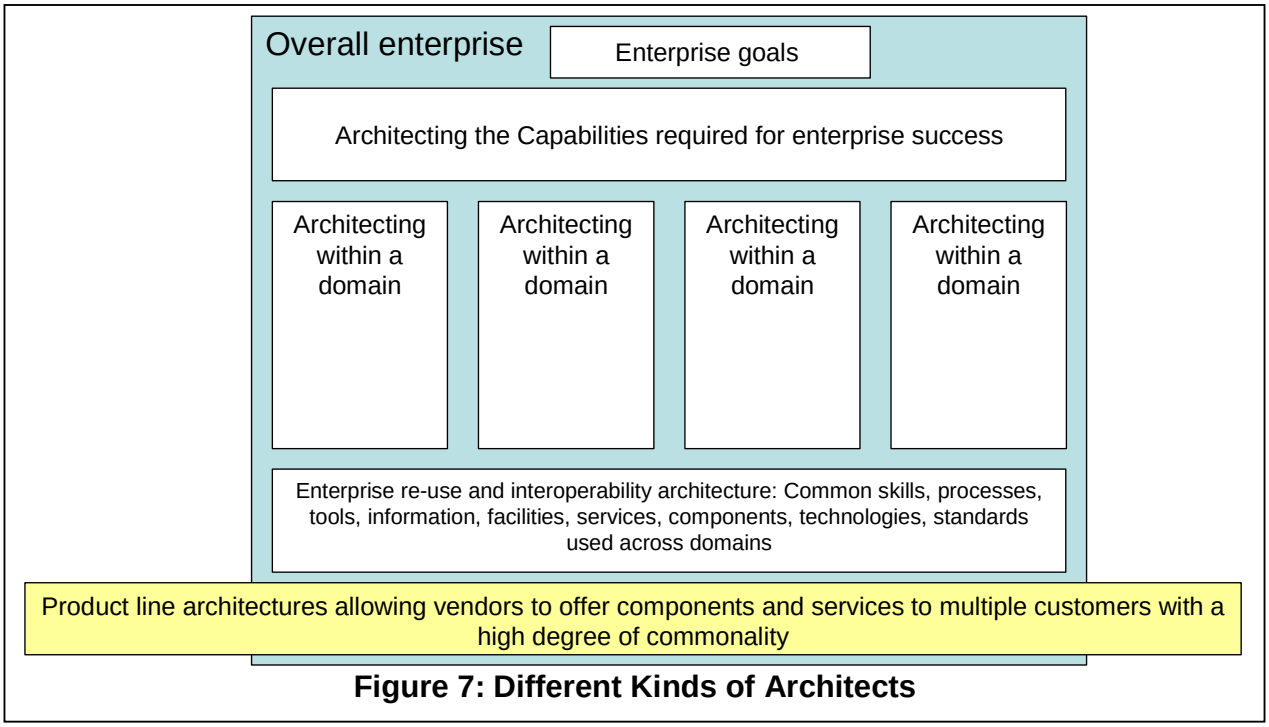

Figure 7: Different Kinds of Architects

Architects are or should be systems thinkers who apply particular forms of structured – or “structuring” - thinking to “open world” system problems. The architect has to discover structures and patterns that allow the architect to reformulate the problem from a messy “open world” problem into a set of “closed world” solution elements that can be integrated to provide a viable solution to the initial problem.

The architect may work in the client organisation, on behalf of the client, or in an industrial delivery organisation. Within the delivery organisation he may be responsible for a whole system, for a subsystem, or for a family of products. Particularly interesting is the concept of the

"product line". Topologically it turns out that the product line problem is identical to the "system of systems" problem as now widely understood: the need to satisfy a range of stakeholders with different purposes with a partially common solution.

Figure 8 shows the pattern of Figure 7 applied to a number of domains of architecting practice.

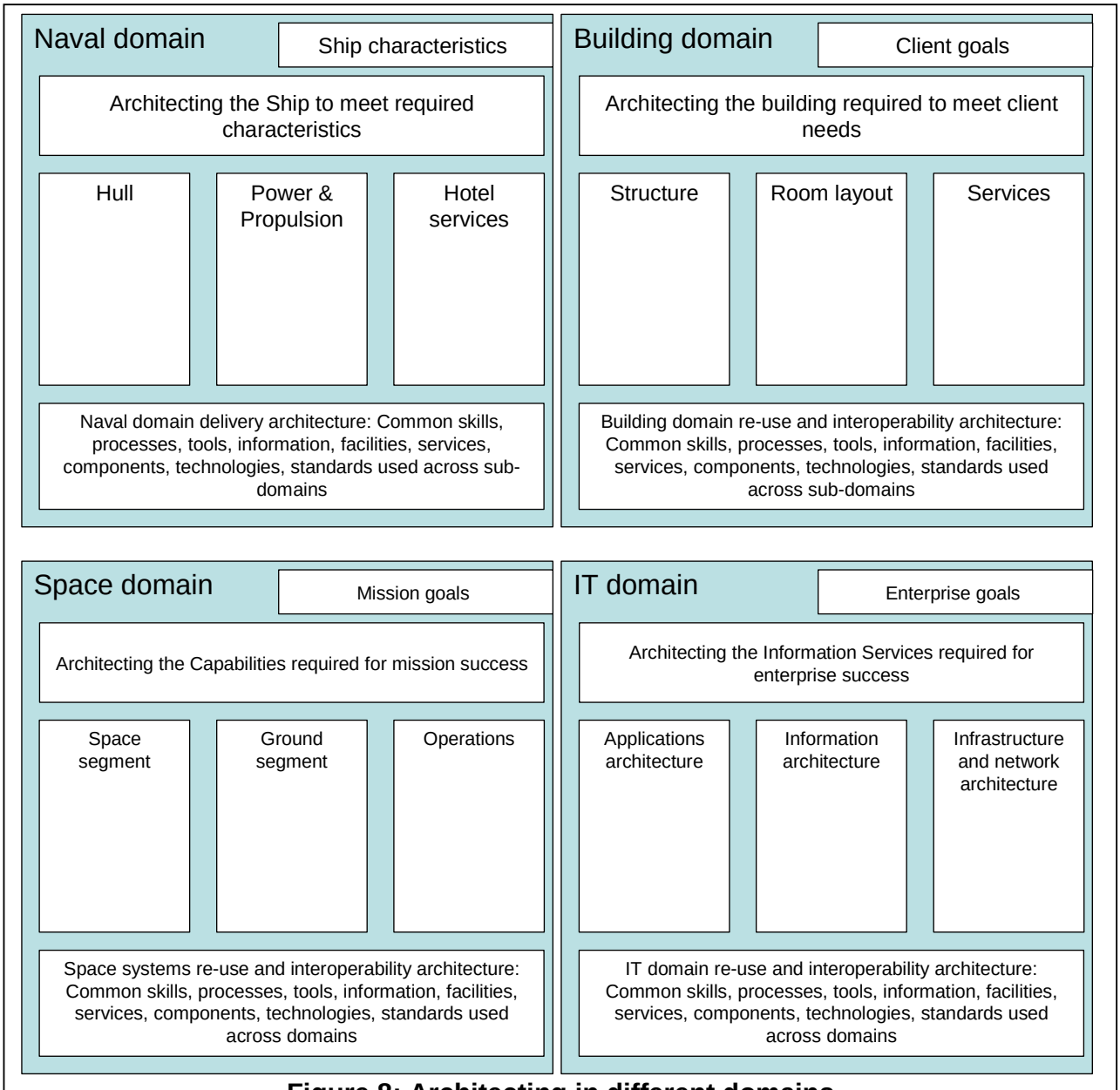

Figure 8: Architecting in different domains

Lean architecting

It is useful to apply lean and effects-based thinking to architectural efforts. If the job of the architect is to stabilise the rest of the project or programme, the architect needs to concentrate on:

1. Establishing and promulgating extreme clarity in customer value, quantified as a very small number of MOEs and KPPs (Measures of Effectiveness and Key Performance Parameters respectively).
2. Rapidly and expertly establish a clean, stable and robust high level architecture that
    - facilitates and structures customer interaction and concept exploration
    - can fully and affordably deliver whole life customer value,
    - maximises flexibility in implementation and deployment,
    - minimises and clarifies cross-coupling between different subsystems
    - hence minimises cross-dependencies between delivery teams.
3. Assist the design of the process and organisation to bring WBS, OBS and PBS into alignment at the highest practical level.
4. Understand the level of uncertainty and plan appropriate analysis, modelling, experimentation, de-risking and decision flow
5. Know when you are still in “discovery” phase versus when you can switch to “lean delivery”

The architects can then switch their focus to supporting delivery teams (now their primary customers) and work to minimise waste, NVA (non-value-added), waiting time and queuing across the programme, and avoid costly and time-consuming rework in the integration phase.

The essence of architecting

The essence of architecting is to understand and describe the purpose and context, and develop a system concept and process that will satisfy purpose in context, in order to “control the design” (including the organisation and the process) and define the approach to integration and acceptance of the solution.

Some architects work in the client space, and their job is to develop a model of the context and establish the purpose of proposed interventions, as an aid to client side decision support and acquisition. Others work in the supplier space, either within a single programme or to architect a product line that can meet a variety of client purposes with common elements. All should be systems thinkers with the ability to move rapidly between the problem and solution space, and between the subjective and the objective.

“Hard systems exist inside soft systems” (Blockley & Godfrey). Similarly, “hard processes operate inside soft processes”. Architects provide the interface between “open world” problems in the client’s domain, and “closed world” system and service solutions delivered through contracts. Architecting is a “soft and hard” process: a coherent blend of the “open world” soft processes concerned with stakeholder engagement, and the “closed world” hard processes for formally defining and delivering systems and services through “closed world” contracts.

The critical question is how architects approach a messy open world problem to understand it and develop a solution based on realisable elements. Some of the guiding principles are the traditional “cohesion and loose coupling”; these apply in many dimensions - physical, functional, behaviour, process, user, technology - - -. Other more basic approaches are drawn from physics: the concepts of “symmetry and elegance” that drive many theoretical physicists are also good

principles for system architects. Physicists may be the ultimate “reverse architects” – drawing out the fundamental structure of the natural world to explain and provide a unifying structure for observations that from a human perspective may appear baffling or incoherent.

Symmetry and layering are powerful structuring principles in complex problems. A key characteristic of architects is the ability to identify the underlying structure and dimensions in a complex problem situation and use this structure to master the inherent complexity. Structuring dimensions may include characteristic time constants and key logical elements of the problem and solution space, as well as the more commonly used physical, functional and performance dimensions.

Architects develop an integrated multi-dimensional abstract model to ensure coherence of their system concept. The integrity and completeness of this multi-dimensional model can be validated to an extent using an appropriate architectural framework. Architecture frameworks force the architect to describe his concept from a number of viewpoints: any gaps or incoherences in the concept are revealed by gaps in the model when viewed from the prescribed viewpoints.

Understanding, analysing, predicting, specifying and controlling system behaviour is an essential part of system architecting. The concept from complexity theory that we can create arbitrarily complex emergent behaviour from the interactions between apparently simple system elements proves the need for tools and methods for “hard” aspects of architecting sophisticated beyond the scope of current architectural frameworks.

Putting it to use

Simple practical guidance was developed from these principles and used with a group of students on the Sustainable Systems module available to MEng and MSc students at the University of Bristol. The concept of frameworks and architectural modelling was introduced first, through a lecture. Then we ran an exercise designed to get the students thinking about “similarities and differences” between different capabilities: a journey by plane or train, and sending a message by post or by e-mail.

All the groups started by capturing their ideas in the form of a “rich picture”, a method they had been taught to use previously, and were told to draw a simple use case diagram to identify the actors and their roles in the system. They were then encouraged to develop a structured approach to the problem by analysing process flow and behaviour of the actors in the system, and by identifying MOEs and MoPs. Swim lane diagrams and n-squared charts proved the most effective and intuitive tools for this. Valuable discussion points were raised about system boundaries – as far as the passenger was concerned, the journey started at the front door, whereas at least one transport service provider firmly set his boundary at the air terminal entrance.

A lecture the next day introduced the role of the system architect in an organisation, described the concept of the different “analysis and responsibility boundaries” of complex systems, showed how “descriptive architectures” could be developed to provide mutually coherent representations of systems at many different scales, and how these views and the framework that integrated them could be used to develop and explain a range of system sustainability issues, notably closed-loop

resource cycles. The lecture then illustrated a “prescriptive architecture” using the example of “the photography system” showing how the photography industry has exploited almost continuous technological innovation over the last 150 years (notably the recent transition from film to digital media) using a mixture of proprietary and open interfaces to support incremental migration at subsystem level. The concept of capability roadmaps was then introduced, showing how the structure of a UK MOD-style capability roadmap was equally relevant to the problem of migrating road transport from hydrocarbon fuel to hydrogen fuel cells.

All of the teams engaged enthusiastically in the exercise and produced useful artefacts; none was able to produce a complete and consistent set in the time available. Making the step from an integrated “rich picture” to a set of structured views of the problem clearly does not come naturally. We decided that future courses will provide more direction for the exercise and developed a simple structure for this purpose based on the various artefacts the students produced, shown in Figure 9.

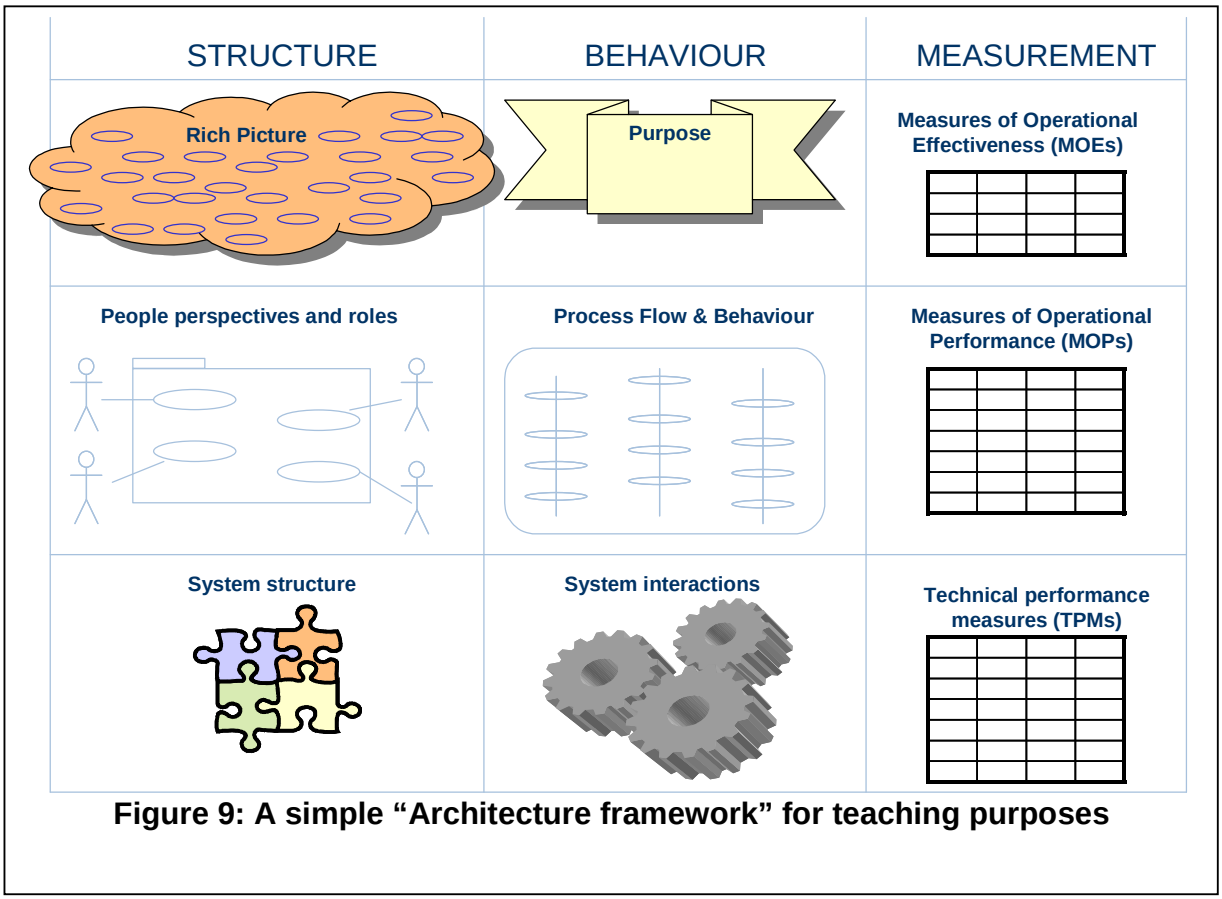

Figure 9: A simple “Architecture framework” for teaching purposes

Summary and conclusions

I have discussed at some length the principles and purposes behind the current generation of architecture frameworks used in many system-of-systems management agencies, and argued that investment in them must be managed judiciously until the community has reached a high stage

of maturity and they are better integrated into the mainstream of the overall systems design, delivery and operational process. I have also shown that current tool implementations of these frameworks do not map well to the way architects think and systems evolve, and that it is necessary for architects to step back to fundamental understanding of principles of systems and of effective communication with diverse stakeholders. Experience communicating these principles in an actionable way to students and colleague will be presented at the conference.

Acknowledgements

I am immensely grateful to countless professional colleagues in MOD, Thales, INCOSE and elsewhere for the interchange of ideas that contributed to the thinking that informed this paper. I am particularly grateful to Barry Trimmer (Thales Aerospace) for introducing the concept of "the two-headed Janus" to describe the nature of the architect's relationship with client and developer; and to James Martin (Aerospace Corporation and INCOSE) for numerous illuminating exchanges and in particular the dialogue that led to the creation of Figures 3, 7 and 8; and to Patrick Godfrey and Theo Tryfonas (University of Bristol) for their collaboration on the teaching exercise. I would like to acknowledge the support of Thales and UK MOD over the period during which these ideas were developed.

References

Blockley DI and Godfrey PS, Thomas Telford, ISBN 0-7277-2748-6, (2000): "Doing it differently - systems for rethinking construction".
Born, Max, and Wolf, Emil, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light* (7th ed.), Cambridge University Press (1999) ISBN 0-521-64222-1
EIA 632, Processes for engineering a system
Kemp, Duncan, and Linton, Rob: "*Service Engineering*", INCOSE IS Utrecht (2008),
MOD (UK) *Systems Engineering Handbook,* Draft D, (March 2007)
Rechtin, Eberhardt, Prentice Hall (1991) – "*Systems Architecting – creating and building complex systems*"
Sillitto et al, "Systems engineering meets professional recognition, INCOSE UK Spring Conference (2009).
Sheard, Sarah, *Practical Applications of Complexity Theory for Systems Engineers*, INCOSE IS Rochester (2005),
Smith, Sir Rupert, Allen Lane (2005)"*The utility of force – the art of war in the modern world*"
Warfield, John N, World Scientific Publishing (2006), "*An Introduction to Systems Science*"
Wilkinson, Bryant and King, *Architecture session*, INCOSE UK Autumn Assembly (2008),

Biography

Hillary Sillitto is Chief Systems Architect for Thales Land and Joint Systems UK. He has a Physics degree and started his career in optical system design, subsequently finding his systems skills useful in other domains and complex cross-domain projects. He was active in the Thales Systems Engineering network from 1996, served as INCOSE UK Chapter president from 2004 to 2006, and was seconded to UK MOD to run the Integration Authority from 2005 to 2008. He is a Thales Expert, a Chartered Engineer, a Fellow of the Institute of Physics, and a visiting Fellow at the University of Bristol.