model futures
www.modelfutures.com

MODAF and the Zachman Framework

Chris Partridge
Ian Bailey

Version 1.0 22 February 2010

Prepared for:

MINISTRY OF DEFENCE

Produced by Model Futures Ltd. under FATS/III

Model Futures Limited is a company registered in England and Wales with company number 05248454
Registered Company Address: 1 Nelson Street, Southend-On-Sea, Essex, SS1 1EG
VAT Number: 848 7357 75
MOD FATS/II: FATS/2/MFL
DGFM Supplier Code: 56945

Contents:

Executive Summary

The Zachman Framework and the MOD Architecture Framework (MODAF) have different origins and purposes. The Zachman Framework is an analysis tool for large and complex enterprises, intended to ensure that the whole enterprise is well understood (and planned) at business and technical levels. MODAF has its origins in systems and communications architectures and its structure reflects the MOD acquisition and systems engineering process. The Zachman Framework is structured around six key interrogatives; *what*, *how, where, who, when* and *why*. MODAF and its meta-model are structured around the concept of capabilities – abilities the enterprise possesses.

Both frameworks have a concept of levels of abstraction (in Zachman terms, reification). The ZF has six, and MODAF has either three or four depending on whether services are considered to be an intermediate level of abstraction. The abstraction levels do not correspond cleanly between the frameworks. The MODAF layers of abstraction are based on systems engineering principles (including soft systems). The top most abstraction layer is the Strategic Viewpoint which specifies capabilities independently of how they are implemented. The Operational Viewpoint is the logical layer which describes how elements of capability (nodes) are expected to interact and behave, but still does not commit to a particular implementation. Finally, the systems layer is the specification how the capabilities are implemented. The ZF layers are about adding new layers of detail, with each layer becoming more and more concrete. The ZF does not explicitly define capabilities and logical architectures.

Key findings are:

- The MODAF meta-model can be mapped to the Zachman Framework, but due to context-sensitive nature of Zachman, the mapping is loose in places (particularly in the *what* column). In addition, it is not clear that the concept of capability, or the strategic-logical-physical structure of MODAF maps well onto Zachman. See Appendix A for a summary of the mapping.

- Zachman captures the perspective of the enterprise using the context sensitive interrogatives. Version 2 of the Zachman Framework is intended to be an ontology, with the cells representing elements of the ontology. The ontology is context-sensitive, however, due to the cells being tied to the interrogatives (particularly the *what* cells). For example, if the enterprise manufactures widgets, the *what* cells will contain widget types, but if it is a human-resources enterprise, the *what* cells will contain types of role, post, etc. This would present a potential problem in using the Zachman ontology as the basis for any kind of federated architecture project. MODAF has been designed with federation in mind from the outset, so its meta-model is not context sensitive.

- MODAF would benefit from having a more intuitive structure. It lacks the symmetry of the Zachman Framework. An obvious underlying structure to MODAF does exist (see Appendix C), but this structure exists at a more superficial level - types of diagram / model. More analysis would be required to uncover the conceptual patterns in MODAF, but some immediate benefit would be gain from using a more intuitive view numbering structure.

- The Zachman interrogatives can be used with MODAF, both to capture architectural information (i.e. forming the basis for an architecture method – see Appendix B), and as an analysis tool for architectural data (see page 11)

- The Zachman Framework would gain a great deal by de-coupling its ontology from the framework itself. This would allow for federation, and would give EA tool/repository vendors a much easier target for implementation. Finally, it would also provide an easier route to a more defensible formal ontology built on formal logic.

Introduction

This document describes the findings of a study into two architecture frameworks; MODAF version 1.2 and the Zachman Enterprise Framework™ version 2 (ZEF). Particular emphasis has been placed on comparing the MODAF Meta-Model (M3) to the ZEF.

The MOD CIO EA team is looking for three key outputs from the study:

- An M3-ZEF mapping – examining the constructs of the Zachman Enterprise Framework (EF) and the definitions of the primitives of each of its cells, and to identify where these cells map to the MODAF Meta Model. The structures of the Zachman Product, Professional and Classification Frameworks were also examined to elicit a full understanding of the Zachman approach.
- A proposal for a business oriented analytical technique for using the Zachman EF to populate a MODAF architecture, based on the mapping exercise.
- Gaps between MODAF and/or Zachman EF to provide evidence for change.

In addition to studying the frameworks, an informal and brief straw poll of MODAF and Zachman users was carried out to elicit a user viewpoint on the key differences between the frameworks.

The general comparison of the two frameworks (and meta-models) has been carried out by Chris Partridge using a lightweight BORO™ analysis. The detailed mapping of the M3 onto the ZEF (shown in Appendix A) required detailed knowledge of MODAF and M3 and was carried out by Ian Bailey.

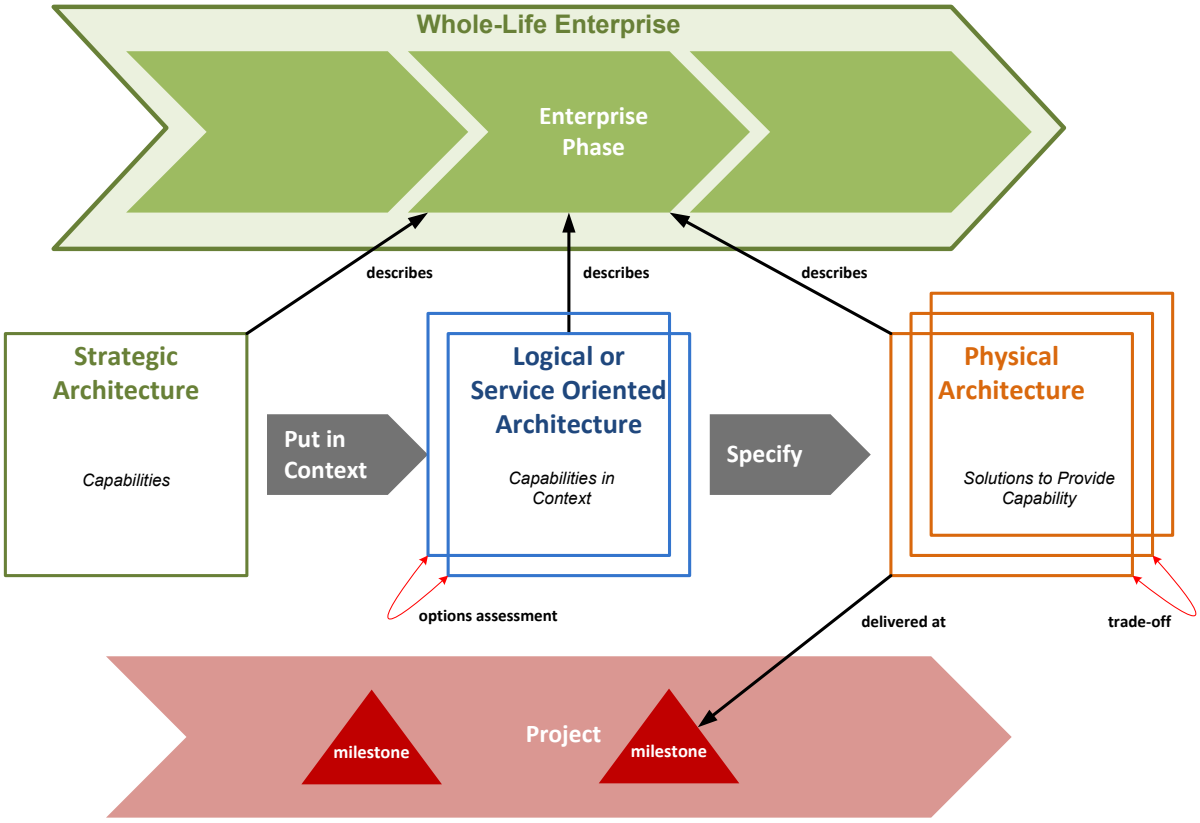
Key Differences Identified between Zachman Framework and MODAF

Architectural Principles

The Zachman Framework (which includes the enterprise, professional, product and classification frameworks) has evolved over a long time, refactoring and refining its set of underlying principles, resulting in its current symmetry and intuitiveness. MODAF is newer and so had less time to uncover its underlying principles . MODAF was originally developed from DoDAF, which itself had few underlying principles. Since version 1.1 of MODAF, some principles have started to emerge – for example, the stratification into strategic, logical and physical. Without a clear view of the underlying architectural principles in MODAF it has no compelling front end and organisation of the framework is awkward in some places, for example there remain some views which don't fit well within their viewpoint – e.g. StV-3,5, OV-4, SV-8,9. In addition, the service oriented views sit uncomfortably alongside the other views.

MODAF has been developed from user requirements. For version 1 of MODAF, rather than develop something from scratch, the MOD chose to enhance DoDAF with new views based on the MOD acquisition policy and processes. Successive versions of MODAF have fully integrated these new views with the existing DoDAF ones, and clarified some of the more obscure aspects of DoDAF v1.0. At one level, consistency between views is maintained by having a meta-model (the M3). However, developing a clear view of the underlying key architectural principles would provide a basis for making it significantly more consistent and intuitive, especially at the front end. Care must be taken in doing this though, as it is important not to lose any of the practicality of the MODAF views.

MODAF has a strong acquisition focus – where the ZEF was aimed at business change, MODAF intended to bring clarity and consistency to defence systems acquisition, and improve interoperability between systems. Since version 1.1, MODAF has adopted some sound systems engineering principles, with the strategic architecture roughly corresponding to a capability requirements document, the logical (OV) to a user requirements document and the physical (SV) to a system requirements document:



Any analysis of MODAFs architectural principles should place them in the context of general EA principles, which would provide a front end for both EA and MODAF in MoD.

Intuitiveness

A direct consequence of the ZF's top-down organisation based upon clear architectural principles is that people understand the ZF much more easily than other frameworks, including MODAF. Although there are some subtle and important nuances in ZF that are not immediately obvious, most intelligent users will understand the basic principles in a matter of hours. A brief straw-poll of MODAF and Zachman users revealed some key perceptions:

- The ZF has symmetry and consistency – this aids understanding and memory
- The ZF has the interrogative columns, which are very easy for people to understand, regardless of a technical or business background. The decision to use these was truly inspired. (Although there are some practical questions about how ZF uses them.)
- MODAF has no consistency in its view numbering – OV-5 has no similarity with StV-5 or SV-5
- MODAF has confusing names for views, and confusing concepts like capability and node

Clearly, if some of the simplicity and intuitiveness of the ZF could be brought into MODAF there would be great potential for improvement and uptake. Care must be taken though. The ZF is more easily understood by the layman than MODAF is, and its name is much more widely known (especially in IT) than MODAF. However, in terms of actual usage on architectural projects, MODAF is much more widely used "in anger". The ultimate architecture framework would combine the intuitiveness and principles of ZF, the user-driven pragmatism of MODAF, with a methodology of the clarity and simplicity of TOGAF™.

One of the reasons that MODAF has had more take-up on projects than ZF is that the ZF has complexities and nuances that are not immediately obvious at first glance. Where MODAF has a higher initial learning curve, once learned, there are no surprises. As the user reads more and more about the ZF, they learn about the subtleties and nuances.

Emphasis

Both ZEF and MODAF cover all aspects of the enterprise; business and technical. However, the focus of the MODAF views is chiefly around systems integration and capability management whereas ZEF is about whole-enterprise analysis. Also, MODAF has been tailored to the MOD acquisition process – hence it has concepts such as capability and project gateways/milestones. What this has meant is that MODAF tends to be used in more technical architecture situations.

That said, the concept of enterprise capability has started to find traction outside defence. In addition, the concept of trade-off between logical and physical architectures is also commonplace in systems engineering and business disciplines such as soft systems[1].

[1] See "Using Soft Systems with MODAF" - http://www.modelfutures.com/file_download/10/modaf_and_ssm_v1_0.pdf

Context Sensitivity

This difference between ZEF and MODAF did not become apparent immediately. The ZEF interrogatives are context sensitive – especially to the "what" column. So the ZEF's interrogative dimension is context sensitive. Furthermore, version 2 of the ZEF is described as an ontology for enterprise analysis. Each cell in the framework represents a model which contains two specific (ontological) types – these types taken together are the backbone of the ontology. Although MODAF does not claim an ontology, its meta-model (M3) serves a similar purpose. A key difference between MODAF and ZEF is that the ZEF is that each cell contains the views/model and its two-type ontology/meta-model whereas the MODAF views use a common meta-model/ontology. This means that in ZEF each ontological type appears in one and only one cell in the framework. In MODAF, several elements are used in each view, and any given element can appear in several views.

The consequence of ZEF tying the ontology to the context sensitive interrogatives is that it becomes context sensitive too. The types of things that appear in the first column ("what") can appear in different columns if the nature of the enterprise (and/or architecture) under analysis changes. For example, if the enterprise manufactures widgets, rows 1 and 2 of the first column of the ZEF will be populated with widget types. If the enterprise manages human resources, those same cells will be populated with types of post, role, etc. If the enterprise manages processes, then the cells will be populated with types of process. M3 is not context sensitive. The same M3 elements will be used to describe the same real-world objects and types always, regardless of which MODAF view they appear in.

The difference in context sensitivity between M3 and ZEF means that there can in principle never be a clean mapping between the two. It will always depend on the nature of the business being analysed. Context sensitivity is both a strength and a weakness. It allows a very clear focus on the enterprise and allows the intuitive interrogatives of the ZEF to be used to great effect. The downside is in integration. Ontologies cannot be reliably shared between enterprises and architectures – this is a serious issue in any federated architecture.

Recommendations for MODAF

Architect the Framework

MODAF needs a spring-clean. It was originally developed as a set of additional views on top of DoDAF. It has since developed by addressing user and vendor issues on a case by case basis. The only consistency in the framework has been provided by the meta-model – the views themselves show no meta-architectural design. Each revision of MODAF has been achieved on a shoestring budget (in some cases less than 5% of the budget assigned to successive revisions of DoDAF) so this not surprising. In fact, the MOD should be proud of having developed such a widely adopted framework with so little funding. The concepts developed in MODAF have been rolled-back into DoDAF. MODAF has been adopted by the UK intelligence agencies, the air traffic control service and by the Swedish Armed Forces. The NATO Architecture Framework is for all intents and purposes the same as MODAF. Finally, MODAF pioneered the idea of a UML profile for describing the framework – an idea now being taken forward by the OMG as UPDM.

Despite all this success, if MODAF is to have appeal beyond the systems engineering community, it needs to be sorted out into a logical and intuitive structure. As it stands, users all have subtly different interpretations of MODAF's architectural principles and so are producing architectures in isolation that will be difficult to align in the future. This is exacerbated by the fact that MODAF training courses and MODAF tools also impose their own interpretations on the framework, and there is no official MODAF method or process to follow. The lack of overarching structure and methodology means that MODAF is not immediately intuitive to new users. Finally, this lack of structure makes the task of maintaining the framework documentation and meta-model more difficult than it need be.

Appendix C shows one potential way of organising the views into a more coherent and memorable structure. This structure is little more than a cosmetic re-arrangement of views based on the type of content within them. Although this would bring about an immediate improvement to the framework, longer-term the MOD needs to decide on some meta-architectural principles for MODAF and refactor the framework based on these principles.

Enrich the Meta-Model

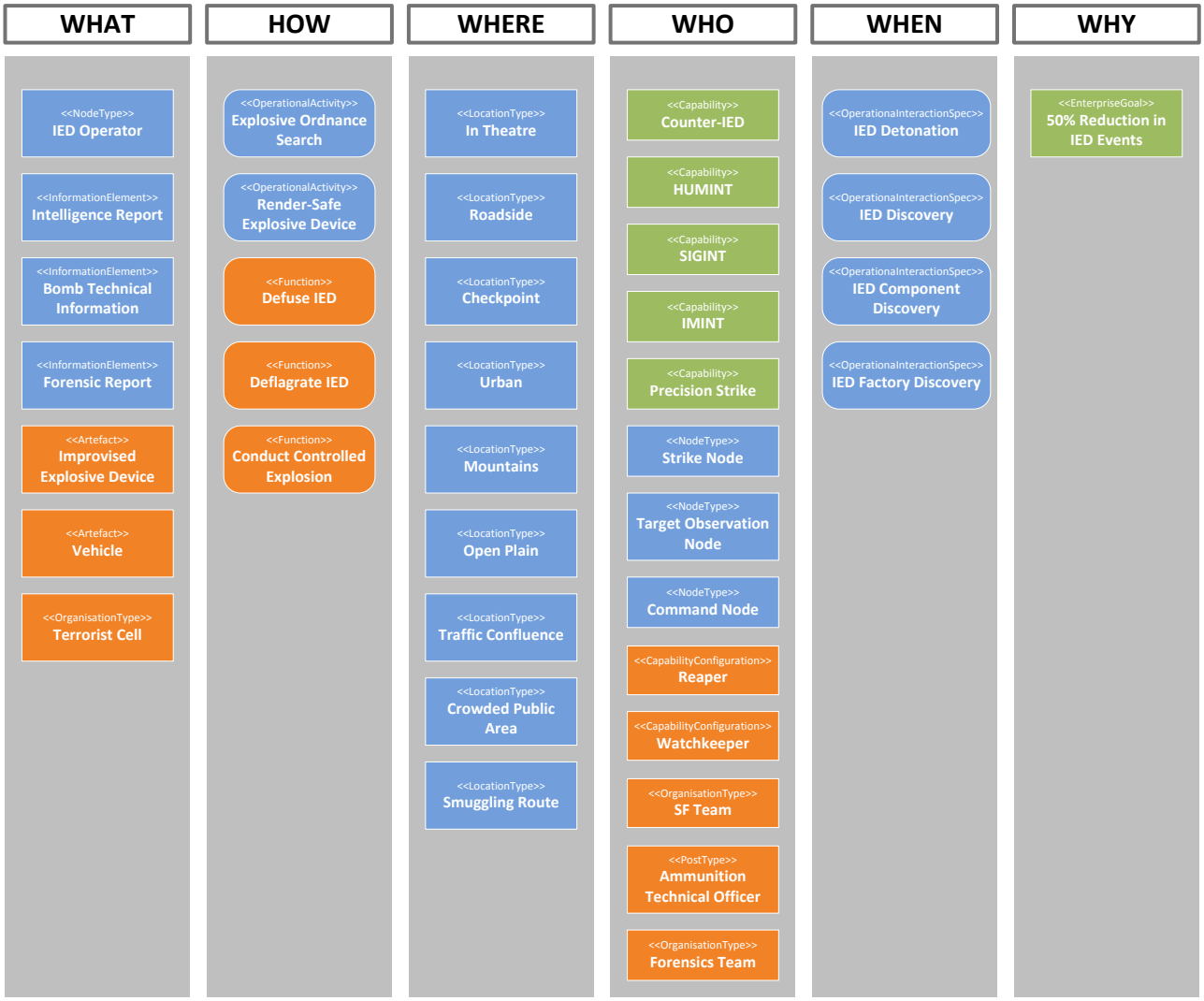
In most part, the mapping of the M3 to the ZEF was difficult due to ZEF being context sensitive and M3 being fixed-context. However, the mapping could have been easier if the M3 had a top-structure that was not based on UML. In many cases, the ZEF cells described concepts that were much broader than the M3 handles. In order to cover this, the analysis is either forced to show all possible M3 elements (this would be large number, particularly in column 1) or to resort to the UML metaclass on which they are based. The problem here is that the UML metaclasses are *too* broad. They are used for any number of purposes that would be outside the scope of ZEF cells. Two solutions are possible

- Add intermediate, abstract classes to the M3 from which common concepts can descend. Although probably the simplest approach, this presents some difficulties – in particular, common concepts may well descend from different UML meta-classes, which is likely to cause some problems in UML profile generation. In addition, ZEF makes an ontological commitment to things in the real world – individuals and types of individuals. UML does not.
- Re-base M3 on IDEAS. This approach, although involving more effort than the previous, would provide an ontology that is equally applicable to ZEF and MODAF, allowing both frameworks to be based on a common core. This would eliminate all the UML baggage that is currently present in M3.

It should also be noted that the DoDAF meta-model has also used IDEAS as its basis, so this approach would offer greater potential for alignment between MODAF and DoDAF.

Use the Zachman Interrogatives to Analyse Architectures

It is quite hard to consistently align the rows of the ZF with MODAF, but it is much clearer how the columns can be used. One option is use the columns (the interrogatives) with MODAF architectures to ensure completeness of enterprise capture. Initially, the interrogatives could be used to categorise elements in the AV-2 – perhaps even being a new AV view (AV-3 ?) in its own right:



This approach takes advantage of the context sensitive nature of the ZF (see section on key differences between frameworks) to show how each element is viewed in the context of the particular enterprise. In the example above, Watchkeeper appears in the WHO column, but if the architecture was concerned with support of the Watchkeeper asset, that element would appear in the WHAT column. This approach allows the M3 to keep its context-independence (enabling re-use of elements across architectures) but allows the context-dependent nature of the ZF to be used for analytical purposes.

The MOD could go further with this, and use the interrogatives as the basis for a methodology for populating MODAF architectures. Appendix B shows one potential process that could overlay the Zachman and MODAF frameworks.

Another option is to use a traditional application of the interrogatives in a single view.

Recommendations for Zachman Framework

De-Couple Ontology from Framework

In working with John Zachman and Stan Locke on this mapping, it was mentioned several times that they've struggled to get a tool vendor to implement the ZEF in a satisfactory way. One reason for this may well be the fact that the framework is its own meta-model – each cell is meta-class. If the ZF were to specify its ontology (meta-model) separately from its rows and columns, it would offer far more opportunity for real-world usage and implementation. As it stands, an ontology based on six rows and six columns is not sufficiently sophisticated to enable development of tools. It *is* very useful for analysing an entire enterprise, so care must be taken not to lose this functionality.

One of the main issues with the strong-coupling of ontology and framework is the assertion that each cell represents a different thing. From an ontological perspective this only works within the context of a given enterprise, and even then may cause problems. This is particularly evident in the WHAT column. The WHAT for one enterprise may well be the WHO, WHERE, or HOW for another. If an enterprise is sufficiently large, it may well even be that some of the same things could appear in more than one column. A de-coupled ontology would allow this to be managed in a consistent and coherent manner by enabling the same element to be re-used from column to column where necessary.

Formalise the Ontology

Basing the ZF ontology on the cells of the framework gives the user a good intuition for what each element in the ontology is (i.e. what it signifies in the real world). However, this may not be entirely clear in all cases. In the case of the WHO column, it is not clear whether this is concerned only with human actors or whether it also allows for the involvement of automated systems that may also conduct the HOW. Secondly, the abstraction relationships from one row to another may not be what they at first seem. This is particularly evident in the WHAT column, where rows one and two appear to cover types of things in the real world. Rows 3 ,4 and 5 appear to cover representations of real world types (e.g. information, data). The question then is what appears in row 6 for column 1. Is it data instantiations (i.e. instances of rows 3-5) or real individual things (i.e. instances of rows 1 and 2) ? This issue is outlined in more detail in the section on Use-Mention Distinction on page 22.

If the ZF were to base its ontology on a formal upper ontology, these questions would be exposed in sharp relief, and could be clarified. This would also formalise the mechanisms of abstraction (transformation) between ZF rows – something which is would lower the barrier to implementation, and de-mystify some of the less well understood aspects of the ZF.

Analysis

Summary of Analysis

The MOD tasked Model Futures to map the Zachman Framework (ZF) into MODAF in the context of a need to provide an easier to understand front-end and an implicit analytical approach to EA. This was clarified in a meeting on 14th December, where initial concerns about just mapping the cells were raised and it was agreed that a top down approach was important. The approach we took was discussed and further agreed in a meeting on 21st December.

The comments below on the ZF are made based upon a review of the documentation supplied, and so may not reflect a proper understanding of Zachman International's 'official' view of their framework. They may need to change when the document is reviewed (and points clarified) with an SME from Zachman International.

In particular the analysis has assumed (or inferred) that:

- The ZF interrogative and reification dimensions provide a facetted classification of the enterprise objects.
- The ZF interrogative dimension is context sensitive (and this is not currently documented).
- The default transformation between rows is the identity transformation (in general, the semantic characteristics of the transformations were not documented – these may be specific to the type of model used).
- The ZF transformation from Row 2 to Row 3 (or maybe Row 3 to Row 4) for Columns 1 and 2 involves a jump up one level of representation and so is not identity (and this is not currently explicitly documented).
- The ZF has no top level ontology at a similar level of (metaphysical) details as the IDEAS top level (in other words, this was not in the documentation provided).

The analysis identified two key points relating to the explicit structures of the frameworks that provide a context for any mapping between the two. Firstly that there is striking difference in the richness and clarity of the top level structures. The ZF has a clear top-down formal structure with a rich backstory. MODAF has some explicit top level structure, but this is weaker than ZF and lacks an articulated rich and coherent backstory - though the material for this seems to exist implicitly in the lower levels and in actual practice. Secondly, the top level structures of the two frameworks have fundamental differences (as they stand today) that make a simple one-to-one mapping difficult. This goes some way towards explaining why ZF is recognised as having an easy to understand front end and MODAF does not. In particular, ZF's 6 x 6 matrix has a clear consistency and symmetry that aids understanding and MODAF has nothing similar.

The structures of the Zachman Product, Professional and Classification Frameworks were examined and compared with MODAF. The examination raised a number of points, of which two were key. Firstly that one of the key reasons for ZFs symmetry and intuitiveness is that it is explicitly constructed from a set of explicit underlying principles. And secondly that Zachman and MODAF have made different choices. For example, MODAF has separated its views from an underlying common model, whereas Zachman has a model for each view (cell). Both these points can be regarded as meta-architectural – as about the architecture of the

architecture. This makes clear that an easy to understand frontend for EA needs to be built upon a clear meta-architecture, one with clear meta-architectural principles and explicit meta-architectural choices.

The analysis technique used here is a version of BORO-Lite that involves reverse-engineering the meta-architecture from both ZF and MODAF to identify the core EAF meta-architectural principles (and patterns) and meta-architectural choices. The initial results are described below.

The results illustrate the technique. If this analysis was completed then these principles and choices could be choreographed into both an easy to understand frontend and an implicit analytical approach to EA. One outcome of this would be to identify improvements to MODAF.

Meta-Architecture History

A starting point for understanding ZF and MODAF is their history. It is plain that they are at different stages in their history and that they have been developed to solve different problems.

Zachman Framework

The Zachman Framework was first published in 1987 (Zachman J. A. (1987). A Framework for Information Systems Architecture, IBM Systems Journal, 26/3, 276–295). As the title suggests, the original focus was on the architecture for Information Systems. It was quite radically extended in 1992 (Sowa J. F. and Zachman J. A. (1992). Extending and Formalizing the Framework for Information Systems Architecture, IBM Systems Journal, 31/3, 590–616). More recently version 2 of the framework has been issued.

One of its strengths is that it has, over time, worked successfully at developing a clear set of underlying principles. Providing a simple framework for these principles is one of the key reasons for its intuitiveness.

MODAF

MODAF was developed more recently than the ZF and has not been designed from the top-down. What explicit architectural principles there are in MODAF lie in the meta-model and in the stratification of the Strategic, Operational and Systems views.

Formal Structure of the Top Level

In order to provide a mapping, it is necessary to develop a clear picture of the formal structure of what is being mapped. Where there are differences in the formal structure, any mapping will need to take account of these. The description below shows clearly the differences in the two frameworks formal structures

Zachman Framework

The ZF's top level is represented by a 6 x 6 matrix with two dimensions. In classification terms this is a two faceted taxonomy (where the dimensions are facets). It has the advantages of faceted classification in that it provides a "clearly defined, mutually exclusive, and collectively exhaustive aspects, properties or characteristics of a class or specific subject" (Taylor, A. G. (1992). Introduction to Cataloging and Classification). As such it has clear advantages in setting out the scope of an architecture. From a mathematical point of view the structure is a simple lattice of cells.

These cells contain models (the ZF does not mandate a particular type of model) – hence the top-level structure classifies initially models and through these indirectly the contents of the models. In addition, there are transformation mappings between the cells that link the elements in the models (and so the models). When these transformations are included the mathematic structure becomes a network graph. Originally ZF placed an additional ordering constraint on both dimensions – which translates into a constraint that the transformations between cells only need to be to neighbouring cells in the same row or

column. In the latest version of Zachman (ZF 2.0), the ordering constraint on interrogatives was removed – so there can be transformations between any of the cells in the same row.

MODAF

MODAF has two top level 'classification' structures: its meta-model – M3 – and the views (and viewpoints) over M3. Unlike ZF, MODAF mandates a single underlying model whose structure is described by M3. M3 amalgamates a number of existing modelling techniques, such as state, process and sequence diagrams. The top levels of this structure have evolved from the bottom up aggregation of these modelling techniques, and as things stand today there has been no segmentation of this into a top level that could be regarded as equivalent to the ZF top level. From a formal mathematic structure perspective it is a network graph, though there is significant additional structure.

The MODAF views are organised into viewpoints – where each view can only belong to one viewpoint. This is mathematically a two level tree structure. The views (and so the viewpoints) cover the (M3) model. In other words, each element in the model will appear in a view. However, the views are not exclusive (in other words, they overlap) and so an element may appear (and often does) in more than one view.

Comparing the Zachman and MODAF formal structures

The formal structures of the two frameworks reflect one aspect of the trade-offs that have been made when selecting a top level entry point to the architecture. One trade-off is between expressiveness and ease of understanding: the more formally sophisticated the structure the richer its expressiveness, but the greater the intellectual effort needed to grasp it.

Both architectures have opted for simple structures. However, the simple stratified two-level tree structure used by MODAF's viewpoint-views is structurally less expressive than ZF's two facetted classification scheme. This makes it less likely to be able to support a mutually exclusive breakdown – which indeed it does not (it only provides coverage (completeness)). The top level ZF promotes the two facets/dimensions and deprecates the transformations. This draws out the lattice/matrix structure from the network structure.

Zachman MODAF formal mapping issues

Examining the framework from the perspective of their formal structures also highlights the formal issues of mapping from a more to a less sophisticated structure (or vice versa) – it is mathematically impossible to make a simple one to one mapping from a two level tree to a two facetted lattice. However, the facetted Zachman structure enables one to look at its facets in isolation – giving two simple one level tree stratifications.

Zachman and MODAF model elements

The mappings need to take account of the content (or ontological structure) as well as the formal structure, they need to reflect an understanding of what is being mapped. Both the MODAF and ZF architectures contain representations (model elements) that are organised into models. In MODAF, these representations are produced under the M3 meta-model. In ZF, no particular style of modelling is mandated. So while there will be differences in the forms of representation (at the simplest level, one form may be a box and the other an ellipse), these different forms can represent the same things. What the MODAF to ZF mappings need to ensure is that they do not result in a change in the underlying things being mapped, that the model elements continue to represent the same things.

Ensuring that the mappings work in this way turns out not to be straight-forward because (as described below) the broad classifications within MODAF and ZF have very different bases.

Meta-Architectural Principle – Understanding the Architecture - Interrogatives

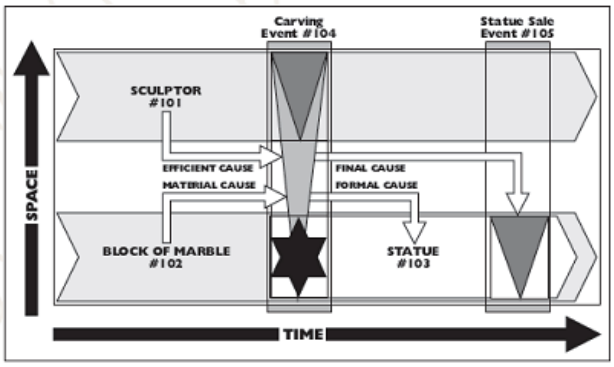
John Zachman's 'Concise Definition of the Zachman Framework' says "The Zachman Framework … consists of a two dimensional classification matrix based on the intersection of (What, Where, When, Why, Who and How) with six rows according to reification transformations"

These two dimensions (six communication questions and six reification transformations) provide the starting point for analysis of meta architectural principles.

The 'Six Communication Questions' - What, How, Where, Who, When, Why - are also known as the interrogatives. They have a history going back to the Ancient Greeks (particularly Aristotle) as a means for understanding and a framework for explanation. Interrogatives are well established as heuristic analysis tools and currently used in a number of fields including journalism and police investigations.

Traditionally the explanatory power of the interrogatives comes from having them in a single interlinked picture. An example of this using the Aristotelian causes (from p. 177 of Business Objects, Partridge 1996) is given below.

Figure 8.26: Sculptor carving a statue space-time map



Each of Aristotle's four types of change appear in the space-time map. Their links to the carving event are illustrated with arrows. Underlying each arrow is a pattern of connections between the extensions of the objects involved in the statue carving event

What these interrogatives offer an Architectural Framework (AF) is an analytic technique to capture a complete explanation of the architecture within a context. The explanatory aspect aids with understanding and communication. The completeness aspect provides quality assurance.

MODAF currently has no views or viewpoints based upon the interrogatives.

ZFs use of interrogatives

ZF uses the interrogatives in a novel way, as a basis for formally partitioning models - and, more strongly, the things in the models. Of course, the Zachman Framework has bridging links between the partitioned models, but these have been deprecated for the top level and do not appear at the top level. Although he does recognise their importance when he says that "It is the integration of answers to these questions that enables the comprehensive, composite description of complex ideas."

Zachman explains his choice as arising from looking at how enterprises were modelled, where there are a number of different views. However, this cannot be the whole story as it has been noted a number of times (in respect of the ZF) that there are not currently good modelling techniques for all the interrogatives – for example, there are no industrial strength methods for modelling Why or motivation.

ZFs approach makes a stronger assumption than the traditional approach, in assuming that every thing of interest can be allocated to an interrogative. The weaker assumption of the traditional approach is that the interrogatives would identify those things that explain the architecture – not every thing of interest.

Context sensitive interrogatives

The analysis of potential MODAF M3 to Zachman mappings revealed an interesting characteristic of the interrogatives, they are context sensitive (M3 is not context sensitive). It became plain that whether an M3 element appeared in the columns – and which column it would appear in - could depend upon what was being architected.

A simple example would be:

- An architecture to build the organisation structure within a factory. In this case, the 'what' would be the 'organisation roles' within the factory – there are what is being architected.
- An architecture to run a production line in a factory. In this case, the 'what' would be the production line and 'who' would be the 'organisation roles', as these are who are responsible for the production line.

This means that the formal partition into interrogatives can be specific to an architecture and so cannot reliably be re-used in other architectures. Any use of the interrogatives would need to be sensitive to this characteristic.

There is a strong argument that an architecture needs to be an enterprise's view of the world and reflect its values. One can see the interrogatives as classifying from the particular enterprise's perspective. One could make a classification of M3 elements into interrogatives (and so a mapping) for a particular enterprise (though appreciate that different enterprises can have different mappings).

Zachman proposes a single architecture for an enterprise, and in this case each enterprise would have a single mapping. However, where an enterprise takes a federated approach (as the MoD does – and MODAF supports), one would need to be sensitive to the possibility that different parts of the enterprises will interrogatively classify things in different ways (as they have different perspectives).

An alternative approach

From a practical perspective, it would make more sense for MODAF to consider following the traditional approach and integrating the interrogatives into a view (maybe existing views) rather than treating it as a dimension.

It would be worth considering the state of the art in modelling interrogatives / explanation when deciding to how to include it in. The current state of the art is not well-advanced and there are no real standards available. It may be best to regard the interrogative analysis as a heuristic support (building on established work) for building the model rather than a formal model.

Another avenue worth exploring would be to do an interrogative analysis of EA/MODAF. This should give some good input into any front end.

Meta-Architectural Principle – Perspectives

In version 2 of ZF, the second dimension is derived from reification, the transformation of an abstract idea into an instantiation and the elements are labelled: Identification, Definition, Representation, Specification, Configuration and Instantiation.

In an earlier version of ZF these were associated with these perspectives:

- Planner's View (Scope)
- Owner's View (Enterprise or Business Model)
- Designer's View (Information Systems Model)
- Builder's View (Technology Model)

- Subcontractor View (Detailed Specifications)
- Actual System View or The Functioning Enterprise

One can rationalise these by saying the latter version characterises what the model is and the earlier version whose perspective the model is from.

The ZF rows, unlike the ZF columns, have a specific order, a natural progression from top to bottom.

The original ZF perspectives were drawn from the working practices of a variety of industries and so reflect the way in which those industries have evolved management structures to deal with large, complex projects. With a little research one can find other business models, which would lead to different breakdowns. What is important about these perspectives (and the reification breakdown) is that taken together they form a complete picture of the enterprise being architected.

MODAF has four viewpoints that could be interpreted as perspectives on the architecture.

- Strategic Views (StVs) define the desired business outcome, and what capabilities are required to achieve it
- Operational Views (OVs) define (in abstract rather than physical terms) the processes, information and entities needed to fulfil the capability requirements
- Service Oriented Views (SOVs) describe the services, (i.e. units of work supplied by providers to consumers), required to support the processes described in the operational Views
- Systems Views (SVs) describe the physical implementation of the Operational and Service Orientated Views and, thereby, define the solution

However, there does not seem to be a clean mapping from the MODAF views onto the ZF rows. This is not a concern in itself; it may well be a reflection of the MoD's different business model. However, there is no rationalisation of the division of viewpoints that would justify regarding them as reasonably complete.

The other three MODAF viewpoints are not perspectives on the architecture.

- Acquisition Views (AcVs) describe the dependencies and timelines of the projects that will of deliver the solution
- Technical Views (TVs) define the standards that are to be applied to the solution
- All Views (AVs) provide a description and glossary of the contents of the architecture

The Acquisition Views viewpoint is interesting in two ways. Firstly, it can be generalised from a MoD specific acquisition view into a more general program view and so to the management of the architecting process. Secondly that it can be seen as a single perspective looking at this process rather than the architecture.

From an understanding point of view, it would be useful categorise and rationalise the MODAF viewpoints and provide a simple high-level explanation of the relations between them.

Meta-Architectural Principle – Cross-cutting concerns

One of the key features of the ZF is the inter-linking between the two dimensions, the rows and the columns. As noted earlier, this has a rich formal structure. To see this consider each cell as a view and each column and row as a viewpoint (a collection of views/cells). Then the ZF viewpoints have a rich structure. They break down into two distinct groups (dimensions), where each view belongs to one viewpoint from each group. This is a facet structure (as seen in the Colon classification). This rich structure reflects a well-thought out rich classification framework.

ZF places an additional constraint on the transformations from one column to another, they can are only be to the neighbouring cells in the same column. This forces column to column transformations to stay within the same interrogative. This would only work if the transformations did not change the interrogative classification. While this may be true, we cannot find where it is explained why it is true.

By contrast the MODAF Viewpoint-View structure is a simple tree, where each view can belong to only one viewpoint. Furthermore, the viewpoints have no further formal structure. This implies a single dimensional structure. It may be worth reconsidering this tree constraint, as it is relatively easy to change. MODAF has, no doubt, a rich implicit structure, and were this to be made explicit it might require a more sophisticated formal structure.

Meta Architectural Choices

ZF and MODAF are the result of a various meta architectural choices. Three are relevant here: Ontology, Top Ontology and Model per View or Views over a Common Model

Meta Architectural Choice – Ontology or Not

The designers of an EAF can choose whether it has an ontology, and how to implement it. Both ZF and MODAF have a commitment to using an ontology, but from different routes. From a ZF point of view an EA has to be grounded in an ontology – and this is central to his framework. Hence he makes a strong commitment, and his commitment is unequivocal (see his comments in Appendix D). Note that this is to ontology as "a theory of the existence …" rather than as an RDF data structure. For MODAF the choice of a common underlying model leads naturally to an ontology as a basis for the commonality.

ZF's ontology corresponds to its two dimensions. Each of the 36 cells is characterised by the (two) types of entity it contains. For example, Column 2, Row 1 contains business entities and relationships. Within an architecture, things are classified into one and only one cell – in other words, once to each dimension. These have some formal characteristics (enough to make the framework work) and intuitive criteria of identity). While cells may contain their own types of relationship, a special set of transformation relationships link entities in one cell with entities in another. ZF makes no detailed ontological commitment below the high level ontological categories allocated to the contents of its cells.

As noted earlier, the M3-ZF mapping shows that one result of ZF's choice of tying its ontology to its views/cells is that the ontology is context sensitive. Different architectures can have different ontologies.

MODAF's ontology is M3. This is an integrated single model across the whole architecture. Where it makes sense MODAF extends this model to identify the various types of things it would expect to see in views, so it goes to a far more detailed level than ZF. Through experience MODAF designers have found that the M3 should not include context sensitive elements, and have excluded these. This lack of context sensitivity allows for ontologies to be shared across architectures, important in a federated environment.

Unlike ZF, MODAF does not make such an explicit commitment to an ontology. In some ways this is odd given how central M3 is to its architecture.

Meta Architectural Choice – Top Ontology or Not

If the designers of an EAF have chosen to give it an ontology, they can then choose whether (or not) to give it a top ontology. If the choice has been made to have an ontology, then it is extremely useful to have an organising top level ontology. Both ZF and MODAF have a commitment to a top ontology. However, ZF's choice is, from an ontological point of view, less well grounded. One reason for this may be its strong linkage to the cells in the framework.

MoDAF is moving towards the BORO/IDEAS Foundation which is an extensional four-dimensional ontology (with the formal structure of set theory). This has three well-founded ontic categories. Each category has formal characteristics and a criterion of identity. These categories can be related in a number of ways – however, every object belongs to one and only one category. There appears to be nothing this well grounded in ZF.

We have found some references (see below) in ZF to what we would call the top ontology, which tells us how it might be fleshed out. But we cannot find any published structure.

Z101 (paper)	Extract
No page number	Models of the Models … Meta-meta Model of all the Cells of Column 1 Entity – Relationship (-Entity) Meta-meta-meta Model of all the Cells of the Framework Thing – Relationship (-Thing)

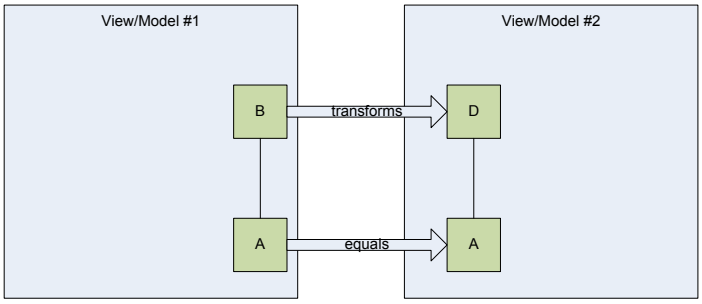
As noted earlier, it became clear during the mapping exercise that the classification of things into the cells (and their ZF ontological categories) was context sensitive. In other words, that in different contexts the same things would appear in different cells.

One would expect this to cause problems for the top ontology as its categories would need to be context insensitive. Whether this is actually the case for ZF would require closer examination of its structure.

Meta-Architectural Choice - Model per View or Views over a Common Model

ZF and MODAF illustrate this choice. ZF has chosen a model per view (cell) whereas MODAF has gone for a common model approach. If an EAF makes the Models per View choice, then it needs to have an integration framework that links the models (as ZF does). If an EAF makes the Common Model choice, then it has to enforce the use of a common model. MODAF mitigates this by prescribing a common data interchange format.

The integration framework maps representations in one model onto representations in another, as shown below.



These mappings need to have a clear semantics. The simplest kind of mapping is (Wittgensteinian) identity (where representation A in view #1 represents the same thing as representation A in view #2). However, this is unlikely to be the only kind of link, and the semantics for the other kinds of links need to be made clear.

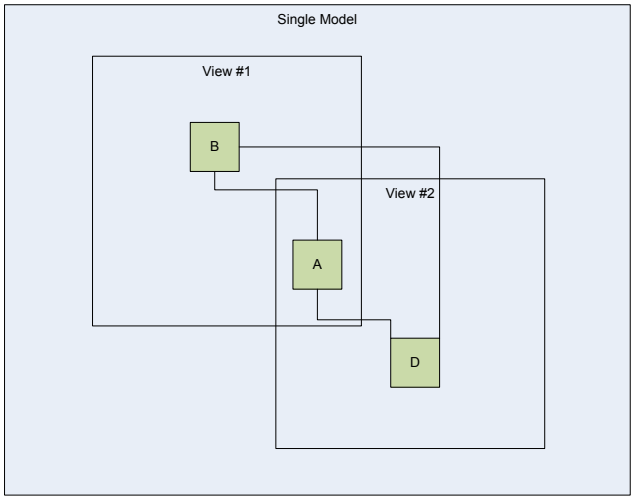
In our analysis we found that these transformations could be of common ontological types; super-sub-type, types-instance and represented by. What is unclear is how heterogeneous the types of transformations can

be between one cell and another. One would expect a high degree of homogeneity. This point underlies comments below on representation levels.

The original ZF greatly simplified its integration framework by constraining all the links to neighbouring cells, the ones in adjoin rows and columns. So each cell is only linked to a maximum of four other cells – rather than all 36 cells. In ZF 2.0, the constraint is relaxed for rows – cells in a row can be linked to any other cell in the row.

We suspect that in practice this may be too constraining in some cases. The transformation from Row 2 to Row 3 (and maybe Row 3 to Row 4) for Columns 1 and 2 may overlap. The issue is well-known in object modelling. It is often the case that a business event/process (Business Transform) is modelled as a data entity (System Entity), the classical case is a transaction. This issue is discussed in detail in Ch 2, Section 4.1 'Imposing the data-process distinction onto things in the business' Business Objects (Partridge 1996).

In the views over common model approach, one builds and extends a common model as one builds the EA. There is no constraint on the model footprint of one view overlapping another. Where an object shown in one view relates to an object in another view, this relationship is documented in the model. This is shown in the diagram below.



This approach avoids the need for an integration infrastructure and its transformations, and has the potential to greatly simplify the architecture.

Meta-Architectural – Anti-Patterns

In some architectural circles the idea of patterns (re-usable feature of architectures) has taken hold. From this has developed the idea of anti-patterns – mistakes that get needlessly repeated. One candidate anti-pattern appeared in the analysis – a confusing way of modelling levels of representing.

Representing 'Levels of Representing'

The challenge for EAFs is that different architectures will cover different (often multiple) levels of representing. M3 adopts a simple approach of directly modelling the levels. However, what complicates the matter is that as one moves through the ZF (or similar) perspectives) the transform will shift levels in different ways depending upon the architecture – and reflecting the concerns of that cell. This is illustrated in the simple example below, with two architectures which start at the same level but one jumps a level whereas the other does not. Furthermore, in ZF the changes in level are isolated to particular columns and not propagated across the whole row.

	IT Architecture #1	Level	Physical Architecture #1	Level
Business Model	Person Icon	1	Organisation Unit Icon	1
Logical Model	Person Entity Icon	2	Unit Icon	1
Physical Model	Person File Icon	2	Battalion Icon	1
Implementation	Person File	0	Battalion	0

The issue for any mapping is that some reification row to row transformations will preserve the mapping to M3, whereas others will involve a jump in representation level. The issue is further complicated by a well-documented tendency (often called use-mention confusion) to ignore that this jump has taken place.

Use-mention distinction

What the jump in between perspectives involves something similar to a use-mention distinction. The classic example in natural language is:

> BOSTON has six letters, and
> BOSTON is a city.

In the first case the word Boston is being mentioned and in the second case it is being used to represent the city (see also http://en.wikipedia.org/wiki/Use–mention_distinction). When there is a need for a clear logical structure, it is recognised that it is absolutely vital to clearly distinguish between the two. For natural language this can be done using a system of quotation marks (though this needs to have a clear foundation – see http://plato.stanford.edu/entries/quotation/#2.2).

How this works when modelling IT systems is that a model of the business is re-interpreted as a model of system that models the business. To give you some indication of its importance, Zachman spends some time in his original paper discussing this issue. The points are key, so much of his discussion is included in the extract below:

> For example, when owners (users) specify an entity such as "employee," what they have in mind would be real beings, that is, flesh and blood employees who work for the business. That meaning of "employee" is entirely different from the one used in an information systems model (the designer's perspective), in which "employee" would refer to a record on a machine, which also happens to be called "employee," however conceptually and entirely different it is. (This data entity, as opposed to business entity, would be found in the cell directly below.)
>
> ...
>
> Finding good "real-life" examples which crisply illustrate each of the architectural representations is difficult. There are two reasons for this difficulty. First, when the real-life representations were being developed, no framework existed to clearly define and differentiate one representation from the others. Therefore many real-life illustrations are a mixture of representations, both conceptually (e.g. business entities and data entities get mixed together) and physically (e.g. entities and inputs/outputs, that is, user views from the process description column of Figure 2, get mixed together). Second, real-life examples are hard to understand because it is not always clear what model, or cell, the author had in mind when developing the representation.
>
> An illustration of this difficulty is found in Figure 3. It is clear that this model is describing data and not process, but the question is, did the author have in mind a description of a business or a description of an information system? ...

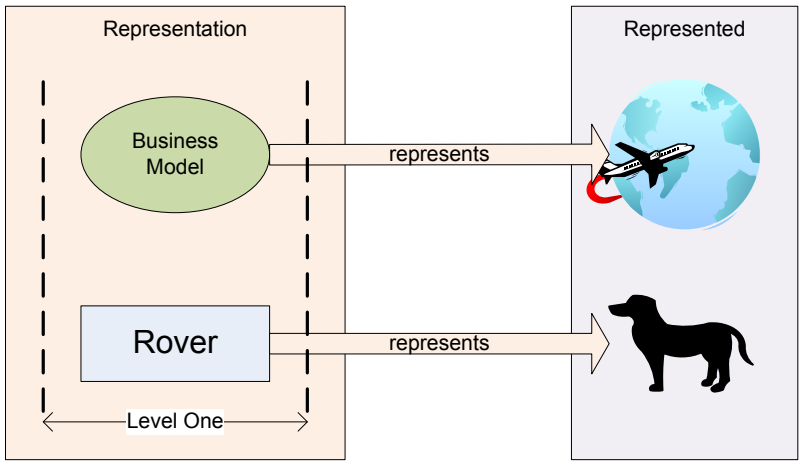
p. 464 - A framework for information systems architecture. IBM SYSTEMS JOURNAL, VOL 26. NO 3. 1987. © IBM 1987.

Zachman notes that at time (1987) representation-represented confusion was endemic ("many real-life illustrations are a mixture of representations") and that an EAF needs to differentiate between them ("no framework existed to clearly define and differentiate one representation from the others"). The situation is only marginally better today.

One of the challenges with this issue is the lack of recognition and clear understanding of the issue. To aid understanding, a brief outline is given below.

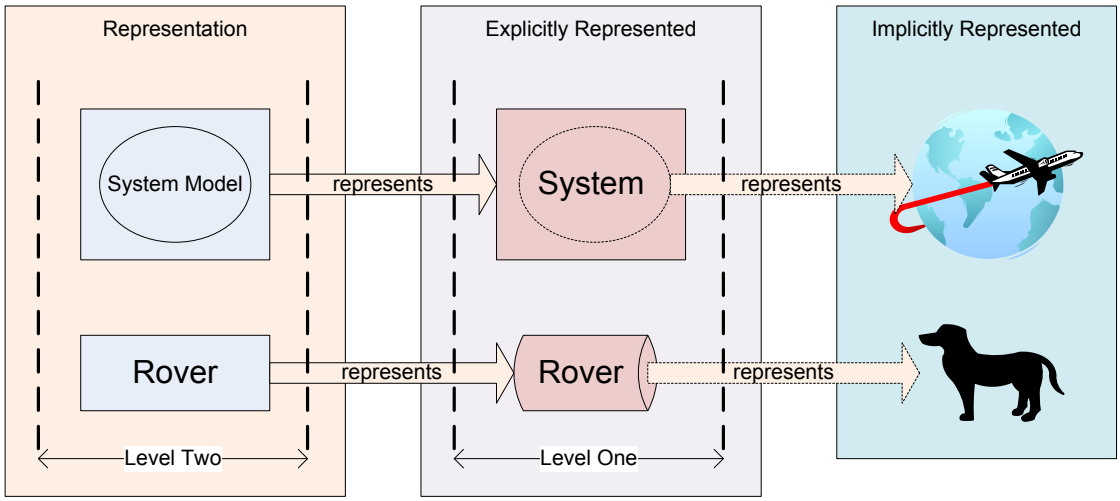
A simple (level one) model represents objects with icons – there is no question of what the icon refers to. In the case the icon 'Rover' refers to the dog called Rover.

Level One Perspective



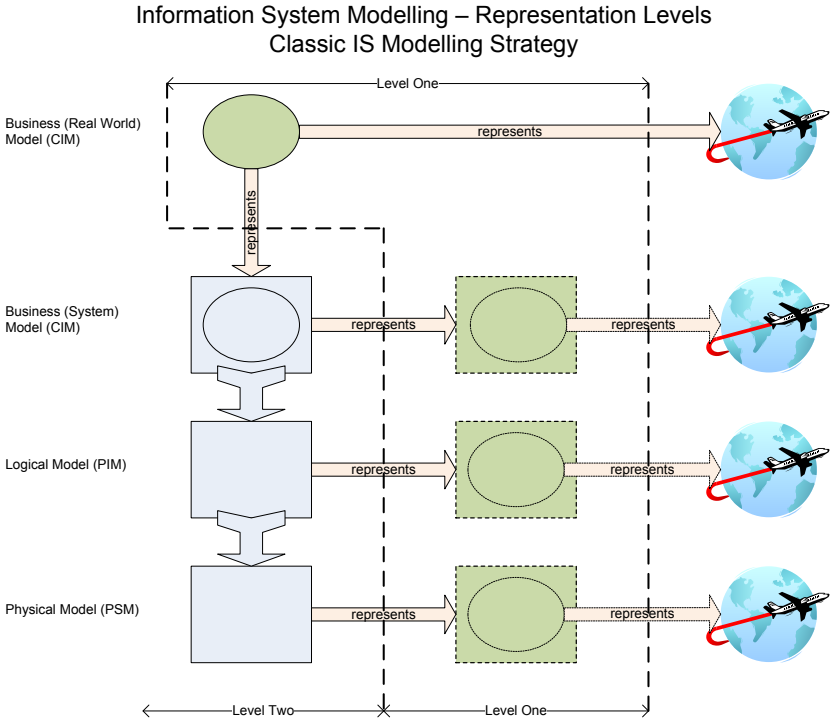
The situation gets more complicated when modelling information systems. From one perspective, the representing pattern repeats itself. In this example, there is a 'Rover' system model icon which represents the Rover data item. But this is a level two perspective, as the Rover data item represents the dog Rover.

Level Two Perspective



Each new model type added to a modelling framework, adds an economic burden, hence there is a pressure to reduce the number of model types. IT practitioners have evolved a practical technique that takes advantage of the characteristics of levels of representing to reduce the modelling burden, but at the

cost of introducing a kind of equivocation between levels (a systematic transition from use to mention) which is now well entrenched.

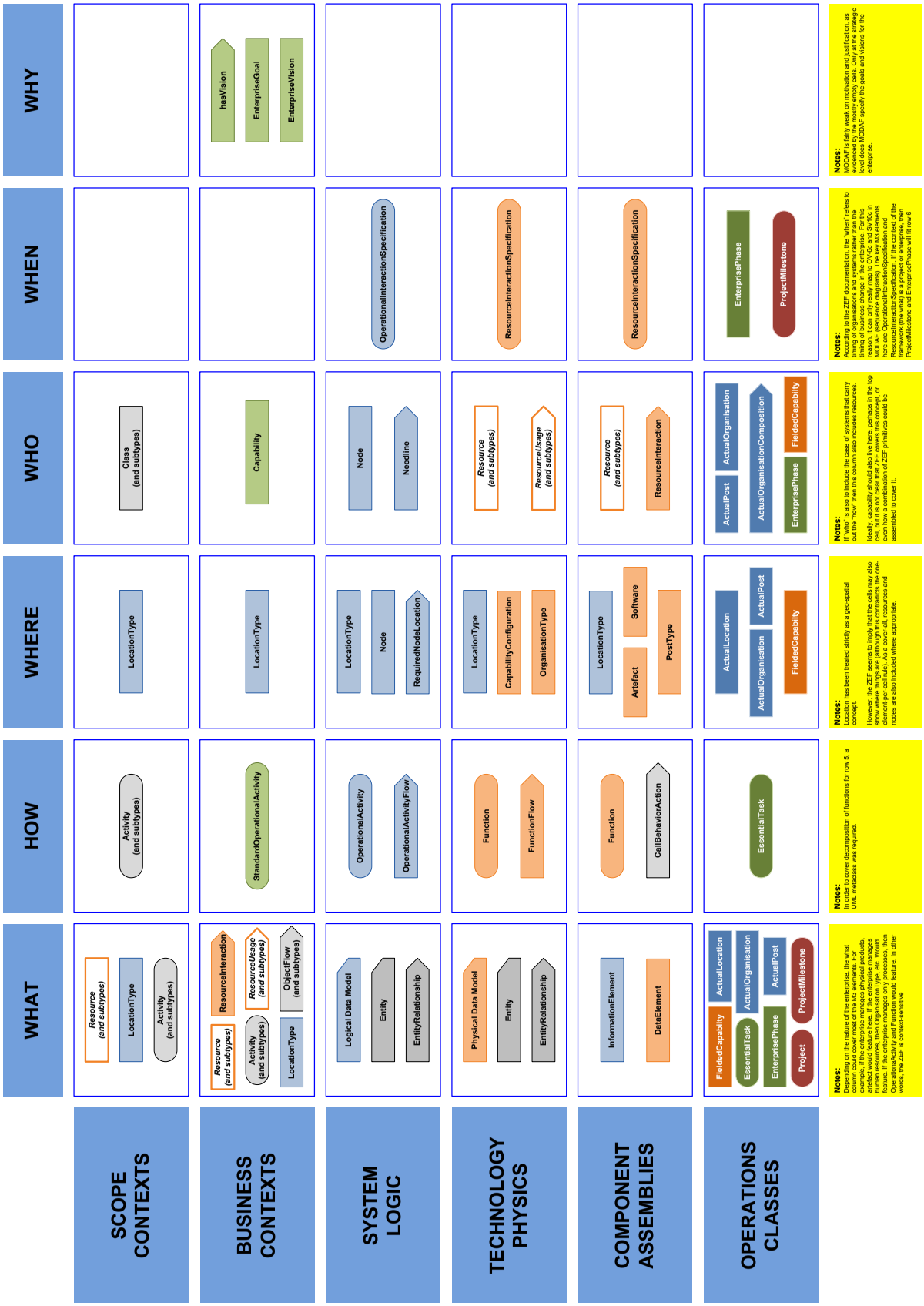


Classic Modelling Strategy hides the Representation Levels

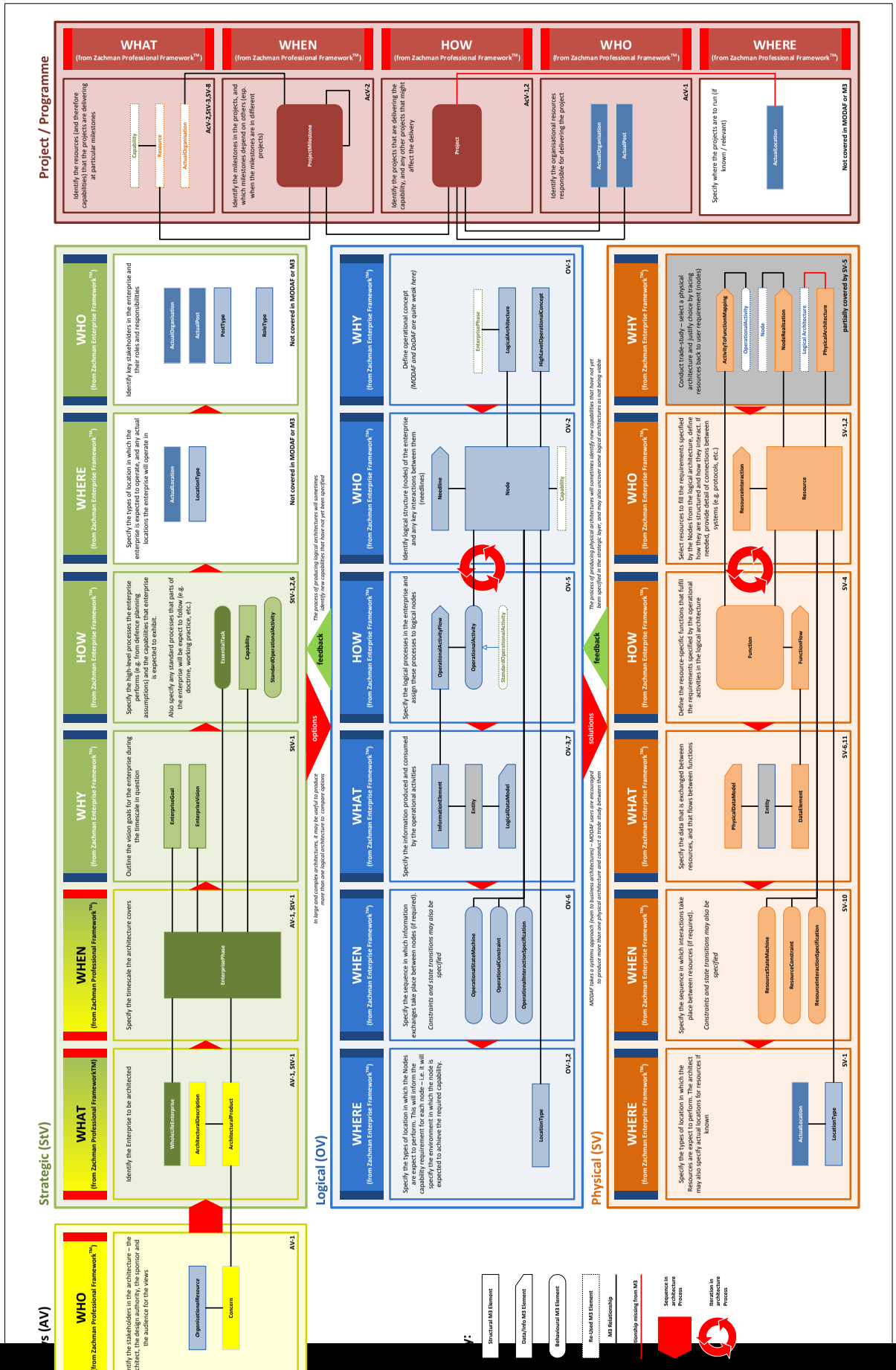
At the business modelling stage, a model is produced that is a level one model of the business. Then at the system modelling stage, the model is re-interpreted (initially unchanged) as a model of the system.

Often this transition is not clearly marked and so it is unclear whether the model is of the system or the business. ZF and MODAF deal with this in different ways. In MODAF this is clearly marked in the M3 has a model element for Information. In ZF this is clearly marked as the two models belong to different cells. What is less clear in ZF is whether the jump has taken place in the transformation between rows 2 and three. If the nature of the transformation was clearly marked, then it would be clear.

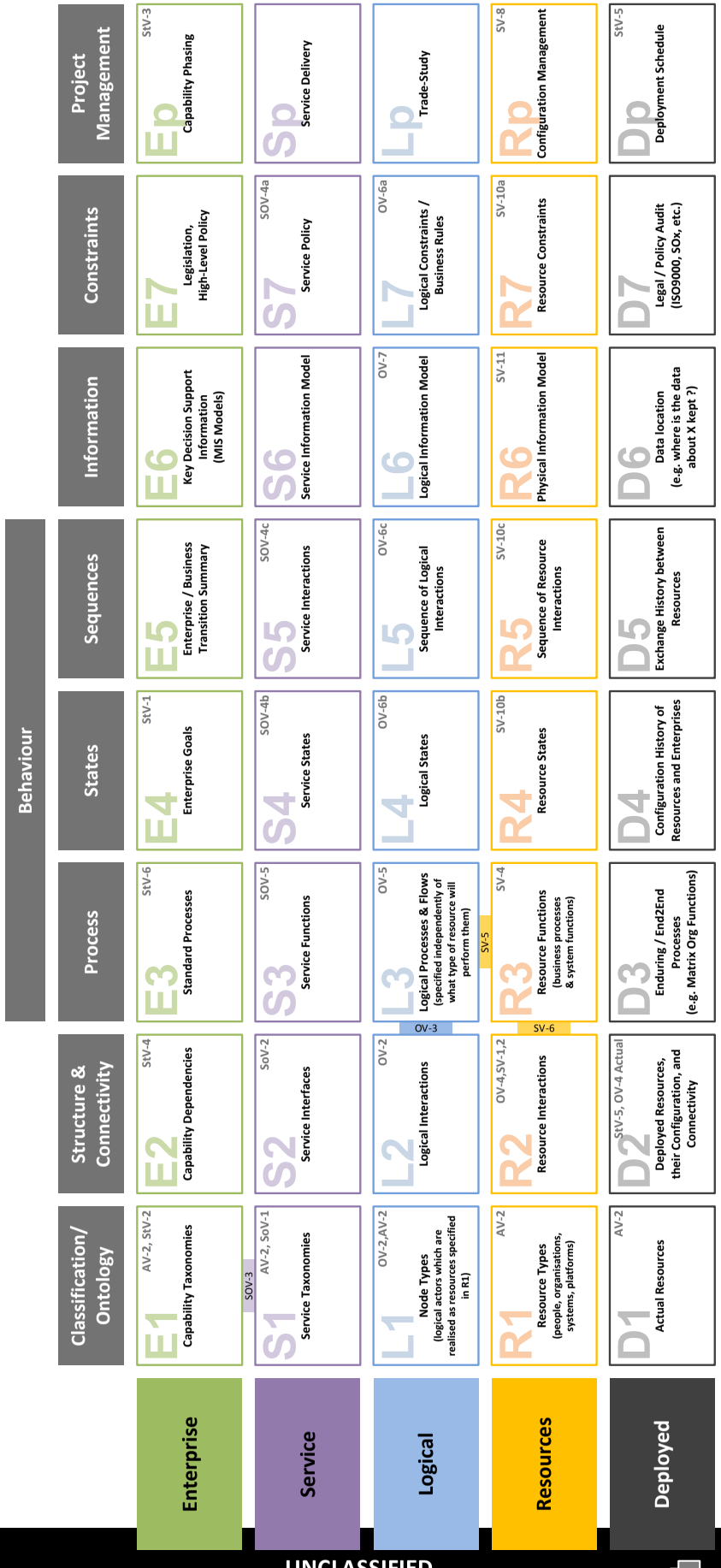
Appendix A – M3-Zachman Mapping



Appendix B – Outline Method for using Zachman Interrogatives with MODAF



Appendix C – "Period Table" Arrangement of MODAF Views



Appendix D - Zachman comments on ontology

Zachman's has a mature commitment to the use of ontology in EA that is clear and unequivocal – as shown by the selection of comments below from Z101.

Z101	Extract
p. 15	Ontology The Zachman Framework schema technically is an ontology - a theory of the existence of a structured set of essential components of an object for which explicit expression is necessary (is mandatory?) for designing, operating and changing the object (the object being an Enterprise, a department, a value chain, a "sliver," a solution, a project, an airplane, a building, a bathtub or whatever or whatever). The Zachman Framework is NOT a methodology for creating the implementation (an instantiation) of the object (i.e. the Framework is an ontology, not a methodology). A Framework is a STRUCTURE. (A Structure DEFINES something.) An Ontology is a theory of existence - what IS An Ontology IS a Structure. A Methodology is a PROCESS. (A Process TRANSFORMS something.) A Structure IS NOT A Process A Process IS NOT a Structure
p. 16	Ontology This is a Structure, an ontological structure ... a fixed, structured set of elemental components that exist of which any and every compound must be composed.
p. 17	Chemistry - A Science A Process TRANSFORMS something. ... A Process based on an ONTOLOGICAL structure will be repeatable and predictable - A SCIENCE. Process A Process with no ontological structure is ad hoc, fixed and dependent on practitioner skills. This is NOT a science. It is ALCHEMY, a "practice"
p. 18	Ontology vs Process An Ontology IS NOT A Process and a Process IS NOT an Ontology. The Periodic Table Metaphor A reasonable metaphor for the Framework is the Periodic Table. The Periodic Table is an ontology ... a schema ... a normalized schema ... one element goes in one and only one cell. The Periodic Table doesn't do anything. It reflects nature. The Periodic Table (an ontology) is used by Chemists (practitioners) to define a Process (a methodology) for producing compounds (results, implementations, composites). If an alchemist uses the Periodic Table to define the process, the process can be dynamically defined (or redefined) and will be repeatable and produce predictable results ... and the alchemist will become a Chemist. On the other hand, if the alchemist ignores the Periodic Table, they can define a process (a methodology) that will produce results, point-in-time solutions, based on their own skills

	and experience (heuristics). The process (methodology) will be fixed (not changeable) and the alchemist will forever remain an alchemist. Practitioners (methodologists) are constrained by time and results. Theoreticians (scientists) are constrained by natural laws and integrity.
p. 20	Ontology vs Methodology The Framework does not imply anything about: a. whether you do Architecture or whether you simply build systems (that is, whether you build Primitive Models, the single variable intersections between the Abstractions and the Perspectives or whether you build multi-variable, composite models made up of components of several Primitive Models) b. how you do Architecture (top-down, bottom-up, left to right, right to left, where to start, etc., etc.) c. the long term/short term trade-off relative to instantiating the expression of the components of the object (i.e. what is formalized in the short term for implementation purposes versus what is engineered for long term reuse). d. how much flexibility you want for producing composite models (Enterprise implementations) from your Enterprise Architecture (primitive models), that is, how constrained (little flexibility) or unconstrained (much flexibility) you make the horizontal, integrative relationships between the Cell components across the Rows and the vertical, transformational relationships of the Cell components down the Columns. (These are significant, identifiable methodological choices ... not prescriptions of the Framework.
p. 56	It is my opinion that the ontology, the "Zachman Framework" primitive models, are infinitely helpful in explaining the "physics" of any intention.”
p. 102	Methodology Conclusions There is not one way to do Enterprise Architecture. Therefore, there is not one Enterprise Architecture tool or methodology. I doubt there will EVER be one Enterprise Architecture methodology or tool. In fact, I doubt there will ever be one method or tool that will support all of the primitive models identified by the Zachman Framework and certainly not all of the potential composite models or analyses that could be implemented from the primitives. Having said that, I do believe there is one architectural metamodel (ontology) that defines Architecture, specifically Enterprise Architecture, that would accommodate the total knowledgebase for any Enterprise. It would be extremely significant if the tools or methods accommodated that ontology. There is a lot of room for creativity and specialization.
p. 103	Methodology Conclusions (cont.) Historically, there has been little or no demand for methods or tools that support primitive models (Architecture) ... only composite models (implementations). The magic is not in the tool - it is in the engineer. If the engineer understands Framework Ontology, many methods/tools could prove helpful.