A Model-based Approach to Architectural Frameworks

Simon Perry, Principal Consultant, Atego (simon.perry@atego.com)

Abstract

The concept of an architecture is now seen as essential to any systems engineering undertaking and is a core element of any model-based systems engineering approach. An architecture should, as advocated by industry best practice, be based on an architectural framework (AF). An AF defines a number of allowed viewpoints of a system that any architecture based on the AF can contain, together with consistency rules between the viewpoints. Many AFs exist and many organisations will adopt one of these AFs for the development of their system architecture. Unfortunately, this is often done without first assessing the stakeholder concerns that the architecture is to address against the viewpoints defined in the chosen AF, resulting in the adoption of an unsuitable AF that unnecessarily constrains or twists the resulting architecture. If the stakeholder concerns are considered, then the conclusion may be that a bespoke AF is needed. This paper discusses a model-based approach to the definition of AFs, the Framework for Architectural Frameworks (FAF), based on the concept of an ontology that defines concepts and the relationships between them, defined viewpoints that use the concepts from the ontology and that are organised into a framework. An example of the use of the FAF is given.

1. Introduction - What Are AFs and Why Are They Important?

Architectural design is seen as an essential part of systems engineering and is one of the key technical processes in ISO15288:2008 "Systems and software engineering – System life cycle processes" [ISO15288:2008]. Architectures are now seen as essential to any systems engineering undertaking and are a core element of any model-based systems engineering (MBSE) approach.

An architecture must cover three key areas: [Stevens et al 1998]

- System structure, defining the major components of the system, their organisation and structure.
- System behaviour, defining the "dynamic response of the system to events, *providing a basis for reasoning about the system*." (our italics)
- System layout, defining the physical layout and packaging of the system.

Unfortunately, architectures are often developed that describe only structure. Another key point regarding architectures is that they must be seen and treated as evolving artefacts that will change through time and require maintenance. Too often they are created, at great expense, and then forgotten.

A key enabler for the production of an architectural design is the *architectural framework (AF).* Many, such as [Dickerson & Mavris 2009], consider AFs to be an essential tool; architectures should always be based on an architectural framework.

An AF defines a number of *viewpoints* of a system that any architecture based on the AF can contain, together with consistency *rules* between the viewpoints. These viewpoints are often grouped into *perspectives* that relate viewpoints that address the same architectural *concerns*. When an architecture is produced that conforms to an AF, then the architecture is composed of a number of *views*, each of which is an instance of a viewpoint. ISO 42010 'Systems and software engineering – Architecture description' [ISO42010:2011] provides a definition of the terms associated with architectural frameworks.

The use of an AF helps to ensure a consistent approach to the production of an architecture. Engineers know what is expected of them because the views that can be produced are defined. This helps ensure that the architecture is fit for purpose by ensuring that all the concerns that the architecture must address are covered.

1.1 The Problem with the Existing Approach to AFs

Many architectural frameworks exits, such as MODAF [MODAF 2014], DoDAF, NAF, TRAK, Zachman [Zachman 2008] etc. However, many architectural frameworks are usually created for a specific purpose (such as military acquisition or enterprise architecture) and as such the viewpoints that they define are those that are deemed necessary to meet the requirements of that framework. This means that not all the viewpoints in a given framework may be relevant or that a framework may be missing needed viewpoints.

Many organisations will adopt one of these AFs for the development of their system architecture. Unfortunately, this is often done without first assessing the stakeholder concerns that the architecture is to address against the viewpoints defined in the chosen AF. This often results in the adoption of an unsuitable AF that unnecessarily constrains or twists the resulting architecture.

For example, consider MODAF. It was created to assist the MOD in acquisition of systems but is often used by suppliers when creating architectures internally (i.e. for their own internal system development work), without any consideration as to whether MODAF defines viewpoints that address the concerns that their architecture must capture. This results in architectures that are very good examples of MODAF architectures but which are not fit for purpose as system engineering architectures. If the stakeholder concerns are considered by an organisation, then the conclusion may be that a bespoke AF is needed. This paper presents a model-based approach to the definition of architectural frameworks.

2. The Framework for Architectural Frameworks (FAF)

The Framework for Architectural Frameworks (FAF) was developed to improve the definition of architectural frameworks by forcing anyone defining an AF to consider the following six questions:

1. What is the purpose of the AF?
2. What domain concepts must the AF support?
3. What viewpoints are required?

4. What is the purpose of each viewpoint?
5. What is the definition of each viewpoint in terms of the identified domain concepts?
6. What rules constrain the use of the AF?

The FAF addresses the six questions through an MBSE approach that is based around the ideas of ontology, viewpoints and framework:

- Ontology - Defines concepts and relationships between them
- Viewpoints and Framework - Defines viewpoints organised into a framework. Viewpoints can only use concepts from the ontology

The FAF consists of an ontology, six viewpoints and supporting processes and is itself defined using the FAF i.e. the FAF is defined in terms of the six viewpoints of the FAF (!). This paper, in the following section, describes the ontology and the six viewpoints. Section 3 gives an example of its use. The supporting processes are not discussed in this paper; the reader is directed to 'SysML for Systems Engineering: 2nd edition: A model-based approach' [Holt & Perry 2013] which gives a complete definition of the FAF, its supporting processes and many examples of its use.

It should also be noted at this point that the FAF is intended to be used in the definition of any size of architectural framework, from a complete AF, through frameworks that address a particular topic (such as requirements) through to so-called *enabling patterns* - specific constructs of modelling elements whose combination and subsequent use enables a number of systems engineering applications. An example of an enabling pattern would be one used for the definition of interfaces or one used to ensure traceability throughout a model of a system

The FAF Ontology

The FAF is built around an ontology that defines a number of concepts, and their relationships, relating to architectures and architectural frameworks. The ontology for the FAF is shown in Figure 1 and is based on [ISO42010:2011].

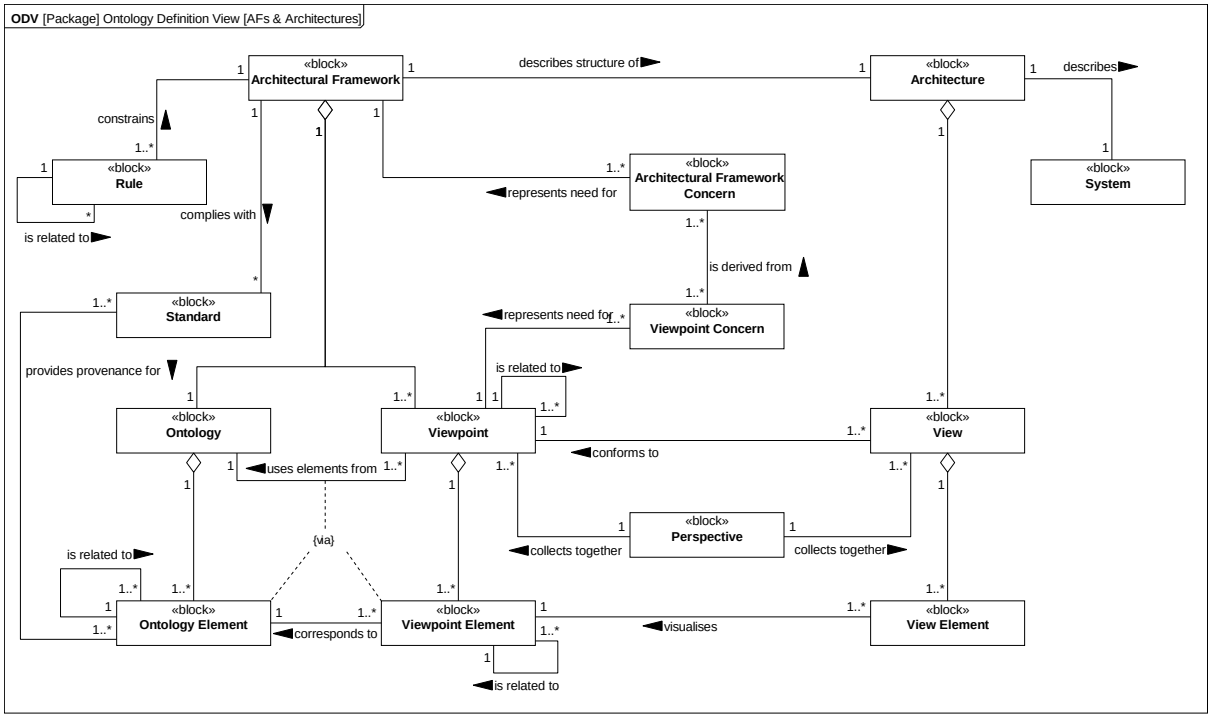


Figure 1 - Ontology for Architectures and Architectural Frameworks

The concepts shown on Figure 1 are defined as follows:

- *Architectural Framework* - a defined set of one or more *Viewpoints* and an *Ontology*. The Architectural Framework is used to structure an *Architecture* from the point of view of a specific industry, stakeholder role set, or organisation. The Architectural Framework is defined so that it meets the needs (requirements) defined by one or more *Architectural Framework Concerns*. An Architectural Framework is created so that it complies with zero or more *Standards*.
- *Architectural Framework Concern* - defines a need that an Architectural Framework has to address.
- *Ontology* - an element of an Architectural Framework that defines all the concepts and terms (one or more *Ontology Element*) that relate to any Architecture structured according to the Architectural Framework.
- *Ontology Element* - the concepts that make up an Ontology. Each Ontology Element can be related to each other and is used in the definition of each *Viewpoint* (through the corresponding *Viewpoint Elements* that makes up a Viewpoint). The provenance for each Ontology Element is provided by one or more Standard.
- *Viewpoint* - a definition of the structure and content of a *View*. The content and structure of a Viewpoint uses the concepts and terms from the Ontology via one or more *Viewpoint Elements* that make up the Viewpoint. Each Viewpoint is defined so that it meets the needs defined by one or more *Viewpoint Concern*.
- *Viewpoint Concern* - defines a need that a Viewpoint has to address.
- *Viewpoint Element* - the elements that make up a Viewpoint. Each Viewpoint Element must correspond to an Ontology Element from the Ontology that is part of the Architectural Framework.

- *Architecture* - a description of a *System*, made up of one or more *Views*. One or more related View can be collected together into a *Perspective*.
- *View* - the visualisation of part of the Architecture of a System that conforms to the structure and content defined in a Viewpoint. A View is made up of one or more *View Elements*.
- *View Element* - the elements that make up a View. Each View Element visualises a Viewpoint Element that makes up the Viewpoint to which the View, on which the View Element appears, conforms.
- *Perspective* - a collection of one or more Views (and hence also one or more defining Viewpoints) that are related by their purpose. That is, one or more Views which address the same architectural needs, rather than being related in some other way, such as by mode of visualisation, for example.
- *Rule* - a construct that constrains the Architectural Framework (and hence the resulting Architecture) in some way, for example by defining one or more Viewpoints that are required as a minimum.
- *System* - set of interacting elements organised to satisfy one or more needs. The artefact being engineered that the Architecture describes.

A note about naming style: from this point forwards, any reference to a concept from the FAF Ontology will be capitalised to indicate that the term is being used in the sense defined on the Ontology, rather than in its everyday usage.

It is important to note here that an Architecture is simply considered to be a description of a System, represented by a number of Views that are created according to a number of predefined Viewpoints from a given Architectural Framework.

The FAF Viewpoints

The FAF defines six Viewpoints, as shown in Figure 2.

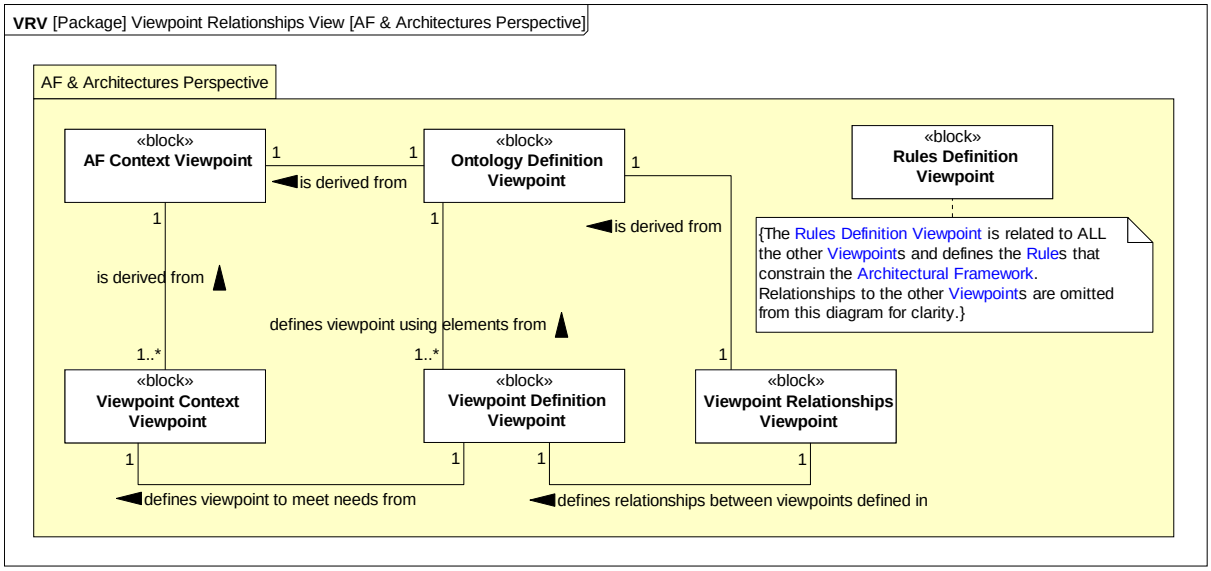


Figure 2 - The Six FAF Viewpoints

Each of the six Viewpoints is designed to address one of the six questions presented above. Each Viewpoint is briefly described in this section. Examples are given in Section 3.

The *AF Context Viewpoint* (AFCV) addresses the question *'What is the purpose of the AF?'* It defines the context for the Architectural Framework. That is, it represents the Architectural Framework Concerns in context, establishing *why* the Architectural Framework is needed.

The *Ontology Definition Viewpoint* (ODV) addresses the question *'What domain concepts must the AF support?'* It defines the Ontology for the Architectural Framework. It is derived from the AF Context Viewpoint and defines the concepts that can appear on a Viewpoint.

The *Viewpoint Relationships Viewpoint* (VRV) addresses the question *'What viewpoints are required?'* It shows the relationships between the Viewpoints that make up an Architectural Framework and groups them into Perspectives. It is derived from the Ontology Definition Viewpoint.

The *Viewpoint Context Viewpoint* (VCV) addresses the question *'What is the purpose of each viewpoint?'* It defines the Context for a particular Viewpoint. That is, it represents the Viewpoint Concerns in context for a particular Viewpoint, establishing why the Viewpoint is needed. It is derived from the AF Context Viewpoint.

The *Viewpoint Definition Viewpoint* (VDV) addresses the question *'What is the definition of each viewpoint in terms of the identified domain concepts?'* It defines a particular Viewpoint, showing the Viewpoint Elements (and hence the Ontology Elements) that appear on the Viewpoint.

The *Rules Definition Viewpoint* (RDV) addresses the question *'What rules constrain the use of the AF?'* It defines the various Rules that constrain the Architectural Framework.

The six Viewpoints are collected into a single Perspective, the *Architectural Framework Perspective*, as shown by the enclosing *package* in Figure 2.

Two points are worth noting here. First, that *each* Viewpoint that is being defined in an Architectural Framework will have its own VCV (establishing its purpose) *and* VDV (defining its allowed content). Second, the use of singular descriptions for the other Viewpoints does *not* imply that there is only a single instance (a View) of each in the AF being defined. Thus, for example, a number of VRVs may be required, perhaps showing one Perspective per VRV.

3. The FAF in Use – an example of use for the Traceability Pattern

This section presents a small example of the use of the FAF. In this case, the FAF is used to define an enabling pattern rather than a full AF. It is not the intention here to describe the details of the pattern, but simply to give examples of the use of each of the six FAF Viewpoints. For this reason, not all the Views of the pattern are shown. For example, since the pattern contains four Viewpoints (see Figure 6), the full definition contains four VCVs and four VDVs, one per Viewpoint. For illustration, only one is shown. The enabling pattern used is the Traceability Pattern, one of a number of enabling patterns defined by the author.

Two points are worth noting here. Firstly, the diagrams below are all Views. They are instances (Views) of FAF Viewpoints that themselves define Viewpoints. This is how the FAF is used. Every time the FAF is used to define an AF or a pattern, Views that conform to the six FAF Viewpoints are created. These Views define the Viewpoints of the AF or pattern being defined. Secondly, remember that these Views are defining the Traceability Pattern by defining its Viewpoints. When the Traceability Pattern is used, then Views that conform to these defined Viewpoints are produced. Examples of such Views from the Traceability Pattern in use are not shown or discussed here.

Figure 3 shows the AFCV for the Traceability Pattern. It captures the needs (in the terminology defined in Figure 1, the Architectural Framework Concerns) that the pattern is designed to address. When using UML or SysML as the modelling language, the AFCV can be represented using a use case diagram, as has been done here.

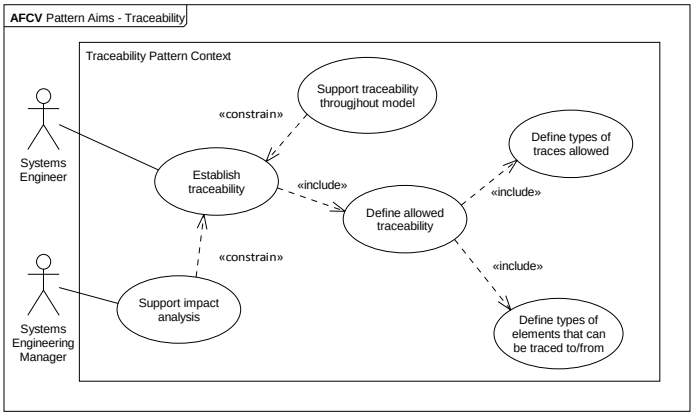


Figure 3 - Example AFCV

With the needs defined, it is necessary to define the concepts and relationships between them that apply to the AF or pattern. In this case, concepts relating to traceability. These are captured on the ODV, as shown in Figure 4. When using UML or SysML as the modelling language, the ODV can be represented using a class or block definition diagram, as has been done here.

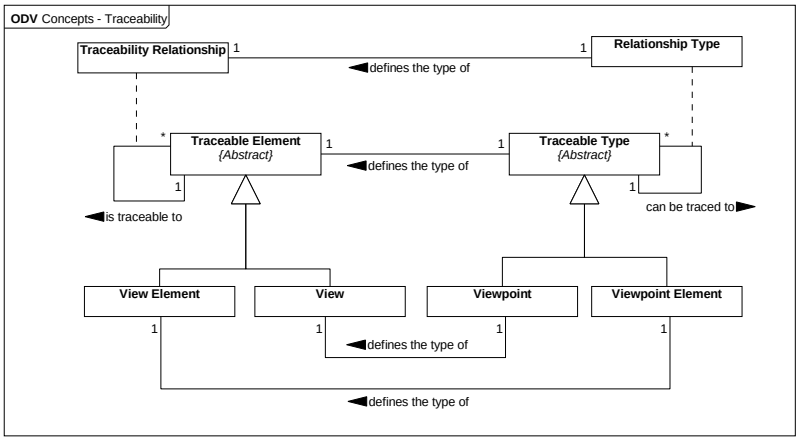


Figure 4 - Example ODV

Remember, the purpose of the ODV is to define all the concepts and relationships that are relevant to the framework or pattern being defined. Only elements appearing on the ODV can be used in the definition of the Viewpoints for the framework or pattern.

The ODV is a useful guide to the possible Viewpoints that the framework or patttern is to

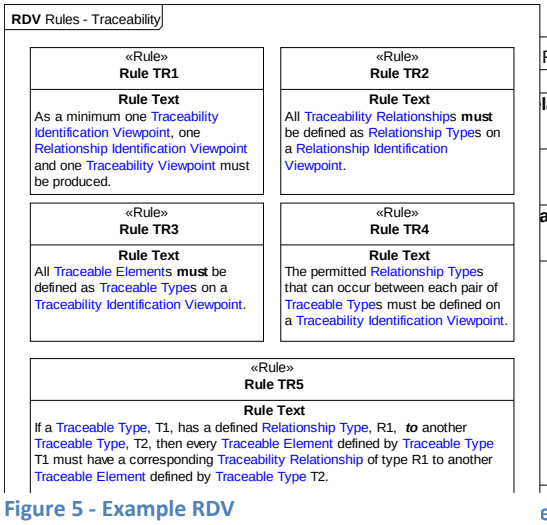
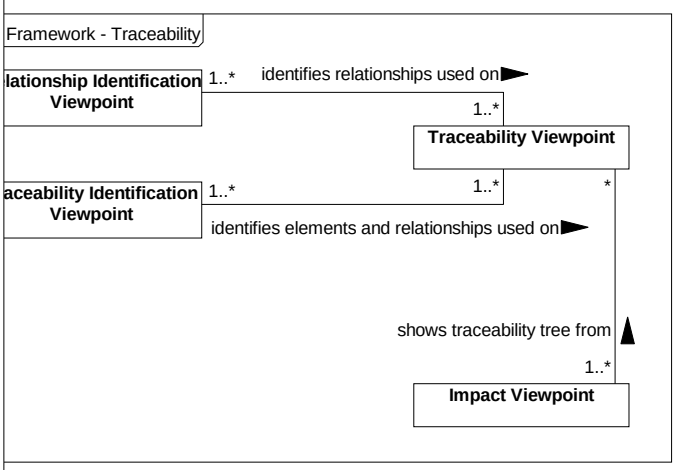


Figure 5 - Example RDV



e 6 - Example VRV

contain. Once these Viewpoints have been identified, they should be captured on a VRV. This identifies the Viewpoints, establishes the main relationships between them and, if necessary, allows them to be grouped into Perspectives. The VRV for the Traceability Pattern is shown in Figure 6. When using UML or SysML as the modelling language, the VRV can be represented using a class or block definition diagram, as has been done here.

Most frameworks and patterns will have Rules that constrain their use. There are typically three types of Rules: Rules that define the minimum set of Views that have to be present in anything based on the framework or pattern; Rules that define consistency checks between Viewpoints and the Views based on them; Rules that define consistency checks within Viewpoints and the Views based on them. Such rules are captured in the RDV, as shown in Figure 5. When using UML or SysML as the modelling language, the RDV can be represented using a class or block definition diagram, as has been done here. Note the use of the «Rule» stereotype and associated 'Rule Text' tag to mark these elements as Rules. Of course, these Rules (and hence the RDV) can just as easily be represented using text in a simple table.

Each of the Viewpoints are then defined through a VCV, which captures the needs (Viewpoint Concerns) that the Viewpoint is intended to address, and a VDV which defines the Viewpoint in terms of the Ontology Elements that can appear on it.

An example VCV is given in Figure 7. When using UML or SysML as the modelling language, the VCV can be represented using a use case diagram, as has been done here.

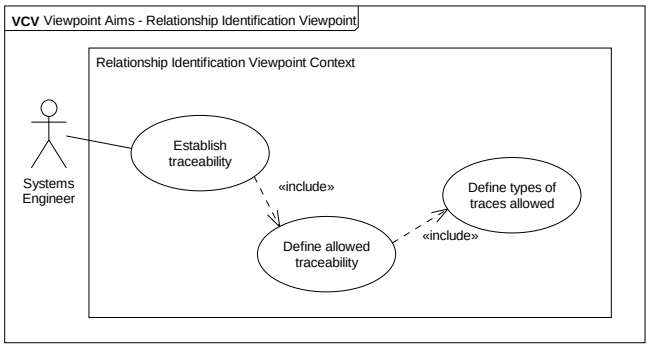


Figure 8 - Example VCV

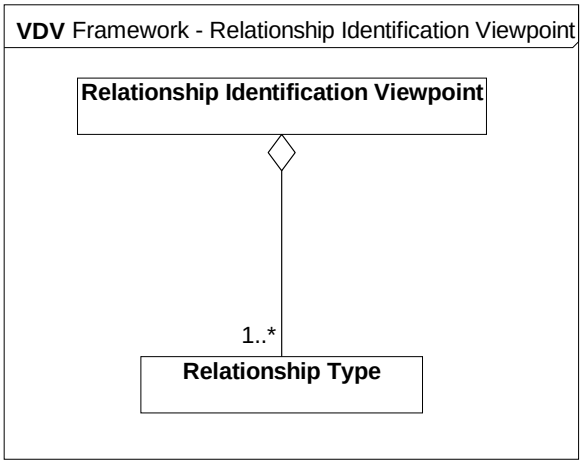


Figure 7 - Example VDV

Note that the Viewpoint Concerns shown on a VCV will typically be a subset of the Architectural Framework Concerns shown on the AFCV.

An example VDV is given in Figure 8. When using UML or SysML as the modelling language, the VDV can be represented using a class or block definition diagram, as has been done here. Remember that a VDV can only use Ontology Elements from the ODV. If a concept is needed that doesn't appear on the ODV then the ODV must be updated to include that concept. Conversely, once all the Viewpoints are defined, then every Ontology Element must appear on at least one VDV. If there are Ontology Elements that are not used, then either there are missing Viewpoints or the elements can be removed from the Ontology.

Finally, although the example Views have been presented here in a logical order, their production typically does not proceed in a linear fashion. Rather, the modeller must be prepared to iterate across a number of the Views until all are complete.

The FAF in Use – Other Examples

The FAF is currently being used in a number of organisations and for a number of purposes. These include:

- Automotive – The FAF is being used for the definition of a bespoke AF used for the definition of architectures for Electric Power Steering (EPS) systems. It is being used to both better understand existing architectures and to define a generic future EPS architecture.
- Home entertainment – The FAF is being used for the definition of a bespoke AF for the media streaming architecture for high-end home audio-visual systems (by B&O). It is currently being used in the engineering of future B&O home entertainment systems.
- Fault modelling – The FAF has been used for the definition of a fault modelling framework (the FMAF) that focuses on support for fault modelling as part of the architectural modelling of Systems and SoS. The FMAF has been created by Newcastle University as part of the

COMPASS project. (See http://www.compass-research.eu/Project/Deliverables/D242.pdf). The FMAF includes support for

  - Definition of faults, errors and failures
  - Identification of the causal chains of dependability threats (faults, errors and failures)
  - Identification of CSs (and the connections and interfaces between them) needed to tolerate faults
  - Identification of erroneous behaviour/recovery scenarios and processes
  - Behaviour description of processes in the presence of faults and recovery processes
- Enabling patterns – The FAF is being used in the definition of additional systems engineering enabling patterns both by the author and by the INCOSE UK MBSE WG.

An example MBSE AF that is defined using the FAF is outlined in [Holt & Perry 2013].

4. Conclusions

When creating a System Architecture, the use of an Architectural Framework can help ensure consistency of approach and coverage of the correct architectural concerns. However, the choice of Architectural Framework *must* be made to ensure that the concerns can be addressed by an Architecture based on the Architectural Framework. This is not always the case, because organisations often adopt an Architectural Framework without understanding its intent, suitability and coverage of concerns. Such a choice may, therefore, require the creation of a bespoke AF.

An MBSE approach to the definition of an Architectural Framework allows the Architectural Framework to be created using the same tools and techniques as are used in an MBSE approach to the definition of the System. The Framework for Architectural Frameworks (FAF) provides such an MBSE approach to the definition of Architectural Frameworks.

The FAF has been used successfully by the author and other organisations in the definition of Architectural Frameworks in a range of application domains including automotive, high-end home entertainment & fault-modelling. In addition, the FAF is being used both by the author and by the INCOSE UK MBSE Working Group in ongoing work on the definition of enabling patterns for system engineering.

The FAF is fully defined and described in [Holt & Perry 2013], which also presents a process that can be used with the FAF for the definition of an Architectural Framework, along with an extended example that demonstrates the use of the FAF.

5. References & Further Information

The key references used in this work are:

[Dickerson & Mavris 2009] Dickerson, C.E. & Mavris, D.N. 'Architecture and Principles of Systems Engineering'. CRC Press, 2009

[Holt & Perry 2013] Holt, J. & Perry, S. 'SysML for Systems Engineering: 2nd edition: A model-based approach'. IET Publishing, 2013

[ISO15288:2008] ISO/IEC. 'ISO/IEC 15288:2008 Systems and software engineering – System life cycle processes'. 2nd Edn, International Organisation for Standardisation; 2008

[ISO42010:2011] ISO/IEC. 'ISO/IEC 42010:2011 Systems and software engineering — Architecture description'. International Organisation for Standardisation; 2011

[MODAF 2014] The Ministry of Defence Architectural Framework, 2010: https://www.gov.uk/mod-architecture-framework (Accessed September 2014)

[Stevens et al 1998] Stevens, R., Brook, P., Jackson, K. & Arnold, S. 'Systems Engineering – coping with complexity'. London: Prentice Hall Europe; 1998

[Zachman 2008] Zachman, J.A. 'Concise Definition of the Zachman Framework'. Zachman International; 2008. Available from http://www.zachman.com/about-the-zachman-framework; accessed September 2014