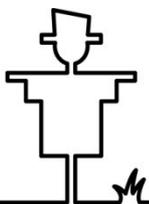


The Traceability Pattern

Date: 2016-04-14



Contents

1	Introduction	2
1.1	Pattern Aims.....	2
2	Concepts.....	3
3	Viewpoints	4
3.1	Overview	4
3.2	Rules.....	5
3.3	Relationship Identification Viewpoint (RIVp).....	7
3.3.1	Description	7
3.3.2	Example.....	8
3.4	Traceability Identification Viewpoint (TIVp).....	9
3.4.1	Description	10
3.4.2	Example.....	11
3.5	Traceability Viewpoint (TVp).....	13
3.5.1	Description	14
3.5.2	Example.....	15
3.6	Impact Viewpoint (IVp)	17
3.6.1	Description	17
3.6.2	Example.....	18
4	Summary	20
5	Related Patterns.....	20
6	References & Further Reading.....	20



1 Introduction

There are many interpretations of traceability, including:

- Measurement
- Logistics
- Materials
- Supply Chain
- Development

However, two themes run throughout all interpretations: an unbroken chain and access to information. The unbroken chain focuses on the need to be able to follow an artefact back to its source, whether that source is a physical place such as the sea, a supplier organisation, a requirement, meeting minutes or a standard such as ISO 15288. Access to information defines the need to know about the artefact being traced. In some cases this may be the history of the artefact. For example, in the case of food, this could be the organisations the food has passed through, when and who did the quality check etc. For an engineering project, it may be why an artefact exists or has been chosen as the solution to a requirement.

The existence of traceability supports areas of analysis including: impact analysis, change management and coverage analysis. All of these types of analysis would be difficult if not impossible to carry out without traceability in place.

1.1 Pattern Aims

This Pattern is intended to be used as an aid to the use of traceability within a Model-Based Systems Engineering (MBSE) application. The main aims of this Pattern are shown in the Architectural Framework Context View (AFCV) in Figure 1.

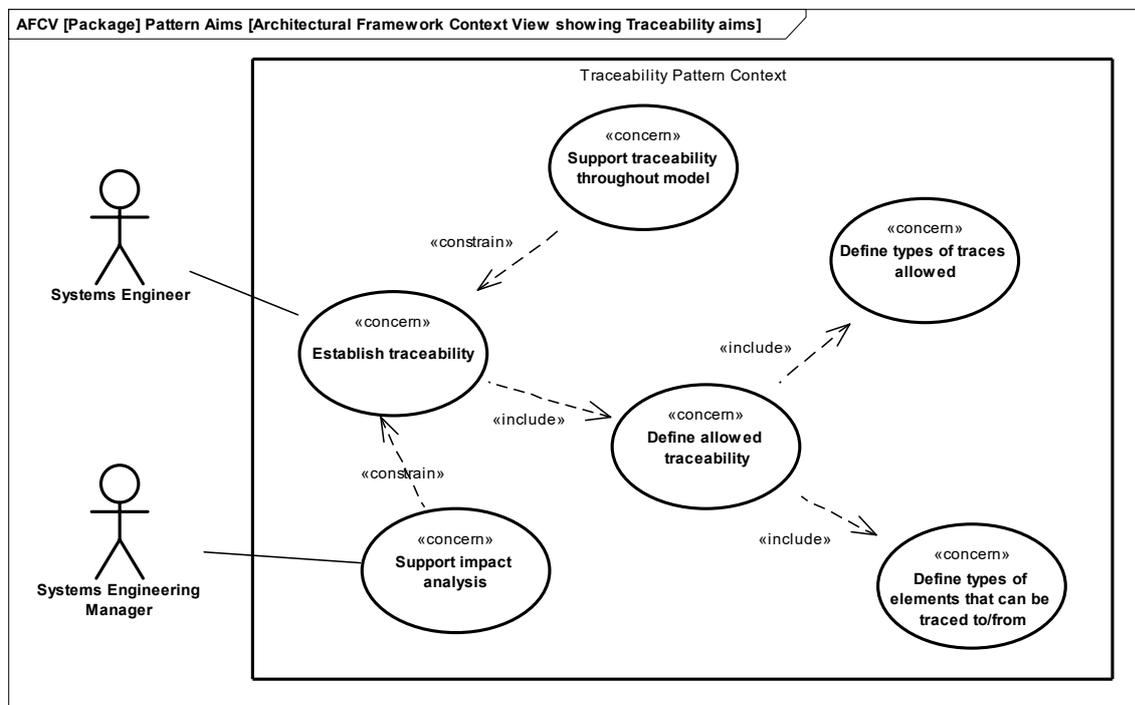


Figure 1 - Architectural Framework Context View showing Traceability aims

The key aim of the Traceability Pattern is to allow a 'Systems Engineer' to 'Establish traceability' in a model of a System. This key aim is constrained by the need to:

- 'Support traceability throughout model' – Support the capture of the traceability relationships between traceable elements in a systems engineering model, throughout the model and between any desired model elements.
- 'Support impact analysis' – Support 'Systems Engineers' and 'Systems Engineering Managers' in the identification of model elements that may be impacted by change.

The main aim of 'Establish traceability' includes the aim of 'Define allowed traceability'. This in turn includes the uses cases:

- 'Define types of trace allowed' – Support the definition of the types of traces that can be used.
- 'Define types of element that can be traced to/from' – Support the definition of the types of elements that can be involved in trace relationships and the relationships that can be used between such traceable elements.

The Use Case 'Establish traceability' provides benefit to the 'Systems Engineer' Stakeholder Role. The 'Systems Engineering Manager' Stakeholder Role gains benefit from the 'Support impact analysis' Use Case.

2 Concepts

The main concepts covered by the Traceability Pattern are shown in the Ontology Definition View (ODV) in Figure 2.

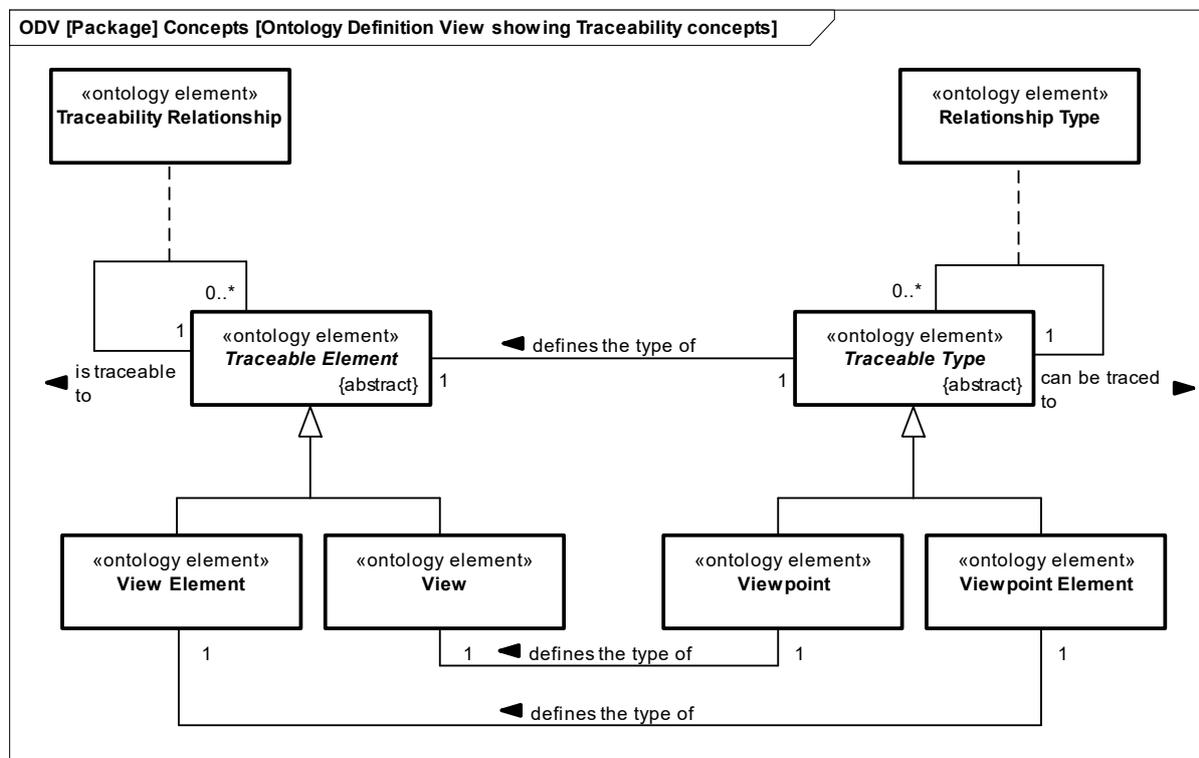


Figure 2 - Ontology Definition View showing Traceability concepts

Key to getting traceability right is establishing the understanding of the way in which traceability will be used. This means understanding the types of information to be traced and the relationships that can be used to realise the traceability.

The right-hand side of this diagram shows the types of things which can be traced - a 'Traceable Type'. These may be:

- 'Viewpoint', representing types of 'View' occurring in a model. 'Viewpoint' may represent underlying diagram types from the modelling language being used, such as a *block definition diagram* if using SysML. They may also represent defined Viewpoints from a framework that is being used, such as a "Validation Viewpoint" from a Model-Based Requirements Engineering framework. Viewpoints are conceptual in nature; they are a definition to which an instance (a View) must conform. See [Holt & Perry 2013].
- 'Viewpoint Element', representing types of 'View Element' occurring in a model. Viewpoint Elements may represent underlying modelling element types from the modelling language being used, such as a *block* if using SysML. They may also represent defined conceptual elements (Ontology Elements) that are used on Viewpoints from a framework that is being used, such as a "System Element" that is used on a "Validation Viewpoint" from a Model-Based Requirements Engineering framework. Viewpoint Elements are conceptual in nature and make up a Viewpoint; when an instance of a Viewpoint is created (i.e. a View is created), then that View is made up of View Elements. See [Holt & Perry 2013].

The right-hand side of the diagram also includes 'Relationship Type' that define the types of 'Traceability Relationship' that can be used to trace View Elements and Views, identified by the Viewpoints and Viewpoint Elements, one to another.

The left-hand side of the diagram shows 'Traceable Element'. This represents the actual things being traced between i.e. the 'View' or 'View Element'. The traceability is made using a 'Traceability Relationship', a representation of the actual relationship which is being considered.

The concepts on the right-hand side of the diagram are conceptual. They represent the types of things that can be traced ('Traceable Type') which can be a 'Viewpoint' or a 'Viewpoint Element', together with the type of relationships that can be used ('Relationship Type'). The concepts on the left-hand side of the diagram are concrete. They represent the actual things traced ('Traceable Element') which can be a 'View' or a 'View Element', together with the type of traceability connecting them ('Traceability Relationship').

Many people consider and define traceability, fewer fully consider the way in which traceability is defined. Often where it is defined it is considered as a database schema, which although useful may not provide a direct relationship to the way a project is being conducted or what that project is trying to achieve though the provision of traceability.

3 Viewpoints

This section describes the Viewpoints that make up the Traceability Pattern. It begins with an overview of the Viewpoints, defines Rules that apply to the Pattern and then defines each Viewpoint.

3.1 Overview

The Traceability Pattern defines a number of Viewpoints as shown in the Viewpoint Relationship View (VRV) in Figure 3.



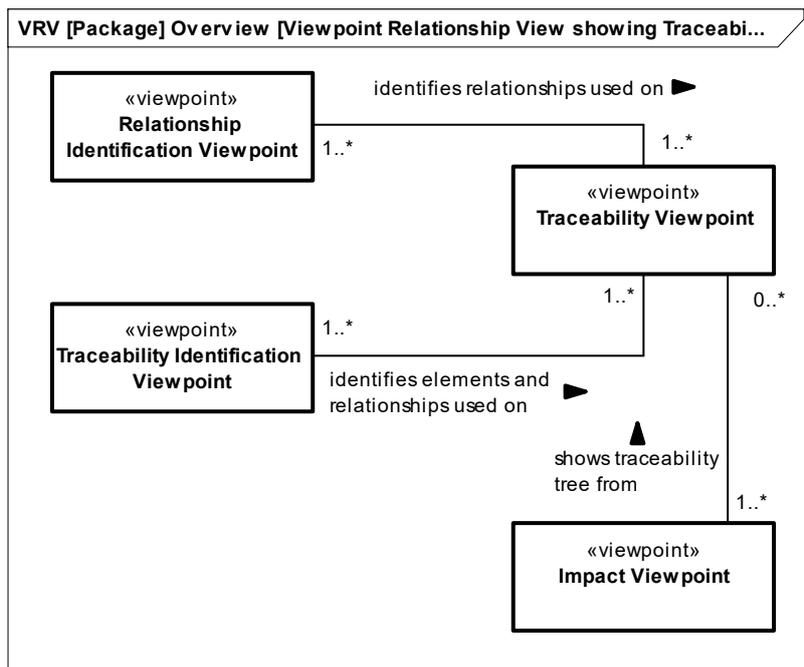


Figure 3 - Viewpoint Relationship View showing Traceability Viewpoints

The Traceability Pattern defines four Viewpoints to enable the definition and capture of traceability:

- The 'Relationship Identification Viewpoint' is used to define the types of permissible Relationship Types.
- The 'Traceability Identification Viewpoint' is used to define the items that can be traced (the Traceable Types) and the types of trace that can be used between items (the Relationship Types between pairs of Traceable Types).
- The 'Traceability Viewpoint' is used to capture and visualise traceability between Traceable elements through Traceability Relationships. This may be in the form of a diagram, table, matrix etc.
- The 'Impact Viewpoint' is used to show a traceability tree for a selected Traceable Element from a Traceability View, allowing the items potentially impacted by changes to the root of the tree (the selected Traceable Element) to be identified.

3.2 Rules

Five rules apply to the four Traceability Viewpoints, as shown in the Rules Definition View (RDV) in Figure 4.



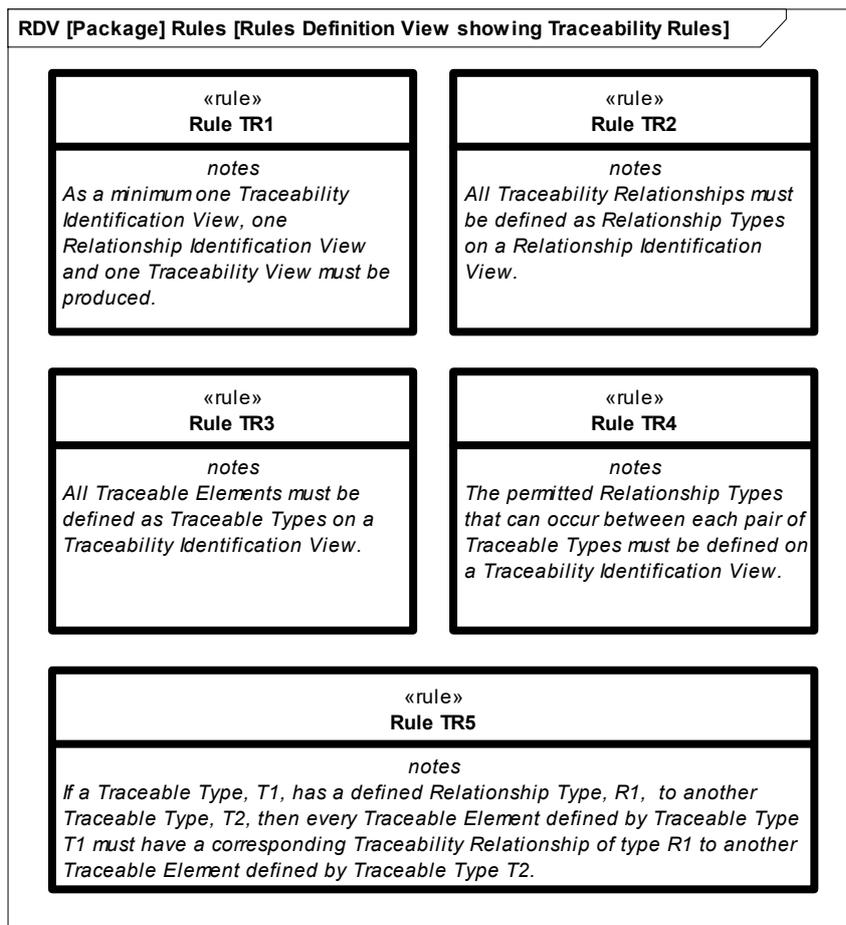


Figure 4 - Rules Definition View showing Traceability Rules

These five Rules are:

- ‘Rule TR1: As a minimum one Traceability Identification View, one Relationship Identification View and one Traceability View must be produced.’ This Rule enforces a minimum set of Views that must be produce. If this minimum set is not produced then an incomplete definition of Traceability will created.
- ‘Rule TR2: All Traceability Relationships must be defined as Relationship Types on a Relationship Identification View.’ This Rule is there to ensure that the types of traceability relationships that will be used are defined. No Relationship Type can be used that has not been defined.
- ‘Rule TR3: All Traceable Elements must be defined as Traceable Types on a Traceability Identification View.’ Again, a Rule to ensure consistency. Traceability can only be established between the kinds of Traceable Types that have been defined.
- ‘Rule TR4: The permitted Relationship Types that can occur between each pair of Traceable Types must be defined on a Traceability Identification View.’ This Rule works with Rules TR2 and TR3 to ensure that the allowed Relationship Types that can be used between Traceable Types are defined before use.
- ‘Rule TR5: If a Traceable Type, T1, has a defined Relationship Type, R1, to another Traceable Type, T2, then every Traceable Element defined by Traceable Type T1 must have a corresponding Traceability Relationship of type R1 to another Traceable Element defined by Traceable Type T2.’ This wordy Rule simply says that if you have said that a particular type of traceability is to be used between particular types of elements, then when performing traceability all elements of those types must be so traced.



Note that the five Rules shown in Figure 4 are the minimum that are needed. Others could be added if required.

3.3 Relationship Identification Viewpoint (RIVp)

The aims of the Relationship Identification Viewpoint are shown in the Viewpoint Context View in Figure 5.

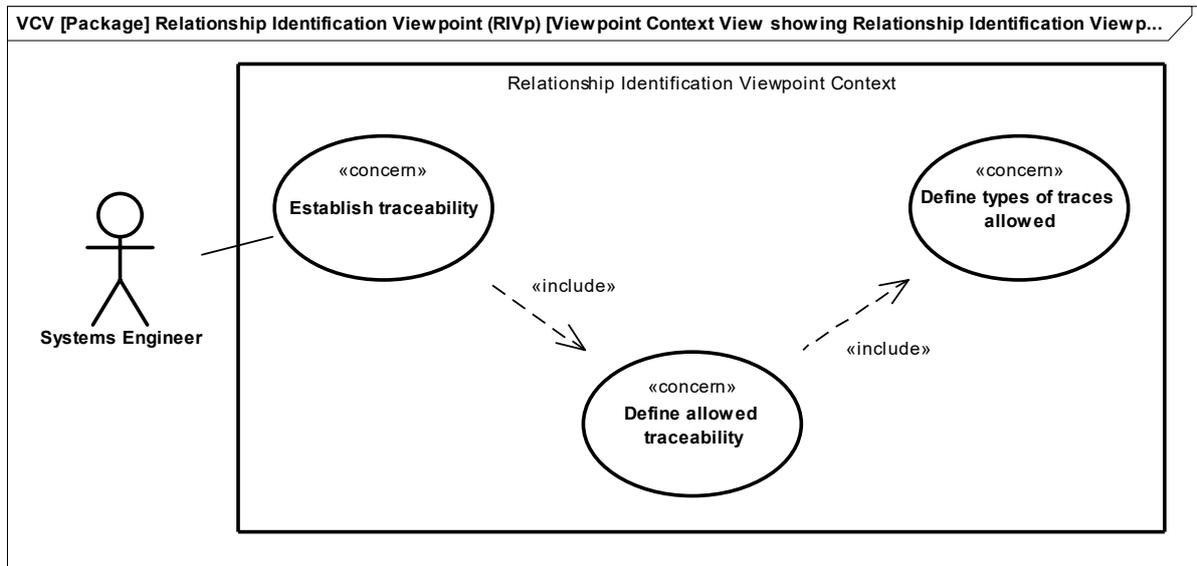


Figure 5 - Viewpoint Context View showing Relationship Identification Viewpoint aims

The main aim of the Relationship Identification Viewpoint is to ‘Establish traceability’ through the aim of ‘Define allowed traceability’. In particular, its key aim is to ‘Define types of traces allowed’, that is the Relationship Identification Viewpoint identifies the allowed Relationship Types that can be used to establish traceability.

3.3.1 Description

The Viewpoint Definition View (VDV) in Figure 6 shows the Ontology Elements that appear on a Relationship Identification Viewpoint.

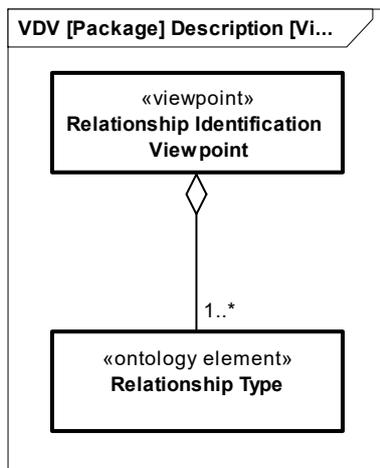


Figure 6 - Viewpoint Definition View showing the Ontology Elements that appear on the Relationship Identification Viewpoint (RIVp)



The Relationship Identification Viewpoint shows a number of Relationship Types.

Relationship Type defines the relationships that will be used to capture traceability. In many cases, traceability relationships are assumed to be loose relationships added by a requirements or traceability tool. However, this Viewpoint also enables other concepts such as parent/child relationships to represent traceability.

A Relationship Identification Viewpoint would typically be based on the Ontology of a specified Pattern or application (such as requirements engineering), adding detail relating to the types of traceability relationship to fully define the information to be captured.

3.3.2 Example

An example View that conforms to the Relationship Identification Viewpoint is shown in Figure 7. This, and subsequent examples, are taken from the domain of Model-Based Systems Engineering (and Model-Based Requirements Engineering in particular). However, this is only one possible application of traceability and, while perhaps the most common use within the systems engineering domain and hence the reason it was chosen for the examples, it is not the only one.

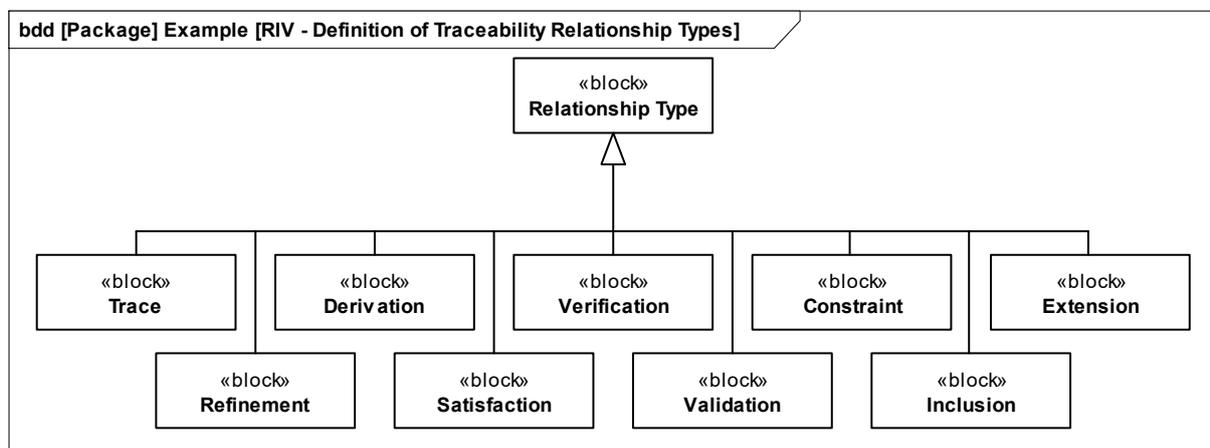


Figure 7 - RIV - Definition of Traceability Relationship Types

The Relationship Identification View in Figure 7, realised as a SysML *block definition diagram*, defines nine Relationship Types: 'Trace', 'Refinement', 'Derivation' etc., that have been identified as being the kinds of traceability needed when carrying out model-based requirements engineering.

Each of the defined Relationship Types should also be accompanied by a description of its intended use. This has been done in the following table, which also serves as a text-based version of a Relationship Identification View:

Relationship Identification View	
Relationship Type	Description
Trace	A general purpose relationship type that can be used if none of the other, more specific, relationship types is suitable. For example, to indicate that a Requirement traces to a Source Element.



Relationship Identification View	
Relationship Type	Description
Refinement	Indicates that one Requirement refines another or that a Use Case refines a Requirement.
Derivation	Indicates that one Requirement is derived, in whole or part, from another Requirement.
Satisfaction	Indicates that a Requirement satisfies a Rationale or Capability or that a Capability satisfies a Goal or that a System Element satisfies a Use Case.
Verification	Indicates that a Test Case verifies a Requirement.
Validation	Indicates that a Test Case validates a Use Case.
Constraint	Indicates that one Requirement constrains another or that one Use Case constrains another.
Inclusion	Indicates that one Requirement includes another as a sub-Requirement or that one Use Case includes another.
Extension	Indicates that one Use Case extends another under stated circumstances.

Table 1 - An example of a text-based Traceability Identification View for MBRE

This View, defined in Figure 7 and Table 1, fulfils Rule TR2. With the Traceability Identification View in Figure 10 and Table 2 and the Traceability View in Figure 13, it fulfils Rule TR1.

3.4 Traceability Identification Viewpoint (TIVp)

The aims of the Traceability Identification Viewpoint are shown in the Viewpoint Context View in Figure 8.



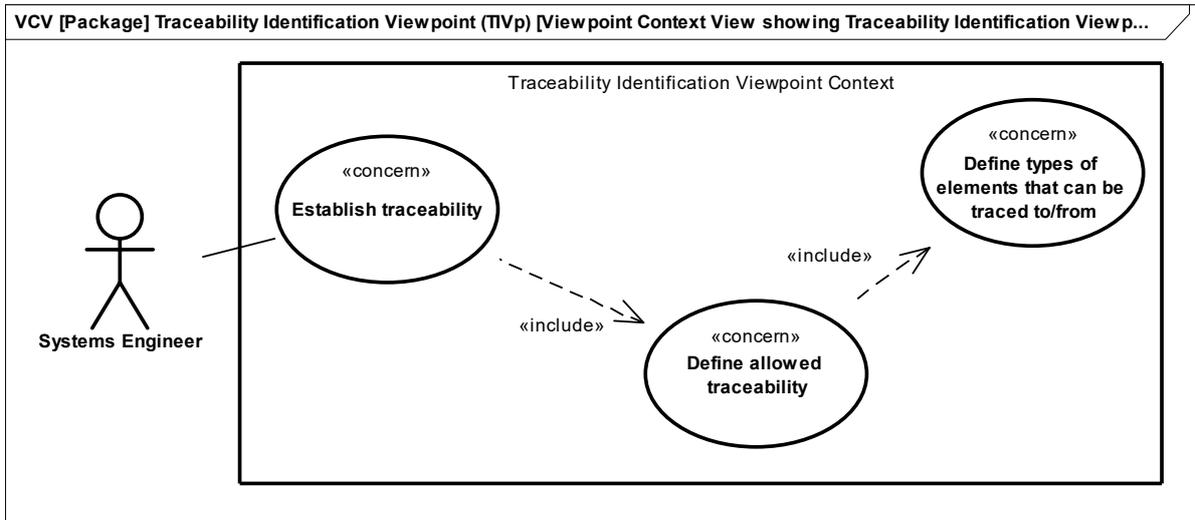


Figure 8 - Viewpoint Context View showing Traceability Identification Viewpoint aims

The main aim of the Traceability Identification Viewpoint is to ‘Establish traceability’ through the aim of ‘Define allowed traceability’. In particular, its key aim is to ‘Define types of elements that can be traced to/from’, that is the Traceability Identification Viewpoint identifies the allowed Traceable Types as well as the allowed Relationship Types that can be used to establish traceability between any two Traceable Types.

3.4.1 Description

The Viewpoint Definition View (VDV) in Figure 9 shows the Ontology Elements that appear on the Traceability Identification Viewpoint.

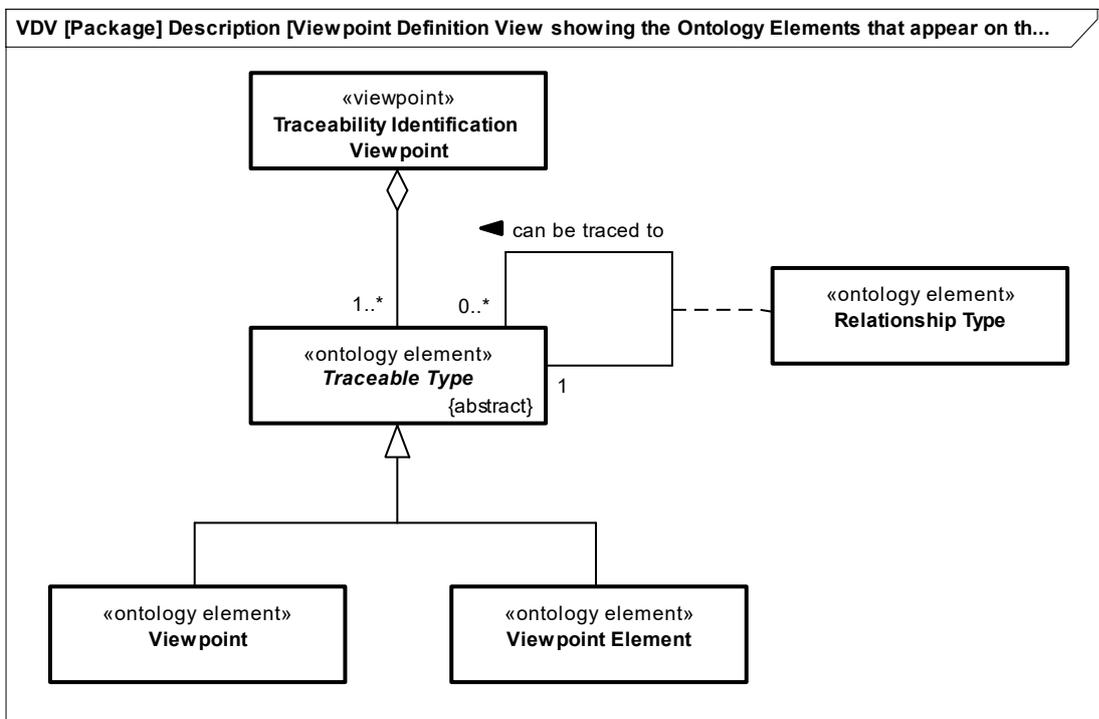


Figure 9 - Viewpoint Definition View showing the Ontology Elements that appear on the Traceability Identification Viewpoint (RIVp)

The *blocks* marked as representing Viewpoints and Viewpoint Elements indicate those elements which can be traced. The *associations* with the «Trace» *stereotype* indicate the possible traceability Relationship Types that are valid between the *blocks* linked by the *association*. Where the «Trace» *stereotype* has been applied, further information about the detail of the trace is also shown through the use of *tags*:

- 'Trace to' - the direction of the trace
- 'Trace type' - the type of trace, e.g. Trace, Validate, Inclusion etc.

One item of note on the diagram concerns the *association* between the 'Stakeholder Role' (from the 'Context Definition Viewpoint') and the 'Requirement Context Viewpoint'. The trace indicated on this *association* is shown as applying in the opposite direction to the reading direction of the *association*. The *association* is read from 'Stakeholder Role' to 'Requirement Context Viewpoint' but the trace goes to the 'Stakeholder Role' rather than from it. This is due to the nature of traceability, which is often considered as going back towards the source of the information. The intention here is to show that a 'Requirement Context Viewpoint' can be traced back to its source 'Stakeholder Role' on a 'Context Definition Viewpoint'.

The same kind of information as shown in Figure 10 may also be shown textually as in Table 2, which shows a superset of the information found in Figure 10.

Traceability Identification View		
Traceable Type (Viewpoint or Viewpoint Element)		Relationship Type
From	To	
Requirement	Source Element	Trace
	Requirement	Derivation
		Inclusion
		Constraint
		Refinement
	Rationale	Satisfaction
	Capability	Satisfaction
Use Case	Requirement	Refinement



Traceability Identification View		
Traceable Type (Viewpoint or Viewpoint Element)		Relationship Type
From	To	
	Use Case	Constraint
		Inclusion
		Extension
System Element	Use Case	Satisfaction
Test Case (Validation View)	Use Case	Validation
Test Case	Requirement	Verification
Requirement Context View	Stakeholder Role (on Context Definition Viewpoint)	Trace
Capability	Goal	Satisfaction

Table 2 - An example of a text-based Traceability Identification View for MBRE

The Traceability Identification View shown in Table 2 defines the Traceable Types (Viewpoints and Viewpoint Elements) and Relationship Types that can exist between them for the Approach to Context-Based Requirements Engineering (ACRE) model-based requirements engineering Framework discussed in chapter 13.

The first two columns define the Traceable Types, giving the source and destination of possible relationships. The final column defines the possible relationships that can hold between each pair of Traceable Types. Only those Relationship Types defined on a Relationship Identification View may appear in this column.

This View, defined in Figure 10 and Table 2, fulfils Rules TR3 and TR4. With the Relationship Identification View in Figure 7 and Table 1 and the Traceability View in Figure 13, it fulfils Rule TR1.

3.5 Traceability Viewpoint (TVp)

The aims of the Traceability Viewpoint are shown in the Viewpoint Context View in Figure 11.



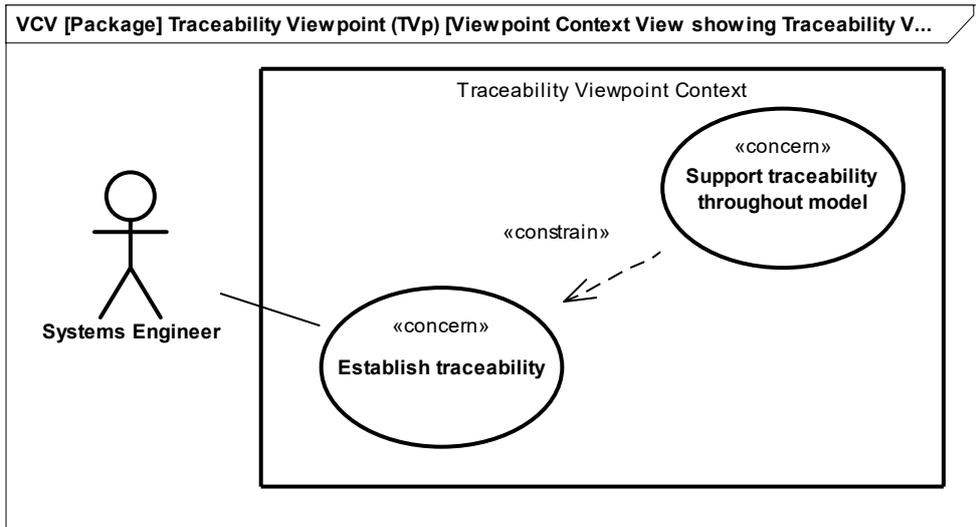


Figure 11 - Viewpoint Context View showing Traceability Viewpoint aims

The main aim of the Traceability Viewpoint is to ‘Establish traceability’ in a model, with the constraint of ‘Support traceability throughout model’. That is, the Traceability Viewpoint allows traceability to be captured and visualised throughout the model of a System; it is not something that is restricted to Requirements traceability. Traceability Views that conform to the Traceability Viewpoint may be in the form of diagrams, tables, matrices, trees, etc.

3.5.1 Description

The Viewpoint Definition View (VDV) in Figure 12 shows the Ontology Elements that appear on a Traceability Viewpoint.

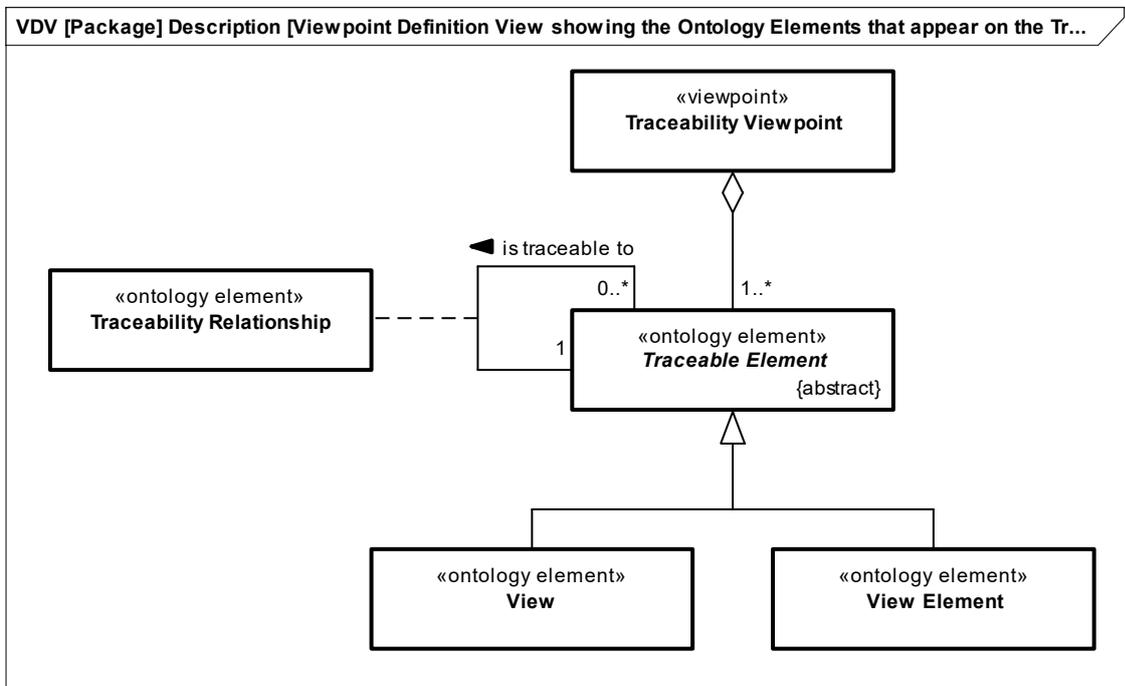


Figure 12 - Viewpoint Definition View showing the Ontology Elements that appear on the Traceability Viewpoint (TVp)

The Traceability Viewpoint shows a number of Traceable Elements and the Traceability Relationships between them. Traceable Elements may be Views or View Elements and Traceability Relationships

may be made View-to-View, View Element-to-View Element, View Element-to-View or View-to-View Element.

The allowed Traceability Relationships are those of the Relationship Types that are defined on a Relationship Identification View. The allowed Traceable Elements are those of the Traceable Types that are defined on a Traceability Identification View. Only the connections between pairs of Traceable Types defined on a Traceability Identification View are permitted on a Traceability View.

3.5.2 Example

An example View that conforms to the Traceability Viewpoint is shown in Figure 13.

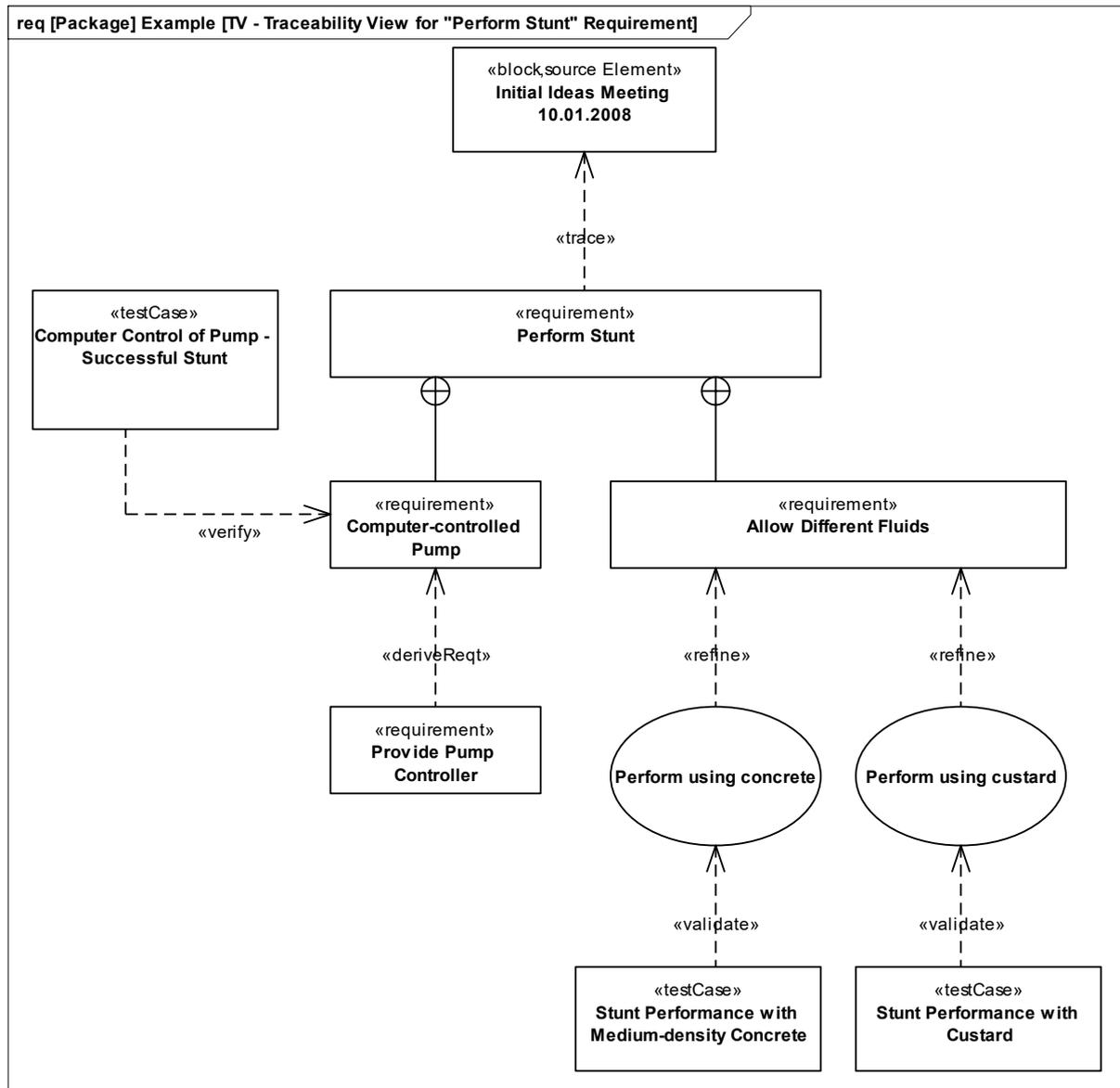


Figure 13 - TV - Traceability View for "Perform Stunt" Requirement

The Traceability View in Figure 13, here realised as a SysML *requirements diagram*, shows six different types of Traceability Relationship between a number of Traceable Elements:

- A Trace relationship between the Requirement 'Perform Stunt' and its Source Element 'Initial Ideas Meeting 10.01.2008', realised using a SysML **«trace»** relationship.



- Two Inclusion relationships between the sub-Requirements ‘Computer-controlled Pump’ and ‘Allow Different Fluids’ and their parent Requirement ‘Perform Stunt’, realised using SysML *nesting relationships*.
- A Derivation relationship between the Requirement ‘Provide Pump Controller’ and the Requirement ‘Computer-controlled Pump’ that it is derived from, realised using a SysML *«deriveReq» relationship*.
- A Verification relationship between a Test Case ‘Computer Control of Pump - Successful Stunt’ and the Requirement ‘Computer-controlled Pump’ that it verifies, realised using a SysML *«verify» relationship*.
- Two Refinement relationships between the Use Cases ‘Perform using concrete’ and ‘Perform using custard’ and the Requirement ‘Allow Different Fluids’ that they refine, realised using a SysML *«refine» relationship*.
- Two Validation relationships between the Test Cases ‘Stunt Performance with Medium-density Concrete’ and ‘Stunt Performance with Custard’ and the Use Cases ‘Perform using concrete’ and ‘Perform using custard’ that they validate, realised using a SysML *dependency* stereotyped *«validate»*. Note that SysML does not have a "built-in" *«validate» relationship*.

As an alternative to a graphical representation, the same information could be represented as a simple matrix as shown in Table 3.

		Traces From								
		«requirement» Perform Stunt	«requirement» Computer-controlled Pump	«requirement» Allow Different Fluids	«requirement» Provide Pump Controller	«testCase» Computer Control of Pump - Successful Stunt	«Use Case» Perform using concrete	«Use Case» Perform using custard	«testCase» Stunt Performance with Medium-density Concrete	«testCase» Stunt Performance with Custard
Traces To	«Source Element» Initial Ideas Meeting 10.01.2008	Trace «trace»								
	«requirement» Perform Stunt		Inclusion (nesting)	Inclusion (nesting)						
	«requirement» Computer-controlled Pump				Derivation «deriveReq»	Verification «verify»				
	«requirement» Allow Different Fluids						Refinement «refine»	Refinement «refine»		
	«Use Case» Perform using concrete								Validation «validate»	
	«Use Case» Perform using custard									Validation «validate»

Table 3 - An example of a Traceability View represented as a matrix



Note that the Traceability View defined in Figure 13 and Table 3 has been created to show the traceability, both forwards and backwards, for a single Requirement, 'Perform Stunt'. However, this is only one possibility. There is no restriction on the focus that each Traceability View has nor on the number of levels shown. A Traceability View could be created showing, for example, a number of Source Elements and all the Requirements that trace to them (at a single level of traceability), or could concentrate on a single Source Element and everything that ultimately traces to this (multi-level traceability); this would be similar to Figure 13 but would show multiple Requirements tracing to the Source Element.

This View, defined in Figure 13 and Table 3, along with the Relationship Identification View in Figure 7 and Table 1 and the Traceability Identification View in Figure 10 and Table 2, fulfils Rules TR1, TR2, TR3 and TR4.

3.6 Impact Viewpoint (IVp)

The aims of the Impact Viewpoint are shown in the Viewpoint Context View in Figure 14.

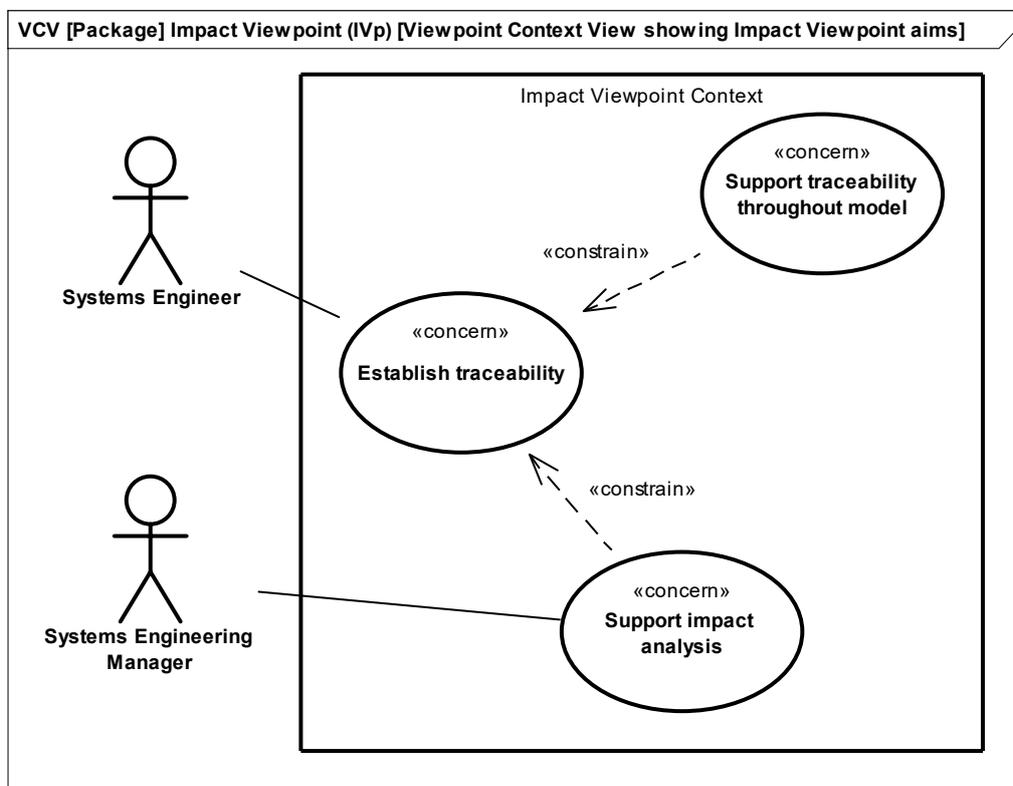


Figure 14 - Viewpoint Context View showing Impact Viewpoint aims

The main aim of the Impact Viewpoint is to 'Establish traceability' in a model, with the constraints of 'Support traceability throughout model' and 'Support impact analysis'. An Impact View that conforms to the Impact Viewpoint is used to show a traceability tree for a selected Traceable Element from a Traceability View, allowing the items (the other Traceable Elements connected to it through a network of Traceability Relationships) that may be impacted by changes to the root of the tree (the selected Traceable Element) to be identified.

3.6.1 Description

The Viewpoint Definition View (VDV) in Figure 15 shows the Ontology Elements that appear on an Impact Viewpoint.



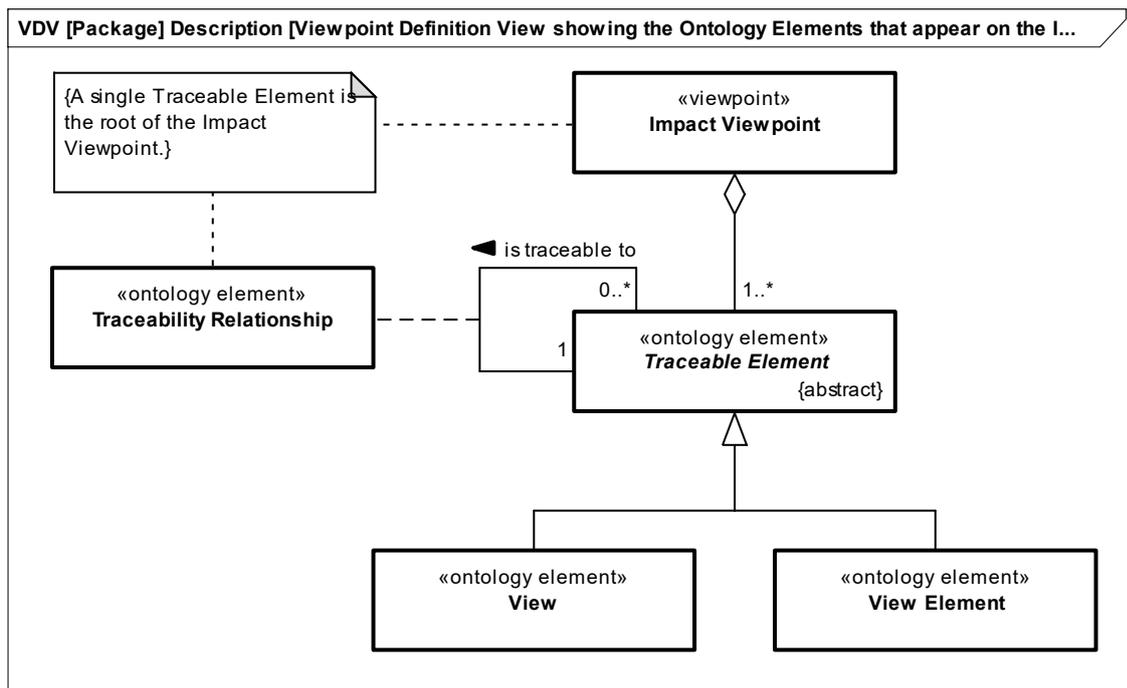


Figure 15 - Viewpoint Definition View showing the Ontology Elements that appear on the Impact Viewpoint (IVp)

Like the Traceability Viewpoint, the Impact Viewpoint shows a number of Traceable Elements and the Traceability Relationships between them. Indeed, the Ontology Elements that can appear on the Impact Viewpoint and the rules governing consistency of those elements are the same as for the Traceability Viewpoint.

Where the two Views differ is in the form and intent of the Views created based on the Viewpoints. A Traceability View can show any number of levels of traceability and may or may not have a single "root" element that everything traces to or from. An Impact View, however, always has a single root element and typically shows all the levels of traceability starting from that root element. Whereas a Traceability View may show a web of related elements, an Impact View always shows a tree of related elements, starting from the element that is the root of the tree.

3.6.2 Example

An example View that conforms to the Impact Viewpoint is shown in Figure 16.



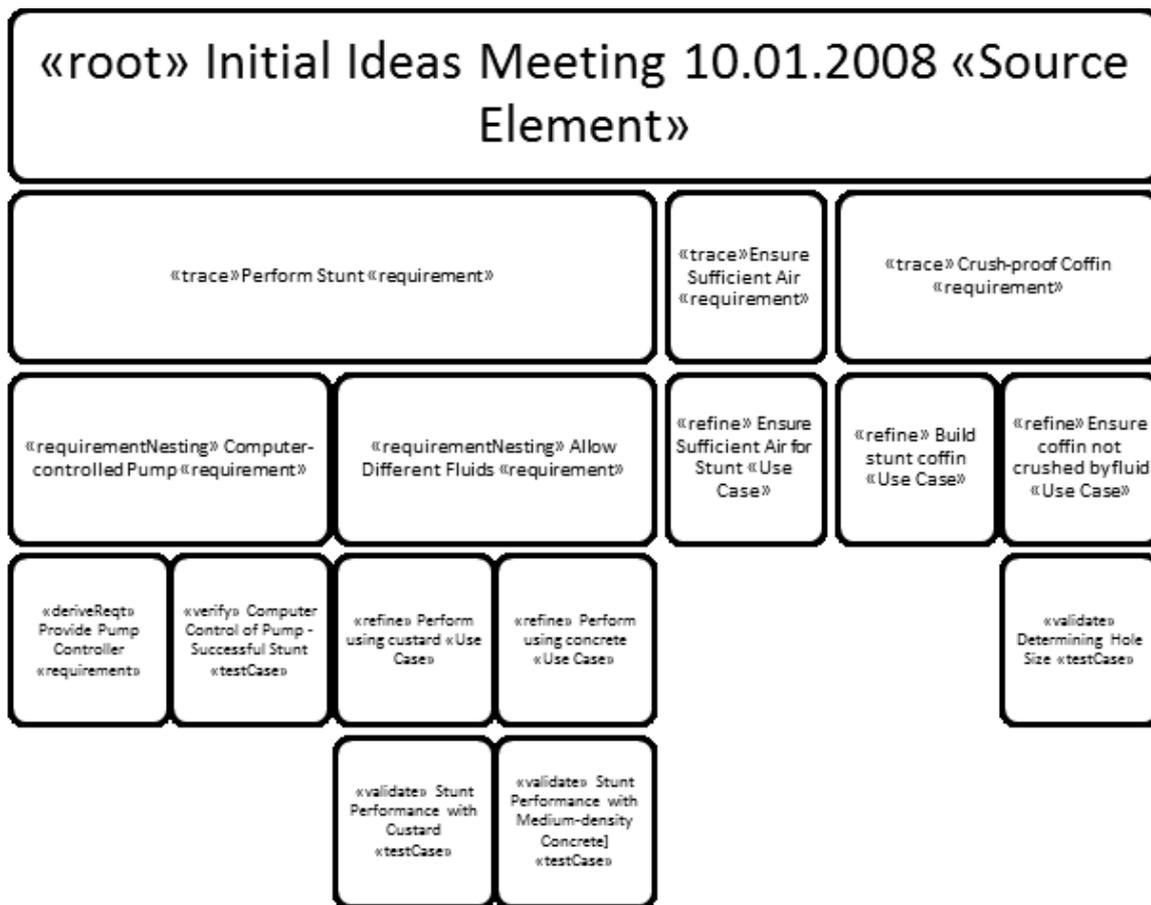


Figure 16 - An example of an Impact View

The Impact View in Figure 16 shows the forward impact tree for the Source Element 'Initial Ideas Meeting 10.01.2008', showing all 15 potential model elements that may be impacted by a change to the Source Element. Note that there is an implicit meaning in the layout of the diagram: items in a given level trace to the item lying over them in the level above. The nature of the trace relationship is shown in the *stereotype* that forms the first part of the name of each item and the type of item traced is shown in the *stereotype* that forms the last part of the name. For example, from the diagram it can be seen that the Use Cases 'Build stunt coffin' and 'Ensure coffin not crushed by fluid' both trace, through refinement, to the Requirement 'Crush-proof Coffin'.

Other graphical representations are, of course, possible. The same information may also be presented as text, as shown in Table 4. Here, the level of indentation is used to show the hierarchical relationships between the elements; those at the same level trace to the element above them that is indented one level less.

```

«root» Initial Ideas Meeting 10.01.2008 «Source Element»
  «trace» Perform Stunt «requirement»
    «requirementNesting» Computer-controlled Pump «requirement»
      «deriveReq» Provide Pump Controller «requirement»
        «verify» Computer Control of Pump - Successful Stunt «testCase»
      «requirementNesting» Allow Different Fluids «requirement»
  «trace» Ensure Sufficient Air «requirement»
    «refine» Ensure Sufficient Air for Stunt «Use Case»
  «trace» Crush-proof Coffin «requirement»
    «refine» Build stunt coffin «Use Case»
    «refine» Ensure coffin not crushed by fluid «Use Case»
    «validate» Determining Hole Size «testCase»
    «validate» Stunt Performance with Custard «testCase»
    «validate» Stunt Performance with Medium-density Concrete «testCase»
    
```



«refine» Perform using custard «Use Case»
 «validate» Stunt Performance with Custard «testCase»
«refine» Perform using concrete «Use Case»
 «validate» Stunt Performance with Medium-density Concrete «testCase»
«trace» Ensure Sufficient Air «requirement»
 «refine» Ensure Sufficient Air for Stunt «Use Case»
«trace» Crush-proof Coffin «requirement»
 «refine» Build stunt coffin «Use Case»
 «refine» Ensure coffin not crushed by fluid «Use Case»
 «validate» Determining Hole Size «testCase»

Table 4 - An example of an Impact View represented using text

Whatever the graphical representation used, what is essential is that such Impact Views are capable of being generated automatically based on the trace information added to the model of a System. The information in Figure 16 and Table 4 was generated automatically from the SysML tool that was used to create the model.

4 Summary

The Traceability Pattern defines four Viewpoints that allow aspects of traceability to be captured. The Relationship Identification Viewpoint defines the possible types of traceability relationships that may be used. The Traceability Identification Viewpoint defines the types of model elements that can be the sources and targets of traceability and the relationships (from the Relationship Identification Viewpoint) that can hold between them. The Traceability Viewpoint shows the *actual* traceability relationships that hold between model elements that conform to the allowed types for traceability (as defined on the Traceability Identification Viewpoint). Finally, the Impact Viewpoint allows a traceability *impact* tree to be produced for a selected *root* model element, aiding in the identification of those model elements that may be impacted by changes to the selected root element.

5 Related Patterns

If using the Traceability Pattern, the following Patterns may also be of use:

- Testing
- Interface Definition

6 References & Further Reading

Holt, J & Perry, S. SysML for Systems Engineering; 2nd Edition: A model-based approach. London: IET Publishing; 2013

Object Management Group. SysML website [online]. Available from <http://www.omg.sysml.org>; Accessed February 2015



Holt, J. UML for Systems Engineering - watching the wheels. 2nd edn. London: IEE Publishing; 2004 (reprinted IET Publishing; 2007)

Rumbaugh, J., Jacobson, I. & Booch, G. The Unified Modeling Language Reference Manual. 2nd edition. Boston, MA: Addison-Wesley; 2005

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

Boehm, BW, Verifying and Validating Software Requirements and Design Specifications, reprinted in Boehm, B. W. (ed.), Software Risk Management, pp. 205 - 218, IEEE Computer Society Press, 1989.

