INCOSE UK Annual Systems Engineering Conference 2015 (ASEC2015)
Oxfordshire, UK, November 17-18, 2015

From catalogues to models: transitioning from existing requirements techniques to MBSE

James Towers
Scarecrow Consultants
james.towers@scarecrowconsultants.co.uk

Abstract. There is a growing consensus that the levels of complication we face in modern systems engineering (SE) projects cannot be controlled via a traditional document-centric approach. This is encouraging organisations to adopt a model-based systems engineering (MBSE) approach. However the transition is not always straightforward. Organisations with a mature document-centric requirements practice often have a significant investment in both existing tools and processes. The move to MBSE raises several questions about how these tools and processes 'fit' in the new world. Often the conclusion is a hybrid approach which attempts to get the maximum benefit from existing assets while attempting to 'cherry pick' the best bits of MBSE. This may, however, introduce more problems than benefits and can prevent an organisation from realising the full benefits of MBSE such as automated model checking.

Document-Centric Requirements

Traditionally, a document-centric approach has been the de facto way of recording, validating and communicating requirements and other systems engineering artefacts. The use of natural language e.g. English, and commonly available office applications capable of storing and manipulating text, such as Microsoft Word, Excel and PowerPoint, make it appear a simple, flexible and easily accessible way to work. Such techniques have become known as 'document-centric' since, while much of the work is performed within the tools, the practice has developed from an initial objective of generating a printable (or displayable) 'document' for review and distribution. One of the advantages of this approach is its apparent flexibility. However one of its major disadvantages is that there is no or little defined linkage between documents. If you wish to rename or redefine a concept, a considerable amount of effort is required to find all occurrences of it. Even if you are able to do this systematically, the overloaded semantics of natural language make it difficult to automate updates. e.g. you may be able to do a 'search and replace' in order to change "ship" to 'boat" but it is likely (unless you are using a very clever tool) that you will need to manually inspect every occurrence in order to determine if it is being used as a verb or a noun, since changing "ship the Order" to "boat the Order" doesn't make much sense in every-day language!

More mature approaches utilise techniques such as structured text patterns (i.e. 'requirement boilerplates') like the Easy Approach to Requirements Syntax (EARS) [1], defined requirements-ontologies and specialist requirement-management tools such as *IBM Rational DOORS* and others. While these provide many advantages over less mature methods and tools - e.g. the use of a single repository within the tool for requirements, validation and verification information reduces (but doesn't eliminate) the inter-document reference problem - they are still dependent on a significant amount of manual maintenance and visual inspection. These tasks become even harder and require increasing effort as the size of the requirements catalogue increases. Figure 1. Illustrates and example of an EARS style requirement.

The pattern for an event-driven functional requirement is as follows -

When [Trigger] [Precondition] Actor Action [Object]

e.g.

"When an Order is shipped and Order Terms are not 'prepaid', the System shall create an Invoice."

Trigger:	*an Order is shipped*
Precondition:	*Order Terms are not 'prepaid'*
Actor:	*the System*
Action:	*create*
Object:	*an Invoice*

Figure 1. Example of a requirement using EARS.

Hybrid Document-Centric / Model-Based Approach

One of the fundamental advantages of MBSE is that while the information may be presented in multiple formats (as per document-centric approaches) all *Views* are derived from a single underlying model [2]. This means there is less maintenance. It is easier to assess the impact of change, and once agreed, these changes generally only need to be made in a single place. The stricter semantics of a purpose-defined modelling language also mean that it is possible to implement automated model checking for verification. Further advantages of a good system-model are described by Long & Scott in *A primer for Model-Based Systems Engineering* [3] as:

1. Providing order, which allows the design team to attack the problem in a coherent and consistent manner.
2. Having the power to demonstrate and persuade,
3. Providing greater integrity and consistency, and
4. Providing greater insight into both the problem and the solution.

All these factors together make MBSE look like an attractive proposition. However, organisations that currently use mature document-centric requirements approaches, while often wanting to do more MBSE, are sometimes reluctant to do less document-centric requirements management for a number of reasons:

1. The risk in moving away from what is considered to be an established 'proven' technique to a newer 'less used' technique is perceived to be too high.
2. There may be resistance from external stakeholders.
3. There is a perceived lack of functionality for the management of requirements within current MBSE tools compared to specialist requirements management ones.
4. An unfamiliarity with model-based requirement techniques results in what appears to be an overwhelming set of potential issues. It just looks too hard.
5. They have already made a significant investment in tools and process from which they wish to maximise return.
6. The authority for the use of the two techniques may be different i.e. a corporate requirements process may be mandated across all projects while MBSE may only be being used on a subset. (It is assumed that the organisation is unaware or doesn't believe this to be an issue.)

The solution to this tension is often seen as a hybrid approach where requirements are 'authored' within the existing toolset and in accordance with any existing process before being 'migrated' to an MBSE tool where they are 'used' in a system-model. This approach results in two parallel representations, more work than simply using one tool alone and ironically, the very issue that MBSE is intended to address i.e. the separation and duplication of information that then requires effort to maintain and verify. Additionally this hybrid approach introduces a number of new questions:

1. At what point(s) in the system lifecycle and under what circumstances should requirements be copied from the requirements-catalogue to the system-model?
2. What form should 'requirements' take in the system-model?
3. How are requirements edited, particularly if the required edits are identified while being 'used' within the system-model?
4. Which tool should be used for which systems engineering task?
5. What is the extra administrative effort required by this approach?

The complexities of these questions are discussed in the following sub-sections.

Synchronisation Point

If two separate tools are used with overlapping content, then a process to copy information from one to the other is required. A commonly held but mistaken belief is that the *requirements* must precede the system-model. This is possibly because the term *'system-model'* is often wrongly equated with a narrower model of the solution. In this case the following sequence, overlaid on a simplified 'v' lifecycle, is often implemented:

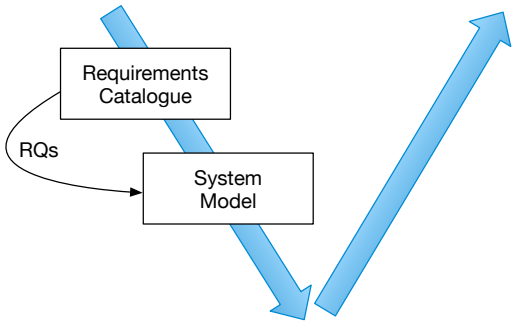


Figure 2. Requirements first then System-Model

In this scenario, *requirements* are copied from the catalogue to the model when they are assessed as 'stable' against some measure of consistency and completeness. The disadvantage of this approach is that we are unable to take advantage of any of the positive features previously described for a model, particularly automated model checking, until this import has taken place. If the assertion that *"The system-model will be a more complete and consistent representation than the requirements-catalogue"* is true, then we are effectively delaying the production of a higher-quality representation in favour of a lower-quality one, something that contradicts the SE principle of 'left shift'. If the assertion is false, then the question becomes *"why bother with the system-model at all then?"*

'Requirements' Form

The 'form' of the requirement within the system-model will be dependent on the modeling language used. For the purpose of this discussion I will consider The OMG (Object Management Group) SysML (Systems Modeling Language), which is INCOSE's preferred

MBSE language. SysML includes the concept of a «requirement» model-element. An example using the previously defined *EARS requirement statement* is shown below together with some additional concepts and relationships:

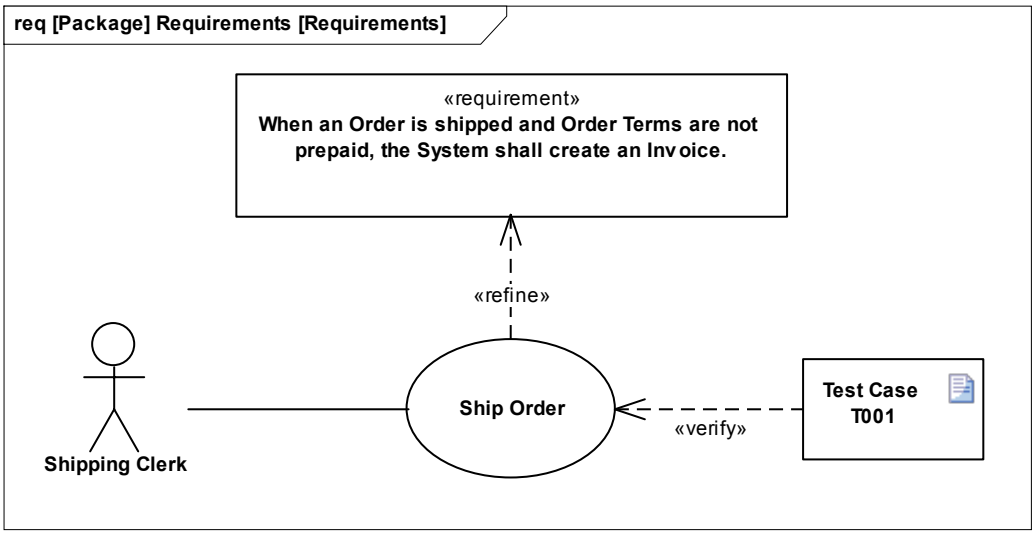
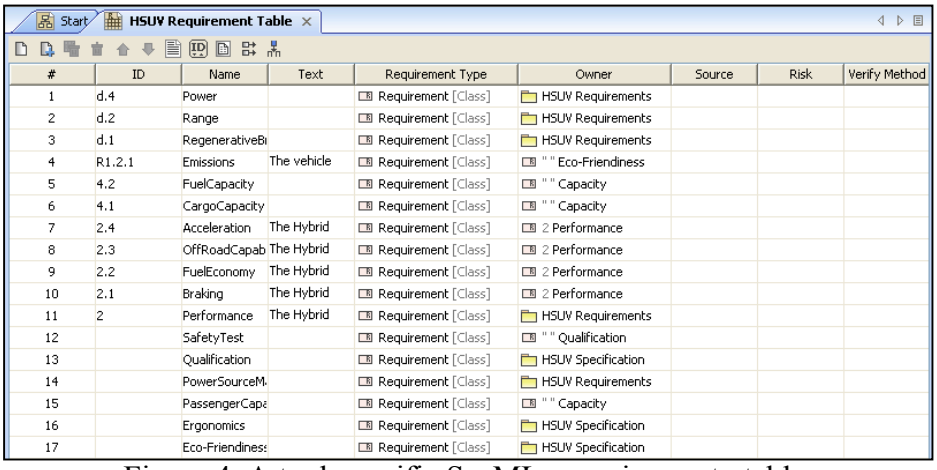


Figure 3. A SysML «requirement» element

As can be seen this is a simple graphical representation of the *requirement text* that allows us to draw graphical relationships with other model-elements. The language specification also allows for a «requirement» to have other properties, which may or may not be displayed in its graphical symbol as the modeller sees fit. The specification also allows for requirements (as well as other model elements) to be displayed in tabular form in a similar way to many specialist requirement tools. The following figure shows a tool-specific implementation:


Start | HSUV Requirement Table

#	ID	Name	Text	Requirement Type	Owner	Source	Risk	Verify Method
1	d.4	Power		Requirement [Class]	HSUV Requirements			
2	d.2	Range		Requirement [Class]	HSUV Requirements			
3	d.1	RegenerativeBi		Requirement [Class]	HSUV Requirements			
4	R1.2.1	Emissions	The vehicle	Requirement [Class]	" " Eco-Friendiness			
5	4.2	FuelCapacity		Requirement [Class]	" " Capacity			
6	4.1	CargoCapacity		Requirement [Class]	" " Capacity			
7	2.4	Acceleration	The Hybrid	Requirement [Class]	2 Performance			
8	2.3	OffRoadCapab	The Hybrid	Requirement [Class]	2 Performance			
9	2.2	FuelEconomy	The Hybrid	Requirement [Class]	2 Performance			
10	2.1	Braking	The Hybrid	Requirement [Class]	2 Performance			
11	2	Performance	The Hybrid	Requirement [Class]	HSUV Requirements			
12		SafetyTest		Requirement [Class]	" " Qualification			
13		Qualification		Requirement [Class]	HSUV Specification			
14		PowerSourceM		Requirement [Class]	HSUV Requirements			
15		PassengerCapa		Requirement [Class]	" " Capacity			
16		Ergonomics		Requirement [Class]	HSUV Specification			
17		Eco-Friendiness		Requirement [Class]	HSUV Specification			

Figure 4. A tool-specific SysML «requirement» table

Editing Requirements

If the system-model 'uses' requirements published from a requirements-catalogue then any changes made also have to be reflected there. This can be done by making changes in the original catalogue and waiting for them to be 'migrated' forward, which may take time and be inconvenient. Alternatively we may change them in the system-model and 'push' them 'back up' to the catalogue, which could result in a conflict if other changes have already been made there. In either case, as soon as changes are made in one tool, the data in the other tool becomes 'stale' until they are synchronised again.

Which tool for which task?

Faced with the situation where we have two tools with overlapping functionality and data, how do we decide which one to use for which SE activity? Should we record verification traceability in the catalogue, the model or both? If the data in the two tools is identical and continuously synced, or synced so frequently as to not cause a problem, then this is just a matter of which tool offers us the most appropriate functionality. But what happens if one of them has 'stale' data? Is it better to run a precise query on 'stale' data or an imprecise query on 'fresh' data? Neither choice is optimal.

Model-Based Requirements

A model-based requirements approach using a single tool does not have most of the issues associated with the hybrid approach outlined in the previous section. Using such an approach *requirements* and all other model-elements are 'authored' and 'used' within the same system-model. This prevents any delay, the need for synchronisation and since it is the only source of data it can never be 'stale'. We are also able to perform automated model checking on any model elements, at any point in their lifecycle.

Other SysML Concepts

As well as the «requirement» concept, SysML has other concepts which can be used to describe '*requirements*' (in a generic sense) in a non-textual form. Figure 5 illustrates how the *requirement* of Figure 1 implies structure and behaviour, and more importantly, how once these are modelled, how it enables us to assess the completeness and consistency of the requirements set.

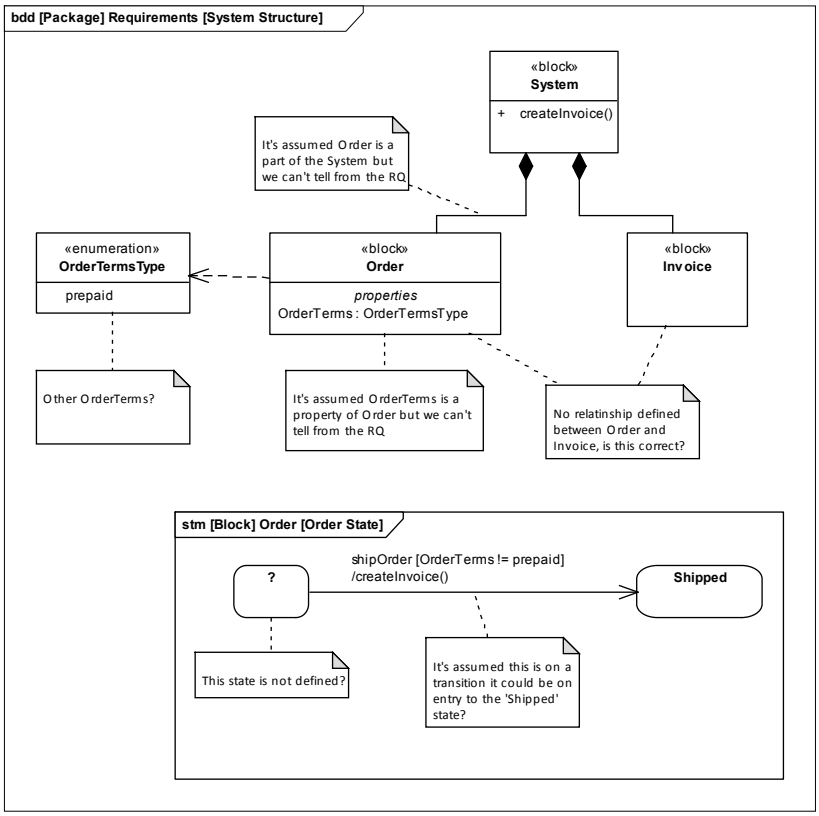


Figure 5. The implied structure and behaviour in a requirement.

The ACRE Approach

One approach to developing model-based requirements is the ACRE *(Approach to Context-based Requirements Engineering)* method first described by Holt, Perry and Brownsword in *Model Based Requirements Engineering* [4]. This approach incorporates an *Ontology*, *Framework* and set of *Views* to provide a rich set of concepts for the description and analysis of *model-based requirements*. Figure 5 illustrates an *ACRE Traceability View*.

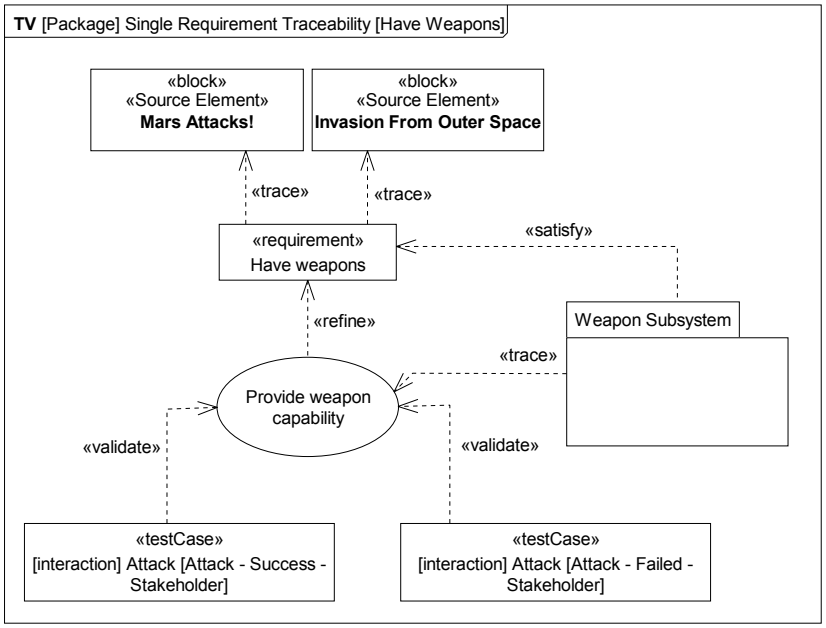


Figure 5 ACRE Traceability View for a single Requirement

Conclusions

We have explored some of the challenges faced when considering implementing a hybrid document-centric / model-based approach which is often regarded as a natural stepping-stone on the journey to MBSE implementation. Conversely it's difficult to identify any advantages that can be gained by 'authoring' text-based requirements outside of your preferred MBSE tool and subsequently importing them. Given the obvious increase in effort, cost and complexity required to use both a requirements-management and MBSE tool on the same project, it's hard to see any advantage in this approach.

For organisations with a less-mature requirements processes and/or no or little investment in specialist requirements-management tools the logical adoption pattern is to move straight to a model-based approach in a single step. This gives smaller and/or more flexible organisations a distinct advantage in the MBSE adoption race.

References

1. Mavin A., Wilkinson P., Harwood A., Novak M. *Easy Approach to Requirements Syntax (EARS)* ; Rolls-Royce PLC, Derby, UK; 2009
2. Towers, J. Editor. *"What is Model-Based Systems Engineering"*, INCOSE UK; 2015
3. Long D. & Scott Z. *A Primer for Model-Based Systems Engineering, 2nd Ed*. Blacksburg, VA, USA: Vitech Corporation; 2011
4. Holt J., Perry S., Brownsword M. *Model Based Requirements Engineering*. Stevenage, UK: IET Press; 2011

Biography

James has over 20 years experience in software and systems modelling. He has presented both papers and tutorials in model-based systems engineering at several conferences and seminars as well as writing and teaching commercial training courses on the subject. He has provided consultancy, training and mentoring to a wide range of organisations working in power electronics, telecommunications, rail, consumer electronics, information technology, finance, pharmaceuticals, retail and supply chain. He is a member of the IET, INCOSE, IIBA and chair of the INCOSE UK Model-Based Systems Engineering Working Group.