Using Relational Model Transformations to Reduce Complexity in SoS Requirements Traceability: Preliminary Investigation

Charles Dickerson
SEIC, Holywell Park
Loughborough University
Leicestershire, LE11 3TU, UK
C.Dickerson@lboro.ac.uk

Ricardo Valerdi
77 Massachusetts Avenue 41-205
Massachusetts Institute of Technology
Cambridge, MA, U.S.A.
rvalerdi@mit.edu

Abstract. *The principles and methods of Model Driven Architecture are applied to the problem of requirements traceability for a System of Systems (SoS). Model transformations of operational threads are used to reduce the complexity of modeling mission requirements and their flow into the architecture of the SoS. The allocation of requirements to operational mission threads (OMTs) rather than to individual systems reduces the complexity of the requirements tracing. Relational transformations provide a mathematically based formalism for model transformations that permit precise computation of the transformation of operational threads into threads of systems allocated from the SoS. Connectivity requirements for the SoS are also exposed in this way and the number of permissible system threads are seen to correspond directly to the number of permissible transformations. The principles and methods are illustrated by an elementary case study for sensor fusion.*

Keywords: Requirements engineering, system of systems, relational transformation, cost estimation.

1 Introduction

The Systems Engineering community is aggressively working on Model Based Systems Engineering (MBSE). Formal languages, methods and tools are being introduced to support MBSE. The Systems Modelling Language (SysML) introduces extensions to the Unified Modelling Language (UML) for the purpose of Systems Engineering. UML was specified for modelling software systems and has become ubiquitous in that community. Working groups in the OMG and INCOSE are developing methods and tools.

Over the past decade a new systems practice in software development called Model Driven Architecture (MDA) has emerged. MDA™ is now entering its second generation of specification by the Object Management Group (OMG). What new ideas are involved and how does MBSE relate to MDA™? How can they be extended to SE and System of Systems SE (SoSE)?

The objective of this paper is to apply the principles and methods of Model Driven Architecture and recent advances put forth in [1], [2], and [3] to the problem of SoS requirements traceability. The traditional systems engineering practice of definition and decomposition can be realized through model transformation (as presented in [2]), which has been given a mathematical basis in [1]. Specifically, the practice of model transformation in MDA™ can be realized using relationship-preserving transformations of the parameters of a model. The power of this approach is that traditional transformations of model parameters can be used to compute the implied transformation of relationships between those parameters.

In a traditional definition and decomposition approach to systems engineering, an allocation of requirements to individual operational activities and individual systems can lead to order of magnitude growth in the number of requirements specified. See for example [4]. And this does not account for the fact that unspecified dependencies or other relationships between system parameters, when discovered, will add further complexity. Recent research [3] has introduced an MDA approach using relational transformations to evaluate alternative SoS architectures against the particular requirements of a specific mission. The preservation of relationships between operational requirements through mathematically defined model transformations ensures that all relationships between SoS parameters implied by the requirements are properly captured. The use of OMTs for SoS requirements engineering simplifies this problem by means of meaningful groupings of operational activities and systems.

1.1 Model Driven Principles for SoSE

MBSE generally seeks to model (vice document in text) the basic attributes associated with a system. This is a move from Document Centric to Model Centric engineering. It should be accompanied by a shift from document centric storage and retrieval of information to a data centric paradigm. This will improve design quality, program management. MDA™ is the exemplar for software development.

The promise of OMG for MDA™ is to allow definition of machine readable application and data models which allow long term flexibility of Implementation, Integration, Maintenance, Testing and Simulation.

The precepts of traditional Systems Engineering design and development are often described by the Forsberg Systems Engineering Vee (see for example [5]), in which system specification is accomplished by means of definition and decomposition. The left side of the Vee is comprised of system concept and requirements analysis, system specification, configuration item specification, and finally 'build to' specification. The bottom of the Vee, which joins left and right hand sides, is implementation. A Model Driven Approach to Systems and SoS Engineering would view each level of decomposition as a model and focus on model transformation as central to system design and SoS assemblage.

One possible approach to applying MDA principles and methods to the traditional Systems Engineering Vee was presented in [2]. In this approach, the principal systems engineering activities on the left hand side of the Vee are realized through the principal MDA™ models: the Computational Independent Model (CIM), the Platform Independent Model (PIM), the Platform Specific Model (PSM), and the Platform Model (PM). These models are described in detail in the OMG MDA™ Guide [6]. Figure 1 illustrates how this approach could be adapted to a Systems Engineering Vee for SoS Engineering (SoSE).

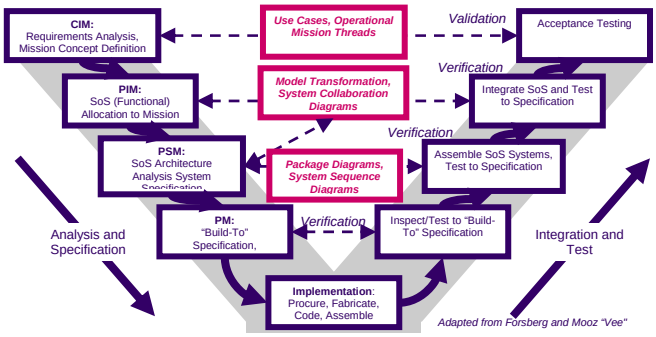

Figure 1. An MDA Approach for an SoSE Vee

At the CIM level, functional requirements are captured with Use Cases and the operational tasks for the mission concept are captured in OMTs. Strictly speaking, at the CIM level only the operational sequence portion of the OMT would be specified. In the DoD Architecture Framework (DoDAF) [7], this sequence is denoted as the OV-6c. The association of systems from the SoS with operational tasks could be captured parametrically in a PIM, which for example, could be described with generic system outputs or system functions. Through relational transformation of models, this allocation gives rise to collaboration relationships between the systems of the SoS. This is the first artifact of the SoS architecture. UML Package Diagrams or SysML Block Diagrams can be used to further model the structure of the SoS, which would be captured in the PSM. The PM can be used to capture the final specifications for the systems of the SoS and the integrating functions for their interoperability.

In this way, the SoS is assembled and integrated to support the mission concept. Operational capabilities are modeled through the OMTs which capture both the ways (e.g., Ov-6c) and means (e.g., systems from the SoS) by which the operational capabilities are realized. Strictly speaking, the SoS is not 'designed' but rather assembled and integrated. However, there could be modifications to the individual systems or additions to the SoS that are required for the systems to operate and interoperate for the purpose of delivering SoS level mission capabilities. These modifications are the subject of a more traditional system design approach. Early research and experiments in network centric warfare by the U.S. Navy [8] provide further elaboration of this concept for SoSE.

1.2 SoS Requirements Traceability

It is clear from the above discussion that tracing mission requirements through the SoS entails not just the specification of how individual systems must operate but also how the systems interoperate with each other to deliver mission level capabilities. Thus, the tracing of relationships through the definition and decomposition process of systems engineering is an integral part of SoS requirements traceability. Applying a mathematically based formalism using relational model transformations for this type of requirements traceability is the subject of the next section of the paper.

2 System Modeling Approaches

Just as Model Driven Architecture can be adapted to the Systems Engineering Vee [2], it can also be adapted to other aspects of traditional systems engineering such as requirements engineering and the Dependence Structure Matrix (DSM).

2.1 Requirements Engineering

The understanding, elaboration, and management of requirements is at the core of the systems engineering function [9]. The dependencies of requirements in an SoS is especially challenging because of the diverse set of stakeholders involved. Requirements traceability, in particular, becomes a complex task that calls for machine automation. Therefore a more structured approach that captures the relationships between requirements is necessary. The model driven approach, using relational transformation, can be used to address some of the complexities involved with SoS requirements engineering. Existing methods can be used to represent some of the complexities but they fall short in capturing the traceability of all the relationships that drive the system functionality.

2.2 Dependence Structure Matrix

An existing method for representing the dependencies between elements of a system is the Dependence Structure Matrix shown in Figure 2. This approach has been widely used to capture dependencies between system elements, which can take the form of components, processes, organizations, and activities [10]. The benefit of DSM is that it can help characterize the dependencies. In the example provided in Figure 2, there are 19 different dependencies between the elements A_1, … , A_7. One disadvantage of DSM is that it does not enable the traceability of requirements across multiple components of a system or system of systems. In other words, isolated relationships between system elements is inadequate to reason about the complexity of the dependencies.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
A_1		✓			✓	✓	
A_2				✓			✓
A_3		✓		✓			✓
A_4		✓	✓		✓		✓
A_5				✓		✓	
A_6	✓				✓		
A_7		✓	✓	✓			

Figure 2. Dependence structure matrix

3 Relational Transformation

Model transformation in MDA is the process of converting one model into another model of the same system. It is a central concept. An MDA mapping provides specifications for transformation of a PIM into a PSM for a particular platform. The platform model will determine the nature of the mapping. Model transformation can be thought of as part of the design and development process.

3.1 Origins

Despite the popular practice of model transformations in MDA™, a mathematical foundation has not yet been given in the OMG guidance and specifications for MDA™. The broader computer science community, on the other hand, has considered mathematical transformations on relational structures. A summary and assessment of a wide variety of these transformations based on universal algebra can be found in [11]. These transformations share the common concept of relationship-preserving functions on structures. Based on a comparison of these concepts and the concepts of model transformation in MDA™, reference [1] introduced a variant of the approaches used in [11] to define an easily computable definition of model transformation that is properly founded on mathematical logic and universal algebra.

3.2 Methodology

The relational transformation of a model is a natural concept because in mathematical logic, relational structures are the basis of models. See for example, [12]. In general, a transformation is specified only for the parameters used to model a system but it will induce a mapping between the relationships in any model of the parameters. The notation below provides a simple but rigorous method of calculating the transformation of relationships in a matrix notation.

In [1], the relational structure of a model is represented by a matrix which is designated by an underlined capital letter, such as $\underline{M}$. The underlying set of parameters are designated by a capital italics letter, e.g. $M = \{y_1, y_2, \ldots y_{m-1}, y_m\}$. A mathematical relation on the parameters M is denoted as a normal capital letter, for example $R = \{(y_2, y_{m-1}), (y_2, y_m), (y_{m-1}, y_2)\}$. The ordered pairs in R represent relationships between the parameters and correspond to the tick marks in the matrix $\underline{M}$, as illustrated in figure 3.

In the figure, Q transforms the parameters $y_1, \ldots y_m$ into a set of system parameters $x_1, \ldots x_n$ by associating one or more of the x_k with one or more of the y_i. We shall use the notation y_iQx_k to mean that Q has associated y_i with x_k. As with a binary mathematical relation, this means that the ordered pair (y_i, x_k) belongs to Q.

To see the implied transformation of the models we extend the notation yQx to the notation RQ to show how Q transforms a subset R of $\underline{M}$ into a subset RQ of $\underline{N}$. Recall that a subset R of $\underline{M}$, in general, represents a mathematical relation on the underlying set of parameters M. RQ is defined as follows: for each pair of parameters (y_i, y_j) that belong to the mathematical relation R, if y_iQx_k and y_jQx_l then the pair (x_k, x_l) belongs to the mathematical relation RQ in $\underline{N}$. If S is a relation in $\underline{N}$ and RQ is a subset of S, then Q preserves the relationships of $\underline{M}$ and it is called a *relational transformation* from $\underline{M}$ to $\underline{N}$. Although the transformation of relationships is easily calculated in this way, it is important to understand that this is not an ordinary calculation involving matrix sums or products.

Figure 3 illustrates a simple relational transformation. Although the association of parameters by the matrix Q is bidirectional, the calculation of RQ is regarded as a 'forward' transformation of the model $\underline{M}$ into the model $\underline{N}$. If S is the mathematical relation on N defining $\underline{N}$, then the 'reverse' transformation from $\underline{N}$ to $\underline{M}$ is denoted by QS and calculated from pairs (x_k, x_l) in S: given (x_k, x_l), if y_iQx_k and y_jQx_l then (y_i, y_j) belong to QS. Note that for the special case of S = RQ then QS = R.

If the parameters of the source model M represent system requirements, for example, then allocation by Q of the 2 requirements in M to the 4 specification parameters in N gives rise to 6 relationships between the parameters of N. Various analyses can also be performed. In figure 3, symmetries in the source or target models can represent concurrencies. In this illustration, the single concurrency in M is transformed into two concurrencies in N.

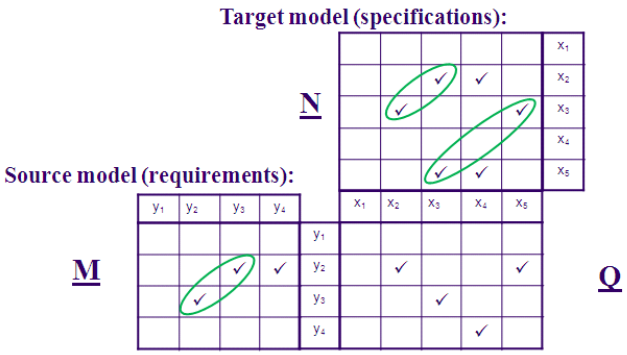

Figure 3. System design example using relational transformation

3.3 Operational Mission Threads (OMTs)

OMTs are generally regarded as operational sequences to which mission resources have been allocated. If the formalism in figure 3 is used to model OMTs, then the matrix M would be a model of operational tasks (e.g. an OV-6c), N a model of the SoS systems and interoperations, and Q would be an allocation of the systems to the tasks. In this formalism, figure 3 would be the model of an OMT. Thus, for fixed M, N, the permissible OMTs and systems threads for a mission are in direct correspondence with the permissible transformations.

3.4 Applications

The graphical representations of models used by UML and SysML are suitable for relational transformation because the transformation preserves graphs. And whenever key parameters such as cost, schedule and risk have dependencies amongst each other, transformations of the dependencies can be calculated.

4 SoS Requirements Case Study

Chapter 14 of reference [1] provides a case study of capabilities assessment for the introduction of a fusion node into the architecture of a sensor SoS. This case study can be used to provide a straight forward example of how OMTs and model transformations can be used to reduce the complexity of modeling mission requirements and their flow into the architecture of the SoS. Following the MDA approach described in the Introduction and depicted in figure 1, the CIM, PIM, and PSM models can be used to model the architecture, mission and capabilities of the SoS.

The SoS in the case study is comprised of four systems: a Micro-Internetted Unattended Ground Sensor (MIUGS), a Global Hawk configured with an imaging radar, a Predator configured with video imaging, and a sensor fusion node. Figure 4 displays a deployed MIUGS ground sensor. The Global Hawk and Predator are unmanned air vehicles. Details of these systems can be found in [1].

Figure 4. Micro-internetted unattended ground sensors

4.1 Concept Definition and Requirements

Following the scenario and case study in [1], the mission concept focuses on an operational capability to track a target for a required minimum of time (t_{min}). The target of interest in the scenario is a SCUD tractor erector launcher (TEL). But the TEL must not only be detected and located; it must also be identified correctly amongst an ensemble of about a dozen ground vehicles that escort it. Four key tasks (operational activities) are used to model the capability to track the target: cue, find, fix, and track. These tasks are commonly denoted as the activity sequence CF^2T. In this sequence each activity must be performed in order to the prior activity. The 'Cue' activity (A_1) is an alert to search a region for a target of interest. 'Find' (A_2) is the result of a successful search of the region and provides geolocated detections of possible targets. 'Fix' (A_3) in this model is the result of associating target identification with a geolocated detection. 'Track' (A_4) is the result of correlating repeated Fixes of the target of interest.

Figure 5 depicts the matrix representation of this simple model. In this model, the information output from A_i will be provided to A_{i+1} for i = 1, 2, 3. The tick marks in

the matrix indicate the precedence order of the tasks. Thus, this is a matrix representation of the OV-6c associated with the CF^{2}T activity sequence. A representation of the OV-6c as an operational flow would be: $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$. This contains the same information as the three ordered pairs (i.e. relationships) represented in the matrix. Techniques such as DSM can be used in more complex problems to re-order the indices so as to create simpler structures in the matrix representation of the operational activity model than might be realized in a first ordering of the indices.

A_1	A_2	A_3	A_4	
	✓			A_1
		✓		A_2
			✓	A_3
				A_4

Figure 5. Matrix representation of activity model

4.2 SoS Allocation to Mission

Allocation of the SoS systems to the tasks of the operational activity model can be accomplished by various methods such as functional allocation. The MIUGS, for example, is a networked field of sensors that can detect the movement of ground vehicles along roads. Denoting the MIUGS as S_1 and allocating it to support the Cueing task (A_1) is represented in the matrix as a tick mark in the cell (A_1, S_1). Denoting the Global Hawk and Predator as S_2 and S_3, these systems are allocated to Find (A_2) and Fix (A_3) based on their fields of view and ability to identify the target. Track (A_4) in this concept is allocated to the sensor fusion node (S_4). This allocation is associated with the SoS when it is in a state of target acquisition and its matrix representation is depicted in figure 6. When the SoS is in a state of track maintenance, cueing is no longer needed. The fusion node then permits the Predator (S_3) to also be used for the Find activity. Figure 7 depicts the matrix of SoS allocations for track maintenance. Thus, the source models and transformations in the OMTs for target acquisition and track maintenance are given by the OV-6c in figure 5 and the allocations in figures 6 and 7, respectively.

	S_1	S_2	S_3	S_4
A_1	✓			
A_2		✓		
A_3			✓	
A_4				✓

Figure 6. Matrix representation of SoS allocation for the state of target acquisition

4.3 SoS Architecture Analysis

Relational transformation of the matrix of the activity model (figure 5) can be calculated for the two states of the SoS using figures 6 and 7 as the transformation matrix. The different SoS allocations will give rise to different system threads, depending on the state of the SoS. For the state of target acquisition, using the matrix in figure 6, the thread (A_1, A_2), (A_2, A_3), (A_3, A_4) is transformed into a single thread (S_1, S_2), (S_2, S_3), (S_3, S_4). The system sequence in the system collaboration matrix is identical in form to the matrix in figure 5 (i.e. replace the A_i with S_i).

	S_2	S_3	S_4
A_2	✓	✓	
A_3		✓	
A_4			✓

Figure 7. Matrix representation of SoS allocation for the state of track maintenance

But this is not the case when the SoS is in a state of track maintenance. In this case, the operational thread represented in figure 5 is transformed (by the matrix in figure 7) into two system threads: (i) (S_2, S_3), (S_3, S_4) and (ii) (S_3, S_3), (S_3, S_4). The (S_3, S_3) collaboration is possible because the sensor fusion node permits the Predator (S_3) to both Find and Fix the target (which is under track). Figure 8 depicts the system collaborations and sequencing when the SoS is in a state of track maintenance. The system collaborations displayed in the matrix expose 3 requirements for SoS connectivity. The two loops in the figure depict the two distinct system threads.

	S_2	S_3	S_4
S_2		✓	
S_3		✓	✓
S_4			

Figure 8. Matrix representation of system collaboration and sequencing in the state of track maintenance

One significant implication of the relational transformation of the operational thread into the system collaboration and sequencing matrix is that it shows how a single thread at the operational level of the mission can be transformed into two system threads at the SoS level.

The requirement for an operational capability to track the target for a required minimum of time (t_{min}) would be flowed down to a performance requirement on each of or a

combination of both of the two system threads. The fusion node would be central to combining the two threads.

SoS architecture analysis would also investigate, for example, whether the two systems threads implied by the OMT are technically feasible. Each of the two threads reflects the Find then Fix then Track precedence from the operational activity model. But whether the collaborations (i.e. sharing of sensor data) are permissible depends on the specification of the SoS communications architecture.

4.4 SoS Specification

SoS specification in this case study would be primarily concerned with how the collaborating systems will communicate with each other [13]. It is important to remember that the system collaboration matrix (which identifies needs for exchanges between systems) should not be confused with the systems communications matrix (which identifies means for exchanges between systems).

In point to point communications architecture, each collaboration would be enabled by direct communication between the collaborating systems. In this case, the systems communications matrix would have the same form as the systems collaboration matrix. However, in net centric communications architecture, each collaboration would be enabled by direct communication with the network. In the sensor SoS case study, the fusion node can be used to implement a net centric communications architecture.

5 Conclusions

This preliminary investigation into the application of relational transformations and OMTs to reduce the complexity of SoS requirements traceability has provided both a framework and formalism for SoSE that builds on and extends MDA™. The case study for sensor fusion illustrates the application of the framework and formalism in a straightforward example that supports architecture analysis for system design and SoS assemblage. Precise computation of the transformation of operational threads into threads of systems allocated from the SoS efficiently and effectively exposes SoS connectivity requirements.

Because an OMT can be represented by a source model, a target model, and a relational transformation between the two, it becomes possible to use one set of requirements on one set of operational threads to govern the flow down of requirements to each permissible OMT.

References

[1] C.E. Dickerson and D.N. Mavris, *Architecture and Principles of Systems Engineering*, Auerbach Publications, New York, 2009.

[2] C.E. Dickerson, "A Review of Model-Based Methods for Systems Engineering", *Proceedings of the INCOSE UK Chapter Autumn Assembly*, 26 October 2009.

[3] P.W. Johnson, "Evaluating systems of systems against mission requirements," Ph.D. dissertation, Loughborough University, UK, 2010.

[4] R. Valerdi and M. Wheaton, "EIA ANSI/EIA 632 As a Standard WBS for COSYSMO," AIAA 1st Infotech@Aerospace Conference, Arlington, VA, September 2005.

[5] K. Forsberg, H. Mooz and H. Cotterman, *Visualizing Project Management*, 3rd edition, John Wiley and Sons, New York, NY, 2005.

[6] J. Miller and J. Mukerji, 2003. *MDA guide, version 1.0.1*. OMG. http://www.omg.org/docs/omg/03-06-01.pdf (accessed 8 August 2008).

[7] Department of Defense. *Department of Defense Architecture Framework (DoDAF), Volume I: Definitions and Guidelines, Volume II: Product Descriptions*. Washington, DC, 2007.

[8] C.E. Dickerson, et al. *Using Architectures for Research, Development, and Acquisition*. Defense Technical Information Center: (www.dtic.mil) *AD Number* ADA427961. Washington, DC, 2003.

[9] P.A. Laplante, *Requirements Engineering for Software and Systems*, CRC Press, Boca Raton, FL, 2009.

[10] M. Danilovic and B. Sandkull, "The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations," *International Journal of Project Management*, Vol. 23, No. 3, pp. 193-203, 2005.

[11] M. Walicki and M. Bialasik, "Categories of relational structures". *Lecture Notes in Computer Science*, Vol. 1376, Selected Papers on the 12th International Workshop on Recent Trends in Algebraic Development Techniques. Springer-Verlag: London, 1997.

[12] J.L Bell and A.B. Slomson, *Models and ultraproducts: an introduction*. North-Holland Publishing Company: Amsterdam, 1969.

[13] C.E. Dickerson, Defense Applications of SoS in Jamshidi, M., editor. *System of Systems Engineering: Principles and Applications*. CRC Press, Taylor and Francis Group: Boca Raton, Florida, November 2008.