Analyses Made to Order: Using Transformation to Rapidly Configure a Multidisciplinary Environment

Bjorn Cole[1]

[1] Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109.

ABSTRACT

Aerospace problems are highly multidisciplinary. Four or more major disciplines are involved in analyzing any particular vehicle. Moreover, the choice of implementation technology of various subsystems can lead to a change of leading domain or reformation of the driving equations. An excellent example is the change of expertise required to consider aircraft built from composite or metallic structures, or those propelled by chemical or electrical thrusters. Another example is in the major reconfiguration of handling and stability equations with different control surface configuration (e.g., canards, t-tail v four-post tail).

Combinatorial problems are also commonplace anytime that a major system is to be designed. If there are only 5 attributes of a design to consider with 4 different options, this is already 1024 options. Adding just 5 more dimensions to the study explodes the space to over one million. Even generous assumptions like the idea that only 10% of the combinations are physically feasible can only contain the problem for so long. To make matters worse, the simple number of combinations is only the beginning. Combining the issue of trade space size with the need to reformulate the design problem for many of the possibilities makes life exponentially more difficult.

Advances in software modeling approaches have led to the development of model-driven architecture. This approach uses the transformation of models into inferred models (e.g. inferred execution traces from state machines) or the skeletons for code generation. When the emphasis on transformation is applied to aerospace, it becomes possible to exploit redundancy in the information specified in multiple domain models into a unified system model. F[1]urther, it becomes possible to overcome the combinatorial nature of specifying integrated system behavior by manually combining the equations governing a given component technology. Transformations from a system specification combined with a system-analysis mapping specification enable one-click combination of domain analyses. This is a flexibility that has been missing from many engineering codes, which often entangle design specification and physical examination much more than is required to conduct the analysis.

This capability has been investigated and cultivated within the DARPA F6 program by a team of JPL and Phoenix Integration engineers building the Adapatable Systems Design and Analysis (ASDA) framework. By embracing system modeling with SysML and the Query-View-Transformation (QVT) language, the ASDA team has been able to build a flexible, easily reconfigurable framework for building up and solving large tradespaces. Examples of application and lessons learned in building the framework will be described in this paper. In addition, the motivation will be laid for various tool vendors to develop open model description standards while being able to maintain competitive advantage through proprietary algorithms and approaches. These standards will also be compared to the underpinnings of model-driven architecture and the OMG standards of the Meta-Object Facility (MOF), SysML, and QVT.

INTRODUCTION

This paper describes the results of extending the use of model transformation from the software domain into the systems engineering domain. Through the use of transformation techniques, multiple analyses can quickly be tied together according to the rules developed in a system model. This section will discuss background on these techniques and the rationale for their use.

The bulk of the work described is the direct product of a task performed for the DARPA F6 program. The DARPA F6 (Future, Fast, Flexible, Fractionated, Free-Flying Spacecraft United by Information Exchange) program is working to prototype and understand the use of fractionated spacecraft. The task undertaken at JPL, called the Adaptable Systems Design and Analysis tool, is intended to help DARPA validate the business model that has been used to justify and promote the idea of fractionation in spacecraft design and deployment. There is currently no flow between the ASDA team results and design constraints upon other F6 teams working to create an integrated flight demonstration. However, once the demonstration has proven the feasibility of fractionation, the intent is to use the ASDA tool to demonstrate utility and guide further optimization of new spacecraft.

Fractionation is the approach of taking the various functions of a spacecraft and splitting them up among multiple smaller spacecraft to achieve several benefits, including shorter development cycles and higher responsiveness to changing conditions. These spacecraft are typically also envisioned to be flying in a close proximity formation. For example, the task of downlinking observation data to a ground station may be allocated to one type of spacecraft, while functions associated with generating these data are allocated to another type. In another instance, a large radar aperture may be simulated by many small receivers kept within formation flight.

[1] 978-1-4673-1813-6/13/$31.00 ©2013 IEEE

Evaluating the merit of fractionation is not a straightforward problem. There are many inefficiencies introduced by fractionation, such as the need to launch highly redundant mass for multiple sets of structures, batteries, control electronics, etc. On the other hand, the relative simplicity of the individual spacecraft may reduce technical, cost, and schedule risk. Making many copies of smaller integrated circuits may be more inexpensive than doing one-offs of more exquisite designs. Yet, although the marginal cost drops, the total buy is almost always larger than buying an individual. Of course, it is also far easier to build redundancy and flexibility into a fractionated architecture than a unified one. But laying out all of the considerations still leaves the grand question: what are the real magnitudes of these different effects on a given fractionated deployment?

Answering this question via analysis or simulation requires some way of covering the whole space of possible combinations of functional assignments to different spacecraft and the resulting impacts on spacecraft requirements. While there may be clever mathematical approaches to make the problem more tractable, in most cases it looks like some fraction of the aforementioned combination space must be sampled and simulated. With typical engineering analysis frameworks, this rapidly becomes an impossible feat.

This paper presents the use of a technique called model transformation in order to keep the dimensionally of the problem posed here from growing too rapidly. Rather than dealing with a need to make analytical configurations in numbers of the order of the number of combinations, which grows exponentially, this technique grows in size more like the order of the number of individual choices to be made.

The goal in this paper is to acknowledge that compute time may still rise with the number of possible combinations, but to introduce a technique that can address the potentially large compute time. However, it is recognized that compute time is relatively inexpensive, while the human cost of manually building analyses to deal with the huge range of architectures and discrete combinations is a true problem. Here, we aim to use advanced development techniques, like model transformation, to reduce the amount of time human analysts must work to deal with alternative cases. This is not a magic bullet – the computer still must deal with the exponential growth in the trade space, but the human designer / analyst has the problem of specification relaxed to something more like O(n) from O(n^2).

BACKGROUND: MODEL TRANSFORMATION

The technique of model transformation is beginning to appear more often in the literature of systems, mechanical, and aerospace engineering. Before its penetration into new domains, the use of model transformation was primarily in the areas of software engineering. An example approach is that of the Open Modeling Group (OMG)'s Model-Driven Architecture (MDA) [1]. The crux of this approach is a transformation language called Query-View-Transformation (QVT) [2].

In the software world, transformation has been successfully used to turn data from one format into another, and to support the direct generation of software code. An example of the first use is the use of XSLT to convert data encoded in one XML schema to another encoding. Another example can be seen in UML tools that can directly create computer code.

For the purposes of other engineering domains, the promise of transformation is in the mapping of outputs or model specifications from one analysis tool into inputs for another. An example of this can be seen in previous work to go from a systems specification model into Simulink or Modelica to support dynamical simulation [3]. A key benefit of transformation is to render what has often been a weakness into a strength. Rather than having information in multiple files, transformation allows for the bootstrapping of information from one set of models into the construction of others. In the multiple file approach, there are as many opportunities for error as there are pairwise connections between redundant files. The redundancy of information between contributing analyses becomes a facility for reuse rather than this error source when transformation is used.

The key to defining a regular, computable transformation between models is to use metamodels to define mappings between model elements. This is illustrated in the diagram

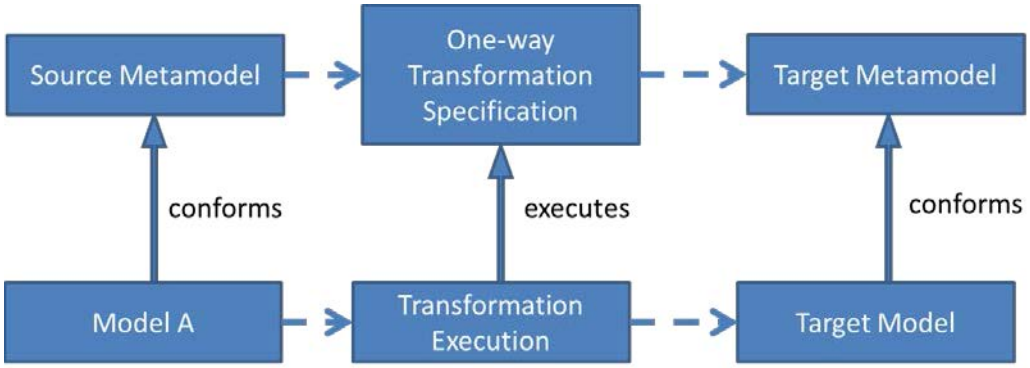

Figure 1. Core elements to perform a one-way transformation.

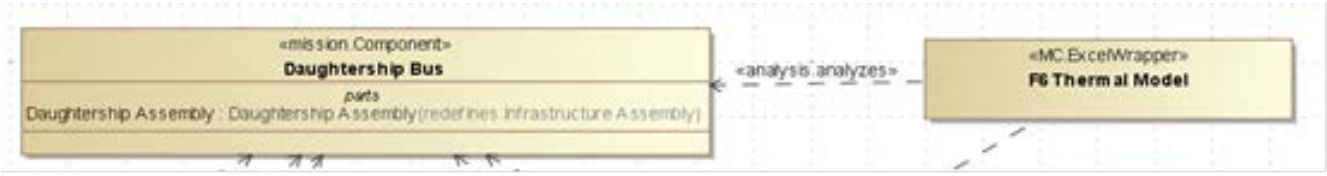

Figure 2. Mapping between Daughtership Bus and Thermal Model..

below, Figure 1. In it, there are two meta models with relationships established between them. A model A, conforming to metamodel MMA, can then be written as a new model B with conformance to metamodel MMB, or vice versa. The benefit of the extra level of indirection is that the transformation can be made generic. If many models are made according to the same metamodel (which may happen if many firms are submitting designs to DARPA eventually), then each can be transformed in the same way.

The final piece of this story is wrapped within the motivation for creating the Systems Modeling Language (SysML). The motivation behind SysML was to capture the definition of a system within a model rather than within static documents. This model would contain constraints upon implementations that are acceptable to the customer (requirements) and the structure of the system to be built, and its desired behaviors. However, there are not many aspects of analysis that are included in defining SysML. So rather than wanting for a grander language, the approach taken in this paper is to blend the best of SysML, specification, with the best of many other engineering tools, which is developing and executing models geared to analysis. Model transformation is the bridge between these two worlds.

It is worth emphasizing this final point. System specification (even if many constraints and equations for analytical models are specified) alone is not sufficient. Analytical models, no matter how powerful or how excellent the simulator, alone are not sufficient. Without a good data structure behind them, attempts to integrate analytical models becomes hard to track and understand. This is why tools like Simulink and Modelica solvers are very popular in engineering work - it is much easier to understand an analysis when contextualized directly by configuration information. And of course a specification of structure and behaviors by itself does not provide the predictive capabilities engineers require. Thus both are important.

APPROACH: TRANSLATING SYSTEM SPECIFICATION INTO ANALYSIS SPECIFICATION

As stated before, the driving problem in defining an analysis tool for the DARPA F6 is the sheer number of possibilities that must be considered, driven by the combinatorial nature of sizing and simulating multiple, interacting spacecraft in multiple potential system architectures. In order to get some traction on this problem, it must be possible to reduce the number of configurations a human designer must develop while still maintaining a comprehensive set of design alternatives.

The strategy for the ASDA team has been to leverage three powerful tools: SysML, model transformation, and Phoenix Integration's ModelCenter software. ModelCenter is a tool that has brought integration between multiple engineering tools into the commercial market. In addition, it provides a series of tools for doing systematic exploration of parametric trade spaces, such as the Design of Experiments and surrogate modeling. The question from the beginning was how to extend this level of sophistication in dealing with parametric studies into dealing with combinatorial ones.

The key pattern to enabling this exploration is to truly separate analysis from specification, but then to leave pointers between the two. This is illustrated in Figure 2. In Figure 2, there are separate descriptions of a Daughtership Bus and the Thermal Model that will be used to analyze it. This means that the Thermal Model will be invoked any time that a Daughtership Bus or a bus that is specialized from it is used in a studied cluster.

In Figure 3, a generic type of cluster element, in this case a spacecraft, is defined to be connected to a given analysis. The diagram shows the path that is traced to apply this pattern to any specializations of the defined component. Thus, a spacecraft of type Daughtership, Mapper, Imager, Mothership, or F6 Enabled Spacecraft will all introduce this module into the analytical problem. The engineer is then free to specify as many spacecraft as desired, confident that the appropriate analyses will find their way into the integral problem.

If there are multiple types of spacecraft, say a daughtership (a client spacecraft that can take observations but not downlink) and a mothership (a spacecraft that provides ground station links for client spacecraft), then there will be multiple copies of the analysis in ModelCenter. Each of these copies provides a place for different inputs to lead to different outputs. The core of this approach is to utilize the concept of inheritance for associating generic cluster elements to different types of analysis.

Another piece of the pattern is to capture the relationships between analyses, if they happen to be instantiated in an integrated analysis. These relationships are captured in SysML using standard Parametric diagrams as shown in Figure 4. If a particular analysis on either side of the connections happens to not be instantiated in the integrated

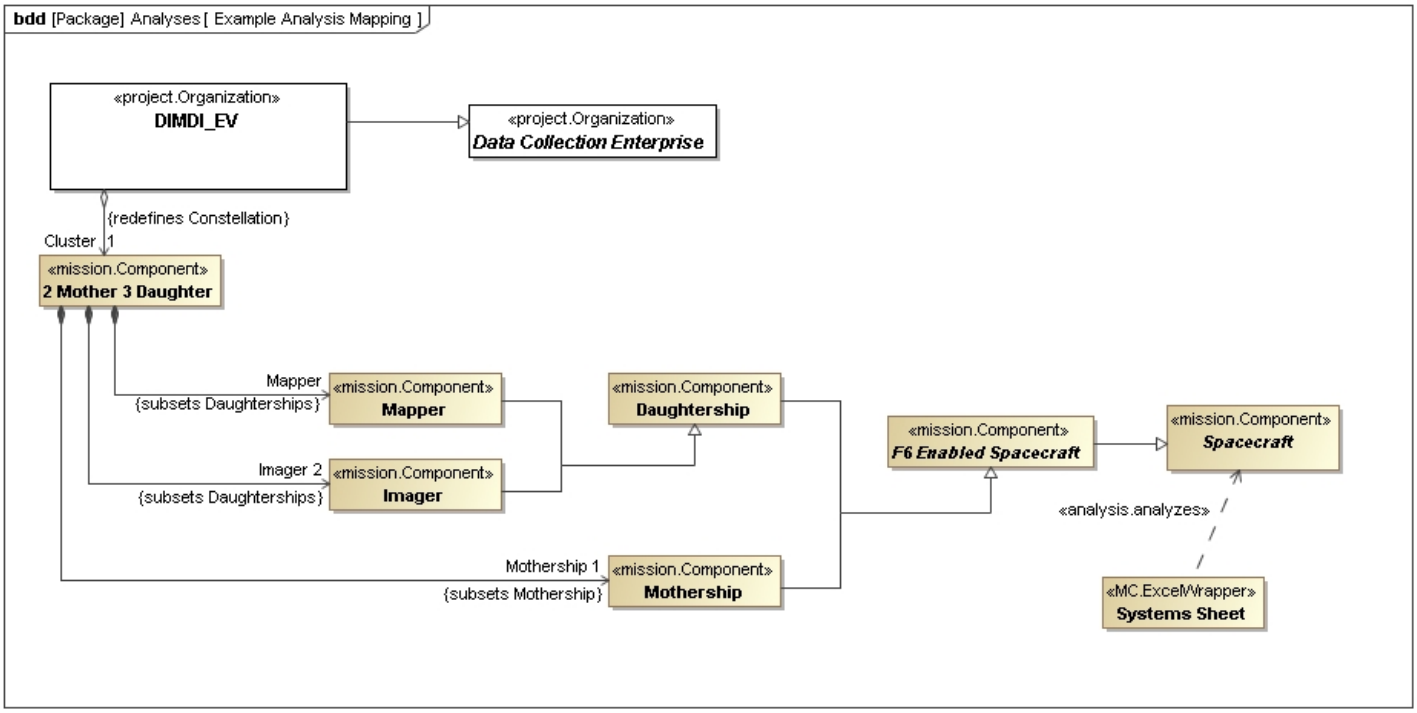

Figure 3. Inheritance path from generic description to specific type for application of analysis.

analysis, the link is not made and the parameter becomes a user-defined input value instead. Since the SysML Binding Connector is bi-directional, directionality is implied by the use of an Information Flow item of type "DirectionBlock." parameter connections.

The system description is specified in SysML. The analysis set is captured and connected in ModelCenter. The bridge between the two is a custom QVT transformation that takes a UML model and renders it into an XML file that ModelCenter can recognize as one of its own model files. This transformation carefully counts the number of copies of each type of analysis that is required by the specification of the cluster (how many of what type of spacecraft), and how they will need to interchange parameters. This transformation greatly reduces the number of connections a person must specify (only defining the connection between parameters) and leverages those connections for many cluster analysis configurations.

One of the chores of the transformation is to deal with how the cardinality of source and target analyses must be reconciled. The issue arises because of the strategy taken to deploy analyses into ModelCenter. If there are two types of spacecraft, taking the mothership / daughtership example, then an analysis on each of these types will likely require different parameters. This is facilitated by making multiple instances of the same analysis type in the ModelCenter model. Multiple cases of connecting parameters from multiple instances of the same analysis type to a single analysis instance of a given type are illustrated in Figure 5.

In general, the maps between parameter sets that must be accommodated are one-to-one mappings, many-to-one, one-to-many, and many-to-many. Multiple strategies are used, including the mapping between collections (e.g., many copies of a scalar value assigned to indices in an array), combination operators (e.g., the mass summation or "roll-up") or some other deterministic mapping. One rule for mapping that has been applied is to assign an index to individual spacecraft that is used to keep vectors of parameters properly aligned. This is shown in Figure 6 with the notation label of "Launch Order," although in the SimPy implementation, it more closely approximates the order in which different spacecraft are ordered to be built (longer builds would make this no longer the launch order).

By combining a disciplined, object-oriented approach to

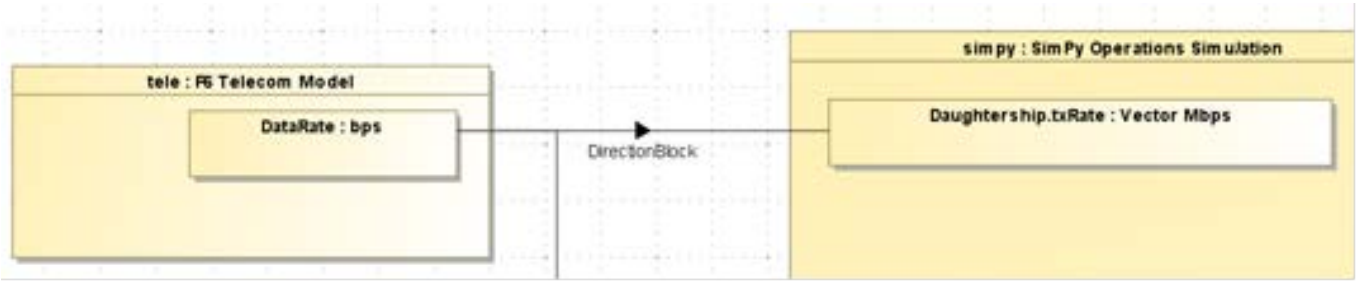

Figure 4. Parametric connections in SysML.

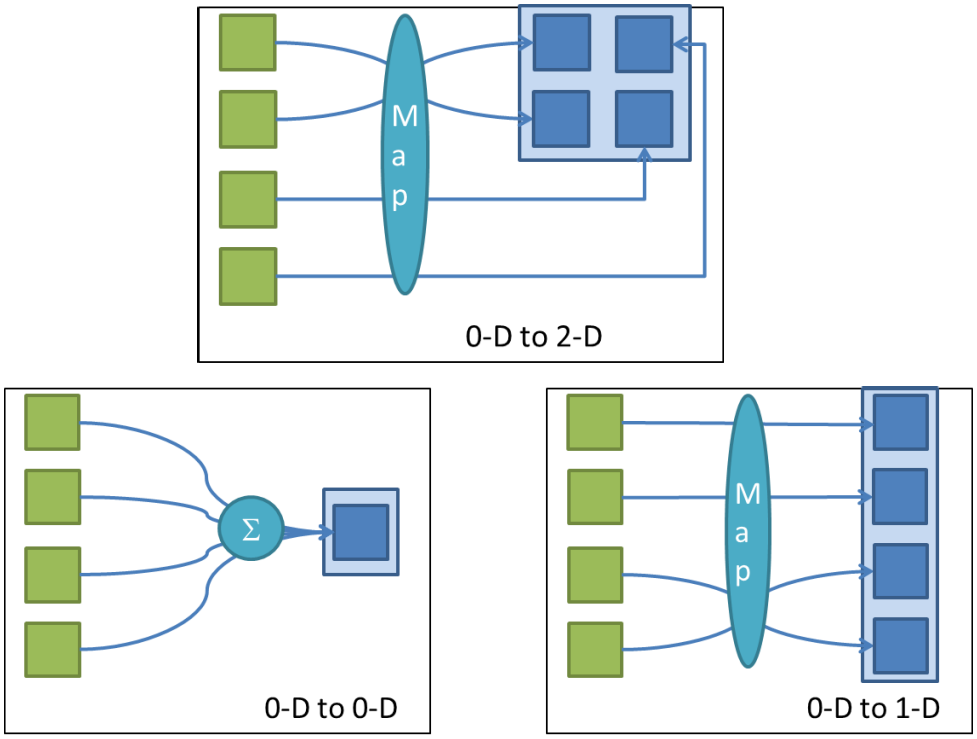

Figure 5. Multiple cases for parameter mapping.

instantiating analyses from the system specification and the resulting connections required between parameters, the ASDA code base provides a powerful platform for computing on combinatorial problems.

APPROACH: INTEGRATION AT MULTIPLE LEVELS

A variety of insights have been gained about model transformation via work on the F6 ASDA tool. The essence of them is about the relationship between a system's specification and the requirements of its analysis. An underspecified system cannot be analyzed, and it is important to be careful not to add aspects to a system's specification purely to support analysis. Doing so will make the analyses brittle and the specification muddled with extra information.

As shown below in Figure 6, a common idiom in the Integrated Model-Centric Engineering (IMCE) patterns below is the mapping between components of a system and their intended functions. While certainly not a one-to-one mapping, there is often a correlation between the decomposition of functions and the way that a system is broken up into constituent components.

For initial sizing, most analyses typically assume some basic functional use. For example, a solid state recorder is usually called upon to store data. So sizing models include parameters such as instrument data rates and downlink opportunity frequency to estimate required storage. These functions are not currently called out in the SysML ASDA model; the implicit connection between component and function in the analysis via parameters is used. A more specialized function of the solid-state recorder may be to serve as working memory for the spacecraft flight computer. In this case, the function may apply a need for new parameters in the analysis, such as latency in memory reads and writes. In this case, the function would be called out explicitly in the model, with the analysis description bound to the connection between the two.

An important consideration for this approach is the difference between a traditional use of a component, and a non-traditional one. For example, a pair of solar panels can be used to generate power (the traditional function), but can also be used to generate a solar pressure torque (a non-traditional or even deleterious function). Analyses that do not consider this repurposing have the potential to ignore important effects, or to encode their definitions in parameters that are hard to map to these alternative cases.

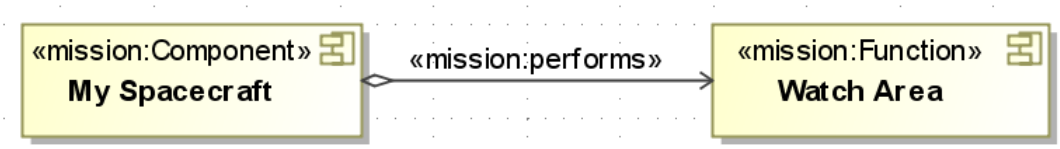

Figure 6. "Component performs Function" in SysML.

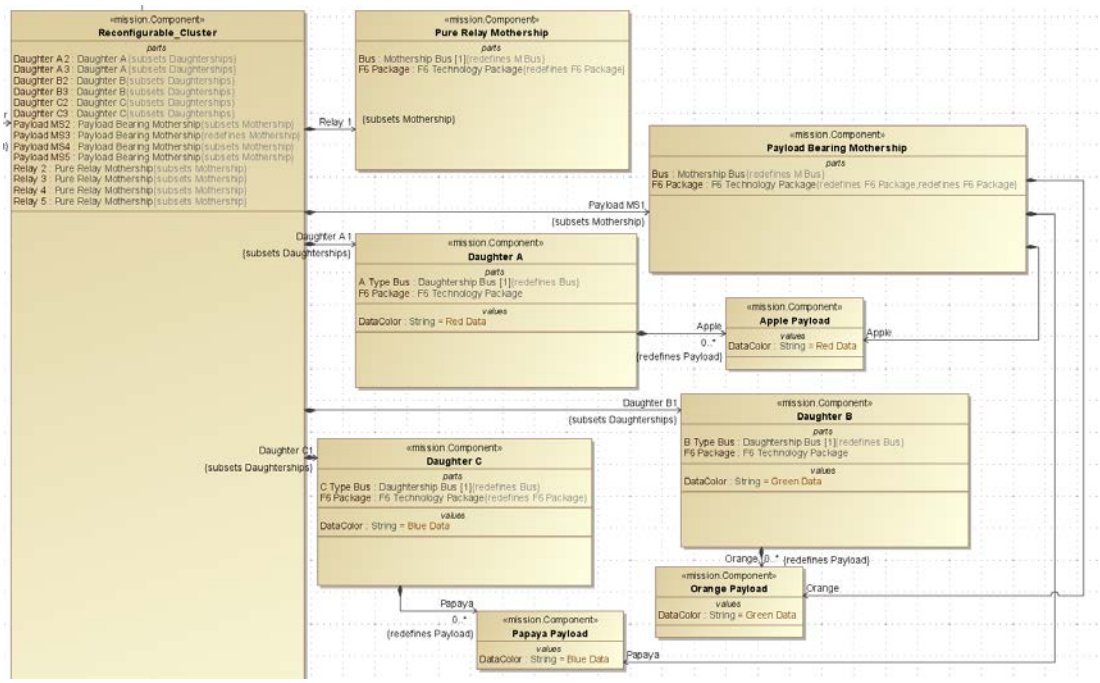

Figure 7. SysML cluster representation concept.

On the other hand, one can consider the discrete event simulation that is used to simulate the cluster functioning. Here, the roles of individual spacecraft are open to be assigned. The mothership-daughtership approach that is the team's main example is defined by what motherships and daughterships do. There are predefined patterns of relationships between them.

The two examples have an important difference. For the sizing models, it was sufficient to make a ModelCenter analysis copy for each type of spacecraft to be sized. These copies would then have general links between their parameters by the transformation. On the other hand, the discrete event simulator needs additional information on how different spacecraft types interact. There is only one copy made of the discrete event simulator in ModelCenter because it simulates the whole cluster, not just a spacecraft. Links between parameters on individual spacecraft analyses must be linked to the integrated analysis. But, it is also important to create the behavioral links and expected connections between spacecraft within the description of the scenario to be simulated.

The links within the scenario imply a new requirement for analyses of interactions between multiple elements that are also analyzed individually. In programs such as Spice (a circuit simulator) or Simulink, the connections are specified in a data structure called a net list. What this means is that

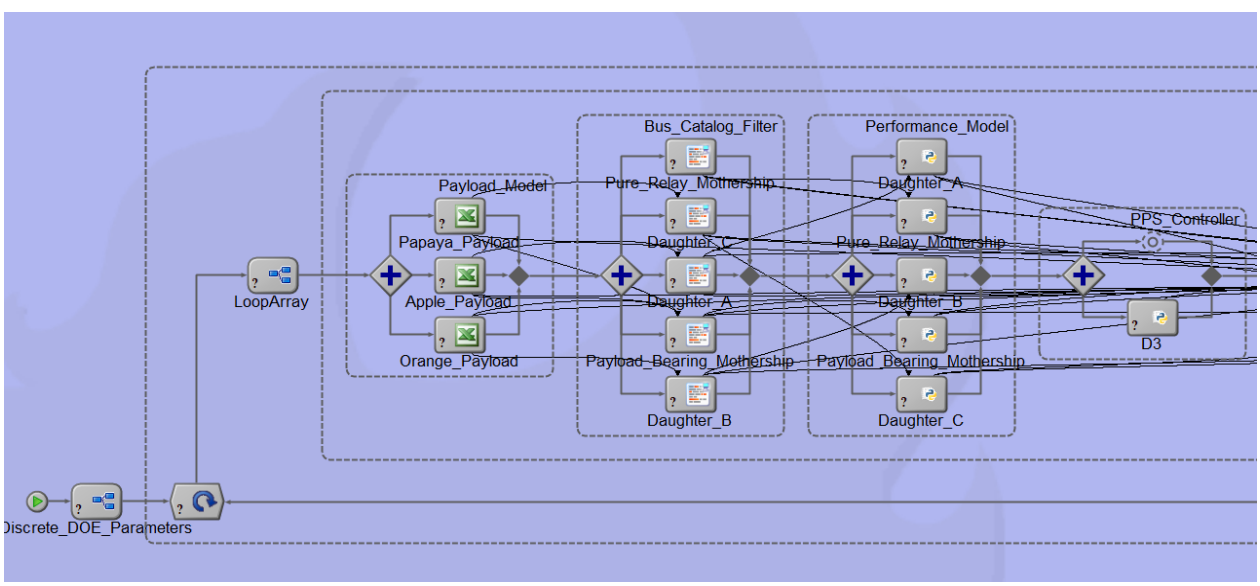

Figure 8. ModelCenter integrated analysis excerpt.

the transformation process must be allowed to not only specify connections between ModelCenter components but also to specify (or ideally, enhance, since the user may want to encode connections by hand before the transformation) the net list of the analysis input file. Manipulating parameter and analytical connections together allows for the complete specification of an integrated analysis from individual parts via transformation from a specification.

RESULT: EXTENSION OF MODELCENTER DRIVERS INTO COMBINATION SPACE

As a result of the ability of transformations to quickly make new combinations of analyses from cluster definitions, developers at Phoenix Integration have created a new ASDA tool. The ASDA tool serves as a front-end for defining parameter values and the order of execution of the multiple models created from the transformations written by the JPL portion of the team. This tool is a direct analog of tooling that ModelCenter has offered for investigating the affects of design parameter choices on system outcomes, except that it deals with choices in cluster topology. Different topological choices are made by looking up different input files generated by the transformation machinery.

The connection between the SysML configuration and ModelCenter can be seen in Figures 7 and 8. A set of SysML Blocks are shown as elements of a cluster in Figure 7. In Figure 8 is the deployment of this cluster topology into an integrated analysis workflow in ModelCenter.

The benefits of the transformation can be seen when considering the time needed to encode a new cluster type. For a cluster of 5 spacecraft (three different types), roughly 150 links are created. For clusters with 5 types of spacecraft, more than 600 links were created. To rebuild this set of connections by hand would be a very time-consuming, and likely error-prone, ordeal. With the transformation apparatus, roughly half an hour is required to define the new cluster configuration and begin generating alternatives. This was recently proven out in configuring a five-spacecraft cluster into a 19-spacecraft cluster in rapid fashion.

Another acceleration of rework for extending or improving the model was made apparent when adding new types of analysis to the model. Most of the way through the first Option Period of the ASDA task, individual subsystem models were put aside in favor of an analogy-based approach to finding performance values from spacecraft bus catalogs. Only the patterns for interconnecting between the spacecraft performance and the rest of the simulation were affected. This allowed the team to focus on interpreting the new models properly in their new context, rather than chasing as many missed connections as might have been necessary in a more literal approach to analysis integration.

As mentioned before, the ASDA tool approaches the traversal of combination space by providing a driver for loading and executing multiple integrated analysis specifications. The transformation apparatus makes it feasible to generate these combinations with sufficient speed to properly populate the ASDA tool with configuration files.

Another ability enabled by the framework is the ability to rapidly shift between sets of analyses to be integrated for a given cluster topology. The SysML model contains a simple table that shows an analysis name, its location in the Analysis Server, and an "ignore" flag. If there are multiple analyses allocated to a given system aspect (or may simply be analyses of different fidelities), one can be chosen as representative while others are simply ignored. The sensitivity of results can be evaluated not just with respect to cluster topologies, but also with respect to favored analyses. This is a powerful capability for examining business models.

- As the ASDA tool is applied to System F6, the transformation has shown itself as an enabler for multiple features of the tool:Rapid change in the basis of analysis by turning “on” one set of analyses and turning “off” another
- Rapid inclusion of new definitions of cluster elements
- Rapid (manual) generation of clusters with different sets of elements and types of spacecraft, payload, etc. considered.

As more of the ASDA toolchain is stood up, the transformation in this paper will become part of a pipeline that smoothly moves from the definition of clusters and dimensions of topology to change to integrated analysis.

RESULT: FORCING FUNCTION TO INCREASED ANALYTICAL REGULARITY

It has been observed in work on developing surrogate models that building these models exercises legacy analysis codes in ways never imagined. The process of developing surrogate models becomes a *de facto* unit test of each analysis, exercising it in a variety of corner cases.

The same is true with this approach to building up integrated analyses. A variety of unexpected connections and interplays creep into the framework. A great deal of effort has gone into the end of the first Option Period for the F6 task to ask the question "do these trends and effects make sense?" For the discrete event simulation at the core of the integrated framework, plotting and data capture tools had to be developed in order to run various impacts and effects to ground.

What this together means is that a framework that can drive so many instances and configurations of analyses requires a robust testing framework to go with it. While the ASDA team has developed unit tests for multiple analyses, more may be required. The transformation framework itself provides the ability to do investigations on a number of n-wise combinations. In fact, this capability was used to

quickly gain confidence in the use of analogy-based spacecraft analysis tools at the end of the first option period.

A testing framework using these tools together looks like the following. During the development of a given analysis, unit tests are handmade to assure that individual calculations and groups of calculations are being performed properly. Then, single instances of these analyses can be run within Designs of Experiments (DOE) in ModelCenter for through checking. Then, pair- and n-wise sets of analyses are created in ModelCenter framework using transformation with certain analyses ignored. These sets of analyses can also be examined with a DOE. Finally, the full set of analyses are generated and both architectural and parameters spaces can be explored for verification. Validation against known data for different individual spacecraft is also feasible.

The complicated interactions between multiple spacecraft within the discrete event simulator also drove a need to carefully log and curate runs from the simulation. A textual log was driven to become a post-execution series of plots, which in turn was made into a capability to write execution traces for later examination in bulk. This plot capability has become an important part of the post simulation workflow to gain an application for the dynamics of the problem.

A post simulation approach would look like the following. When many runs of different configurations are made, aggregate plots similar to those in the Data Visualizer are used to find general trends. Then time series for data, power, and so forth are examined to understand how a given scenario unfolded. Finally, individual parameters for a given design case are examined to further understand the results.

SUMMARY

The ASDA team has now been working with model transformation to build up its tooling to perform integrated analyses for over a year. It is been found to provide a powerful lever for dealing with the rapid-growing nature of combinatorial problems in a real, time- and budget-limited task. Now that there is a stable transformation codebase for ASDA to work from, adapting analyses to different design topologies can't be done in a fairly rapid fashion.

The transformation approach to integrating analysis around changing system configurations is proving fruitful in regularizing large design problems. In addition, it provides a convenient platform to stand up integrated analysis testing. As mentioned before, this has led to a reduction in the time required to adapt to new analysis sets or system cluster configurations. It has also pointed the way to needing an equally regular testing suite to assuring the quality of the results. This technique also helps to augment and guide that testing suite.

In all, model transformation is a core part of the ASDA approach and has proven itself a key to working with engineering problems that have large combinatorial trade spaces.

ACKNOWLEDGEMENTS

This work would not be possible without the efforts of the Phoenix Integration team to define their application's metamodel in an accessible way. Also, the approach to modeling transformation, as well as useful tooling, have been provided by Nicolas Rouquette at JPL. The original IMCE profile describes the separation of analysis and specification, which the author used to develop the approach to the transformations presented in this paper.

Work performed on this effort was carried out the Jet Propulsion Laboratory, California Institute of Technology, under contract to the Defense Advanced Research Projects Agency.

REFERENCES

[1] Truyen, F., "The Fast Guide to Model-Driven Architecture." OMG Website. Accessed Oct 23 2012. [http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf]

[2] "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification." OMG Specification, version 1.1.

[3] Cole, B., Chung, S.H., "Getting a cohesive answer from a common start: Scalable multidisciplinary analysis through transformation of a systems model," *Aerospace Conference, 2012 IEEE*, 3-10 March 2012.

BIOGRAPHIES

Bjorn Cole. Bjorn Cole is a systems engineer in the Mission Systems Concepts section of the Jet Propulsion Laboratory. His research interests are in the fields of design space exploration, visualization, multidisciplinary analysis and optimization, concept formulation, architectural design methods, technology planning, and more recently, model-based systems engineering. His most recent body of work concerns the infusion of systems modeling as a data structure into multidisciplinary analysis and architectural characterization. He earned his Ph.D. and M.S. degrees in Aerospace Engineering at the Georgia Institute of Technology and his B.S. in Aeronautics and Astronautics at the University of Washington.