AFRL-RI-RS-TR-2023-003

FOUNDATIONAL AGENT-BASED SYSTEMS TECHNOLOGY

88SOLUTIONS CORPORATION

JANUARY 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the AFRL Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2023-003 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
TODD B. HOWLETT
Work Unit Manager

/ S /
JAMES S. PERRETTA
Deputy Chief
Information Warfare Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
JANUARY 2023	FINAL TECHNICAL REPORT	AUGUST 2019	AUGUST 2022

4. TITLE AND SUBTITLE
FOUNDATIONAL AGENT-BASED SYSTEMS TECHNOLOGY

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-19-C-0091	N/A	DOD, NAVY
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		R2SS

6. AUTHOR(S)
Manfred R. Koethe

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
88solutions Corporation
10224 Fairhill Drive
Spring Valley CA 91977

8.PERFORMING ORGANIZATIONREPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/RIGC
525 Brooks Road
Rome NY 13441-4505

10.SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11.SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RI-RS-TR-2023-003

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# AFRL-2022-5951.
Date Cleared: 16 Dec 2022

13. **SUPPLEMENTARY NOTES**

14. ABSTRACT

The theme of the project was to explore new modeling methodologies and model-based software production techniques to improve the quality of produced software, while at the same time shortening development times and improve reusability of designs. A good design starts with good decision-making. We designed and implemented a Decision Modeling capability as a loadable plugin for the MagicDraw Unified Modeling Language (UML) modeling tool (a.k.a. Cameo). The Decision Modeler supports the Object Management Group (OMG) Decision Modeling and Notation (DMN) Decision Requirements Diagram, but implements a more sophisticated Decision Definition capability, compared to DMN. Active Decision elements can be analyzed using the built-in Monte Carlo Analysis, and decision models can be executed within the limits of the Cameo Simulation Toolkit. Co-Simulation with external simulation systems is supported via the SimCom protocol (also a MagicDraw plugin). Successful co-simulations between the Decision Modeler and the Advanced Framework for Simulation, Integration and Modeling (AFSIM) tool were also demonstrated.

15. SUBJECT TERMS
Decision Modeling, Digital Engineering, Systems Modeling, Systems Engineering, Mission Engineering, Modeling and Simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	31

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
TODD B. HOWLETT	N/A

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Table of Contents

List of Figures

Acknowledgements

The FAST project team want to take this opportunity to thank our sponsors, and in particular the Air Force Research Laboratory for the strong and continuous support of our work.

1 SUMMARY

Project "FAST" (Foundational Agent-based Systems Technology) was a three-year research and development effort under contract with the Air Force Research Lab in Rome, NY. The project was from the beginning sponsored by the US Navy, with additional contributions from the US Air Force later in the project.

The theme of the project was to explore new modeling methodologies and model-based software production techniques to improve the quality of produced software, while at the same time shortening development times and improve reusability of designs. There is a good amount of similarity between model-driven software and systems development, and what the Navy calls mission engineering. Traditionally both processes grow in a bottom-up fashion, while a top-down approach is more focused and promising. The initial step of this top-down approach must be a conceptual model outlining all (conceptual) decisions required to reach the desired goal based on a set of given facts. For mission engineering, this decision model helps to identify all key decisions to be made, and corresponding tasks to perform, for planning and executing a successful mission. For the software or systems engineer, the decision model outlines the key semantics, and corresponding structure, of the system under design. While modeling support for systems engineering, and to some extent for software engineering, exists, no tool support to build a decision model as a fully integrated part of a system or software design model exists. Consequently, due to its root in the Systems Modeling Language (SysML), mission engineering also lacks a decision modeling capability.

We narrowed this gap through the development of a Decision Modeler capability for the Unified Modeling Language (UML) modeling tool MagicDraw (aka Cameo) as a loadable plugin, compatible with commercially available plugins, like SysML, UAF, etc. The Decision Modeler implements an enhanced variant of the decision requirements diagram defined by the Decision Model and Notation OMG standard, but otherwise deviates from the OMG standard to provide more sophisticated decision expression modeling, decision simulation capabilities, and the ability of seamless integration with SysML (v1.x). For collaboration in larger simulation scenarios, we developed a second plugin (named SimCom) for MagicDraw that allows real-time collaboration between the simulation capability of the Decision Modeler and external simulation systems, like the Advanced Framework for Simulation, Integration and Modeling (AFSIM). The SimCom plugin implements a light-weight protocol inspired by the "High Level Architecture" (HLA) simulation protocol. We demonstrated this capability with a collaborative scenario between Decision Modeler and AFSIM.

While the work on the Decision Modeler took us away from the original plan to develop a temporal and spatial (4D), pattern-based modeling methodology, our intensive collaboration and contribution to the SysML v2, submission to the OMG compensated this to a good extent. SysML v2 has reached similar 4D characteristics to what we had originally planned for. It would be an interesting and beneficial task to rebuild the Decision Modeler in the SysML v2 environment. Besides the participation in the SysML v2 effort, the project had a leading role in several other OMG standard developments.

2 INTRODUCTION

Project FAST started with a kick-off meeting on 14 August 2019, presenting our ideas to make software and system development more effective, while making the produced product more reliable and secure at the same time. We presented our planned approach to make intensive use of Elemental Design Pattern (EDP) and Micro-Components. This presentation spawned the request to look into modernization of software used on ship computers using our model-based approach. We started work on a parser to read target programs and convert them into an equivalent Knowledge Discovery Metamodel (KDM), from which, after some model-driven rework and optimization, a modernized implementation could be generated. However, the work came to an early end since we could not be granted access to the needed source code.

Following this, we were asked to look specifically into requirements handling for mission engineering, and related decision-making processes. Requirements, as they are understood in the context of mission engineering, are very different from the regular understanding of requirements in an engineering context. In engineering, or even very general in the process of creating a product, requirements consist of a set of data and constraints, representing the wishes and needs of the product's customers and stakeholders. Consequently, the product to be designed and created by the engineers, shall then satisfy these requirements. The requirements in this case establish an information flow from the customers and stakeholders (the information providers) towards the engineers (the information consumers), the authoritative and informative dataflow has therefore the same direction.

In mission engineering, as we learned, requirements represent the need for information to support the planning (and execution of these plans). The need and usage of information is the known, the source of the information is the unknown, or at least uncertain. In contrast to engineering (where the source of the information is known, but its usage uncertain), the authoritative direction is therefore opposite to the information flow direction for requirements in mission engineering. Regular business planning applications, and even the predominant business process modeling system, the Business Process Definition and Model (BPDM) OMG standard, are not well suited to handle this kind of requirements scenarios. What is needed is a modeling methodology working top-down from a final goal, recursively detailing the needs ("requirements") to support this final goal. The result is a tree-shaped model with the final goal at the top, and expanding down into an increasingly expanding network of decision elements and supporting facts (input data).

While such a Decision Model is designed top-down, it is eventually then executed bottom-up. This is in contrast to business process models, where model execution flows typically in the same direction as the model design.

Decision points in business process models (for example designed using BPDM) are typically buried as "innocent" and small diamond-shaped elements in an otherwise complex and often convoluted diagram. However, these little diamonds carry in most cases much stronger semantics than the rest of the model. The Object Management Group (OMG) created a decision modeling standard, Decision Model and Notation (DMN) to support modeling of the decision semantics. Unfortunately, DMN is very tightly coupled to BPMN and its internal structure, which prevents a direct integration of DMN models with systems engineering

models or other general-purpose models. Consequently, there is also no tooling available that supports Decision Models in the context and environment of SysML or UML models.

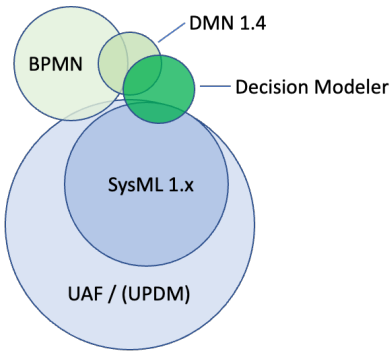

Figure 1 - The Decision Modeler in Relation to other Modeling Technologies

It was the main activity of the FAST project, to extend the modeling methodology of DMN into an open Decision Modeling methodology, which allows seamless integration with UML, SysML and United Architecture Framework (UAF) models, and which supports also direct model execution as a development and test capability. We completed this modeling methodology and implemented a prototype tool for Decision Modeling and execution as loadable plugin for the MagicDraw (aka Cameo) modeling tool.

This document will make frequent references to the FAST Project Technical Report (CDRL A010), which provides an in-depth description of the decision modeling methodology implemented by the Decision Modeler, detailed descriptions how to create, analyze and execute Decision Models, and how to construct and execute co-simulation scenarios based on the Decision Modeler and the associated SimCom protocol.

While the development of the Decision Modeling methodology became the primary task during the FAST project, we contributed to numerous related standardization projects in the OMG, including the upcoming SysML v2 modeling language. In several projects we played the leading role. Please see sections 4 and 5 of the companion report “Project Technical Report” (CDRL A010) for details.

3 METHODS, ASSUMPTIONS, and PROCEDURES

3.1 Relevant Standards and Technologies

Many modeling methods include implicit or embedded decision making. Examples of these are Flow Charts, Activity Diagrams, Business Process Models, and more. Decision modeling as a dedicated discipline is relatively new. The Object Management Group created the Decision Modeling and Notation (DMN) specification originally to make decisions in BPMN[1] business process models more visible, and to support a more detailed decision-making process. The downside of this history is, that DMN is now very tightly related to BPMN, in particular on a metamodel level. As a consequence, DMN, the way it is specified, is not directly integratable with UML or SysML. To make DMN-style decision modeling cooperative with UML and SysML models, and in particular to make existing UML modeling tools capable to do DMN-style decision modeling, a Decision Modeling UML Profile, closely resembling the DMN metamodel, must be created.

3.2 Target Modeling Environment

The development and target deployment platform for the Decision Modeler is MagicDraw version 19.0 SP4. MagicDraw (also known as Cameo) is a UML modeling tool developed and marketed by No Magic, Inc. No Magic had recently been acquired by Dassault Systèmes, which will continue further development and marketing of this tool, possibly under a different name.

MagicDraw is a UML modeling tool implemented in Java. It supports and uses a plugin architecture to extend its modeling capabilities, covering other UML-based modeling languages and methodologies, like SysML, UAF, and others. An OpenAPI toolkit is available to support the development of custom plugins.

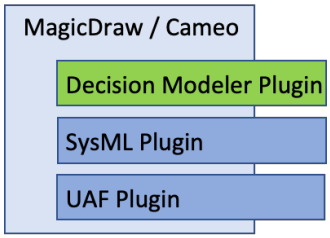

Figure 2 - The Decision Modeler within the MagicDraw (a.k.a. Cameo) Environment

The Decision Modeler and the SimCom communication engine are two custom plugins for MagicDraw developed by the FAST project. The SimCom plugin has no prerequisite requirements, while the Decision Modeler plugin requires the presence of the SysML and Alf plugins for its function. Since UAF is based on SysML, the Decision Modeler can also be used in UAF-based enterprise models. Both, the Decision Modeler and the SimCom plugin have installers consistent with the MagicDraw Resource Manager.

[1] BPMN : Business Process Model and Notation, an OMG specification.

3.3 Standards Development

While the technology developed throughout the FAST project was a best effort toward compliance with relevant standards, like the OMG specifications Meta Object Facility (MOF), Unified Modeling Language (UML), Systems Engineering Modeling Language (SysML), and others; or like the Web Ontology Language (OWL), Resource Descriptor Framework (RDF), or others, developed by the World-Wide-Web Consortium (W3C), we have ourselves been significantly involved in the development of new standards, namely within the Object Management Group.

While standards development is tedious work, it pays back with many benefits. Standardization of a certain subject requires it to be at the frontline of development of that subject. That work is often carried out in the seclusion of research labs or advanced development departments. Standardization then requires developers to open up and discuss the subject with like-minded peers around the world, which in all cases we know of was beneficial.

During the time of the FAST project, we participated in several standardization tasks of the Object Management Group. All these tasks had started at some time before the FAST project, but the ongoing work and discussions on these tasks provided significant input and scientific benefits to the FAST project. The tasks we were involved in are: MOF to RDF Transformation, Metamodel Extension Facility, Systems Engineering Modeling Language version 2, Agent and Event Metamodel, and Precise Semantics for Uncertainty Modeling. See also chapter Standardization Activities later in this document, and the corresponding chapter in the Project Technical Report (CDRL A010).

4 RESULTS and DISCUSSION

4.1 Decision Modeling Methodology

Decision Models look very similar to "regular" structural models on the first glance, like an average UML model, where a whole "thing"[2] contains multiple parts through composite relationships, or is related to other "things" via regular associations. While there is a visual similarity, there is a significant semantical difference. Decision Models always form directed graphs, usually trees. The principal direction in this directed graph is bottom-up, from the leaves to the top of the graph. Semantically, this expresses that the final result ("top goal") relies on facts and decisions made on those facts further downstream in the graph. If such a model is then executed, information streams flow independently bottom-up through every branch of the graph, until they reach the top goal element. Technically, this is a massive-parallel dataflow execution, which behaves very different from a "traditional" von Neuman computer architecture.

UML as a modeling language is not very well suited to express this kind of model correctly and sufficiently. Key elements missing from UML are any notion of Time, and therefore any form of timed information flows. These deficits affect models and modeled systems mostly at execution time, while the need for (and lack of) dynamic type adaptation along information flows complicates model creation.

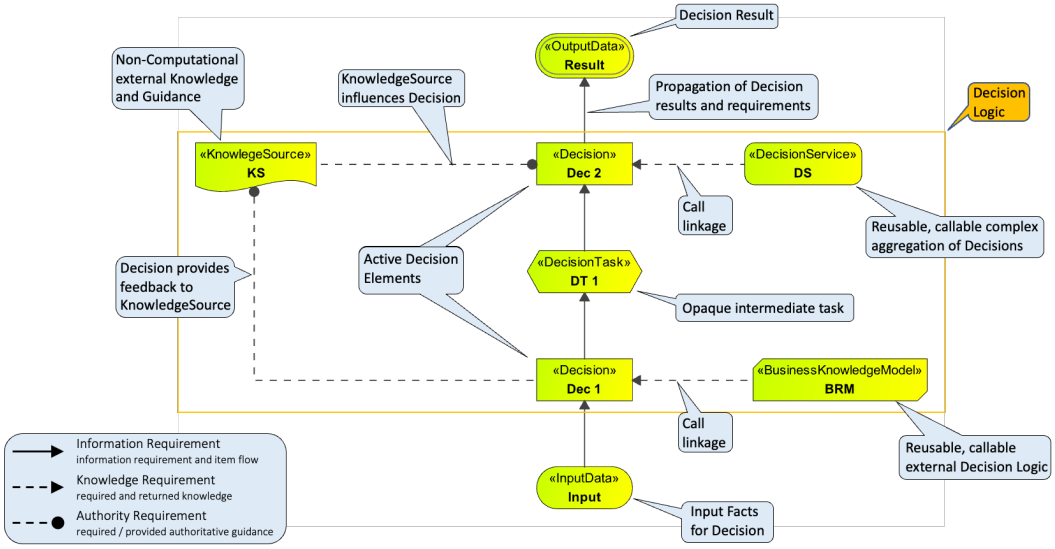

Figure 3 - Decision Model Elements

Decision Models are composed from two categories of elements: active and passive elements.

- Active elements are decision-making elements, which affect the overall outcome of the model result (top-goal value) during model execution. Per the OMG DMN specification, these active elements are defined: `Decision`, `DecisionService` and `BusinessKnowledgeModel`.

[2] Ontology terminology : "thing" is the standardized designator for the most general type or element

- Passive elements do not contain any decision logic, therefore do not directly influence the model results. They may be required to assist information flow between consecutive active elements, or to annotate the Decision Model. The OMG DMN specification defines the following two passive elements: `InputData` and `KnowledgeSource`.

All elements of our Decision Model, active and passive, have the same basic structure: They will take one to many inputs, called "input facts", and produce one single output, called "result fact". All facts may be single values or complex values, in which case they are structures of single values.

With the design of the FAST Decision Model definition, our intention was to stay as close as possible to the OMG DMN specification. However, a number of deviations were necessary:

- The DMN definition of `DecisionService` is not well defined in the standard. We have therefore not included `DecisionService` in our model. This is not a limitation, a single `DecisionService` is equivalent to a set of connected `Decision` elements and can therefore be replaced by such a set without any loss of semantics. If desired, `DecisionService` could be added to our Decision Model definition later, when the DMN specification has been sufficiently clarified.
- `InputData` elements could benefit from a transformation capability, which would allow data harmonization within the Decision Model. This transformation capability is foreseen in the metamodel, but not yet implemented in the Decision Model definition and runtime infrastructure.
- The DMN specification lacks any ability to connect results of a Decision Model to other model elements outside of the Decision Model. We introduced the `OutputData` element as a passive access point that could become the source of a value binding, as for example through a `BindingConnector` in SysML.
- Often there is the need of intermediate processing along the information flow from one active Decision Model element to another. To support this, we added a new passive Decision Model element `DecisionTask`. This element provides a kind of container for intermediate processing, which *does not* affect the decision-making directly. For that reason, we originally named this element `OpaqueTask`, but that name caused objections from our sponsors.
- The OMG DMN specification introduces a new expression language "FEEL"[3], which is a unique language in the sense, that it is different from, and incompatible with, any other known expression or action language in the UML modeling domain, or within OMG specifications in general. In particular, it is incompatible with execution semantics of executable UML (fUML[4]) and the Action Language for UML (Alf). Besides this, FEEL requires a specific graphical environment called "boxed expression". We removed FEEL completely from our Decision Model definition and replaced it with Alf.

[3] FEEL : Friendly-Enough Expression Language
[4] fUML : Semantics of a Foundational Subset for Executable UML Models (fUML)

Core of the active Decision Model elements `Decision` and `BusinessKnowledgeModel` is the decision expression. In general, this could be a simple expression, or a decision table, where every row constitutes a decision expression. In the literature (and in the DMN specification), decision tables can be horizontally or vertically organized. However, the organization is pure notational and does not influence the decision table semantics, therefore our Decision Model supports only horizontal organization[5], where every row of the decision table contains one decision expression. The expression (row) to be evaluated is selected top-down in the table, and by the conditionals contained in each expression. The arguments driving the selection of a particular expression (row) are the collection of input facts presented to the `Decision` or `BusinessKnowledgeModel` element. Expressions (rows) following the selected row will not be evaluated. Every decision table must contain a default expression to cover the case where no expression matches the presented input facts.

Decision and BusinessKnowledgeModel elements are mostly identical. The difference is the way they are placed in the Decision Model, and how they are invoked during Decision Model execution. `Decision` elements are primary members (nodes) of the Decision Model graph, receiving their input facts from prior `Decision` and/or `InputData` elements in the graph. They will propagate their result facts to a `Decision` or `OutputData` element higher up in the graph. BusinessKnowledgeModel elements, however, are sharable elements, each containing a single and pure[6] decision expression or decision table. The evaluation of the expression contained in a `BusinessKnowledgeModel` is called during, and out of, the expression evaluation of a `Decision` or other `BusinessKnowledegModel` element. The evaluation results from the `BusinessProcessModel` expression are directly returned to the point of call.

4.2 Decision Model Implementation in MagicDraw

The Decision Model and Notation (DMN) OMG specification defines DMN as a metamodel, with limited conformance to MOF[7] and the UML metamodel, and incompatible with any UML tool platform, including MagicDraw. Our Decision Modeling implementation overcomes this limitation and adds some new notations and new semantics. Despite these differences, the appearances and semantics of the model elements in our Decision Model resemble the semantics of the original DMN elements defined in the DMN Decision Requirements Diagram (DRD) as close as possible.

[5] The rationale for this was that vertically oriented decision tables present the contained expressions less clear, and, in addition, are difficult to edit interactively within the limited screen area available.
[6] pure : these elements *do not* retain any state, their execution is therefore re-entrant
[7] MOF : Metaobject Facility, the metamodeling language used to define the OMG modeling specifications

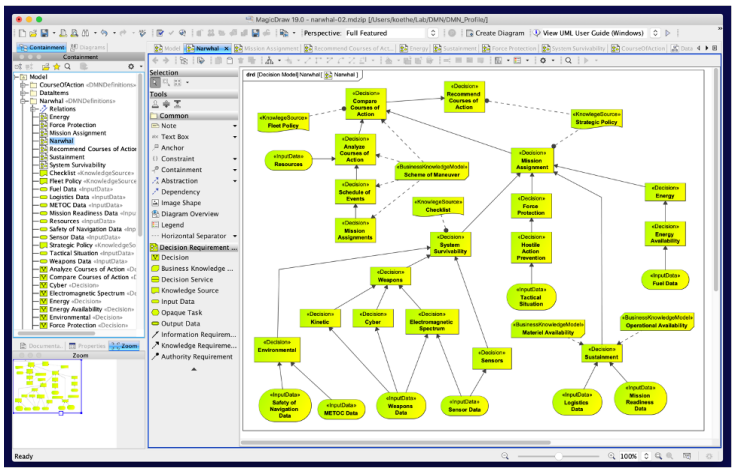

Figure 4 - A Decision Requirements Diagram loaded into the Decision Modeler running within MagicDraw

In order to introduce the decision modeling capabilities into an UML environment, a UML Profile equivalent to the DMN metamodel must be created. UML provides the Stereotype mechanism to create new metaclasses within the Profile. However, this comes with serious limitations: UML requires that a Profile application to the UML metamodel is reversible at any time, which means it must be possible to remove all effects and extensions introduced by the Profile without altering or damaging the original metamodel. As a consequence of this requirement:

- Stereotypes can only extend metaclasses already existing in the UML metamodel, Stereotypes cannot create new metaclasses that are unrelated to the UML metamodel.
- Stereotypes can only add new features to existing metaclasses, they cannot take features away that are already existing in the extended UML metaclassses.
- The ability of defining Associations is severely limited for Stereotypes. This comes from the requirement that the resulting links must be removable during Profile un-application without causing any effects on the original metamodel.
- A complication for implementers is, that an instantiation of a "stereotyped metaclass" at any point in the user model creates an instantiation of the original metaclass at that point, with a separate instantiation of the Stereotype created somewhere else in the model, linked to the metaclass instance to be extended through an Extension link.

MagicDraw provides a customization capability that softens some of the Profile/Stereotype limitations, and offers some additional processing rules. The customizations are actually special Stereotypes residing in a separate Profile, and are linked to the original Stereotypes. We make significant use of this customization capability.

One of these customization capabilities is the association of completely new and fully functional model symbols with a particular Stereotype (UML originally allows only attaching a static icon to a Stereotype). We use this capability for the representation of the new Decision Model symbols. The non-connector Decision Model symbols are derived from the definition of the UML Class symbols. Their shape and graphical interaction in a diagram are then altered to fit their specific role and semantics in a Decision Model through individual Java code for each symbol, using the MagicDraw OpenAPI and custom rendering capability.

The three connector elements use rendering variations offered natively by MagicDraw, but still need Java programming for implementing their specific semantics. This includes the connection rules for each connector, and the related smart manipulator menus of the connectable Decision Model elements.

The Decision Modeler provides five new diagrams to create and analyze Decision Models. These diagrams, and their usage, are in detail described in the companion report "FAST Project Technical Report" (CDRL A010). Here we provide information related to the diagram implementation. The Diagrams are:

- The Decision Requirements Diagram (DRD). This diagram supports the creation of the Decision Requirements Model, also known as the Decision Model Graph.
- The Decision Definition Diagram (DDD). A separate DDD exists for, and is subordinate to, each `Decision` and `BusinessKnowledgeModel` element. The DDD is used to define the decision expression logic of the element it is associated with.
- The Result Table Diagram presents the result(s) of Monte Carlo Analysis performed on a particular `Decision` or `BusinessKnowledgeModel` element. It is related to that particular element and exists only after a Monte Carlo Analysis had been performed. It will be deleted when the simulation environment for that particular element is re-initialized.
- Result and Condition Chart. This diagram is also specific to a particular `Decision` or `BusinessKnowledgeModel` element, and is only available after a Monte Carlo Analysis had been performed on that element. The diagram supports a more detailed exploration of analysis results, and experimentation to show the potential effect of altered analysis constraints.
- Status Overview Diagram. The Decision Modeler provides a capability to perform user or application controllable element annotations in the Decision Requirements Diagram. The Status Overview Diagram provides list and a statistic summary about these annotations.

From the five diagrams listed above, only the Decision Requirements Diagram could be implemented using the original diagram definition capabilities provided by MagicDraw, in combination with Stereotype customizations. The remaining four diagrams are completely implemented through Java programming.

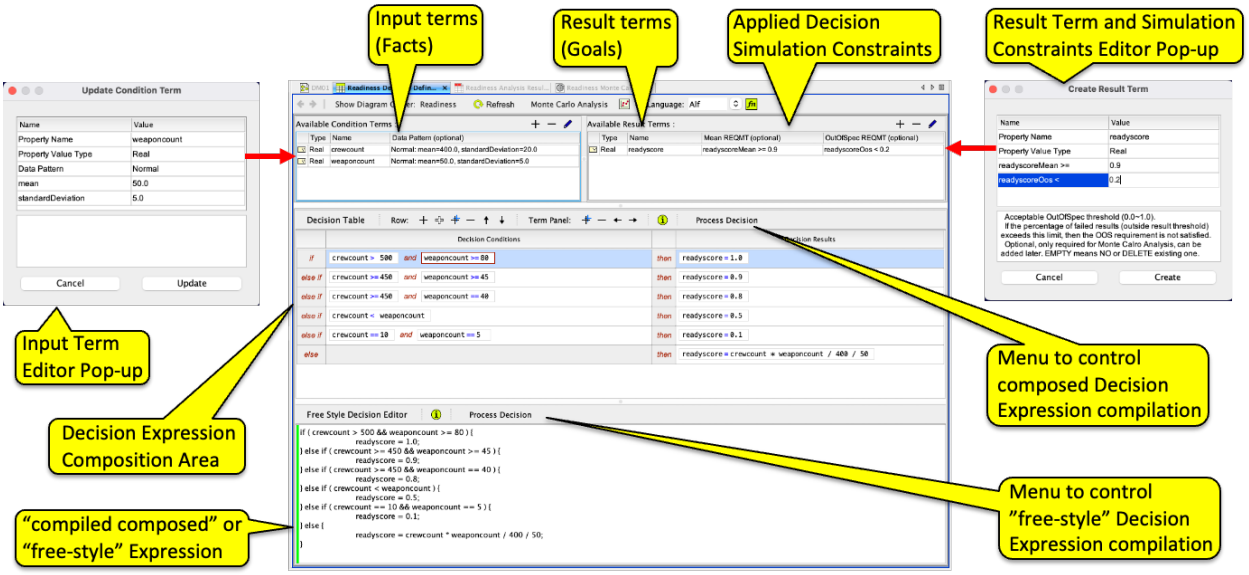

Figure 5 - The Decision Definition Diagram

The Decision Definition Diagram is the central and most complex part of the Decision Modeler. As visible on the surface, it implements an interactive decision table and decision expression editor. Decision tables are composed row-by-row, where each row consists of a condition expression and a result expression. Evaluation of the condition expression determines the selection of the decision table row. Evaluation of the result expression of the selected row produces the corresponding result fact value of the decision table for the presented input fact values. Automatic pre-population of expression fragments, also called expression panels, based on provided input and result facts provides the user with a fast start. All fragments can be modified using in-place menus or through direct editing. Fragments can be removed and new fragments added. A library provides a broad selection of functions, allowing the construction of more complex decision expressions. All user input is immediately validated on the spot. If errors or invalid inputs are discovered, the background color of that fragment changes from white to red.

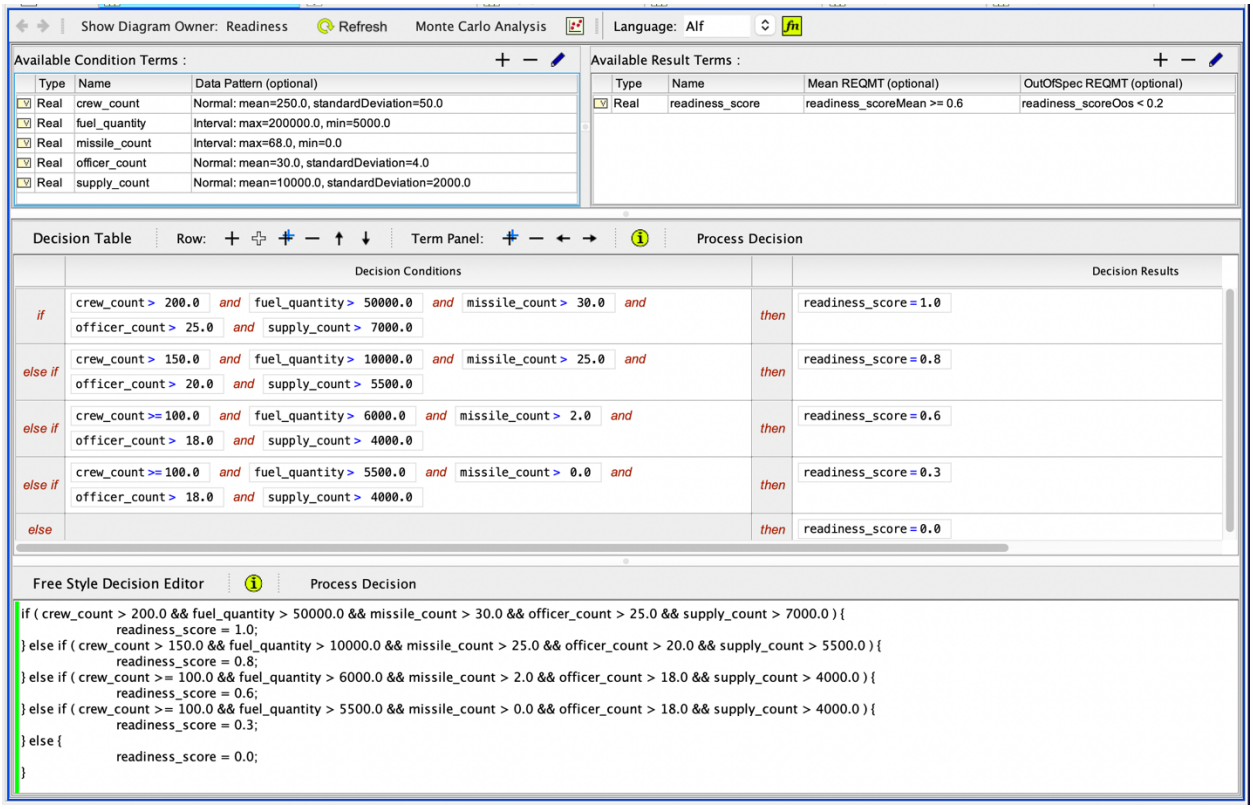

Figure 6 - The Decision Definition Diagram for the Readiness Decision of the No-Fly Zone Scenario

The expression editor supports two language syntaxes: Alf and JavaScript. All expression fragments use the universal mathematical syntax common to all modern programming languages, including Alf and JavaScript. Function syntax, however, differs between the two languages. The function library offers therefore all functions in an Alf and a JavaScript version. The selection is automatic, based on the language preference set by the user. If the user changes his/her mind and wants to switch from Alf to JavaScript, or vice versa, the decision expression parser underlying the decision table editor provides an automated expression re-writing algorithm, translating the whole decision table from Alf to JavaScript syntax, or from JavaScript to Alf.

The user can store the decision table at any point in the editing process, and pick it back up later to continue the editing process. When the user is satisfied with the decision table, it needs to be compiled into an executable `OpaqueExpression`[8], using a simple menu button.

Advanced users can write the decision expression directly, using the selected syntax, in the Freestyle Expression editor provided in the lower part of the Decision Definition Diagram. Input is directly interpreted by the selected language processor (Alf or JavaScript). Errors are flagged with red underlining. The Freestyle editor can be loaded from the interactive decision table editor and vice versa.

[8] OpaqueExpression is a standardized UML Behavior element.

4.3 Model Analysis and Execution

The exact execution behavior of `Decision` and `BusinessKnowledgeModel` elements containing complex decision tables in response to wide ranges of input fact values is hard to estimate in all details and possibilities by a human user or developer. This is the situation where automated simulation can become an enormous help. The Decision Modeler offers Monte Carlo Analysis for decision tables (or simple decision expressions), if input facts can be annotated with distribution properties.

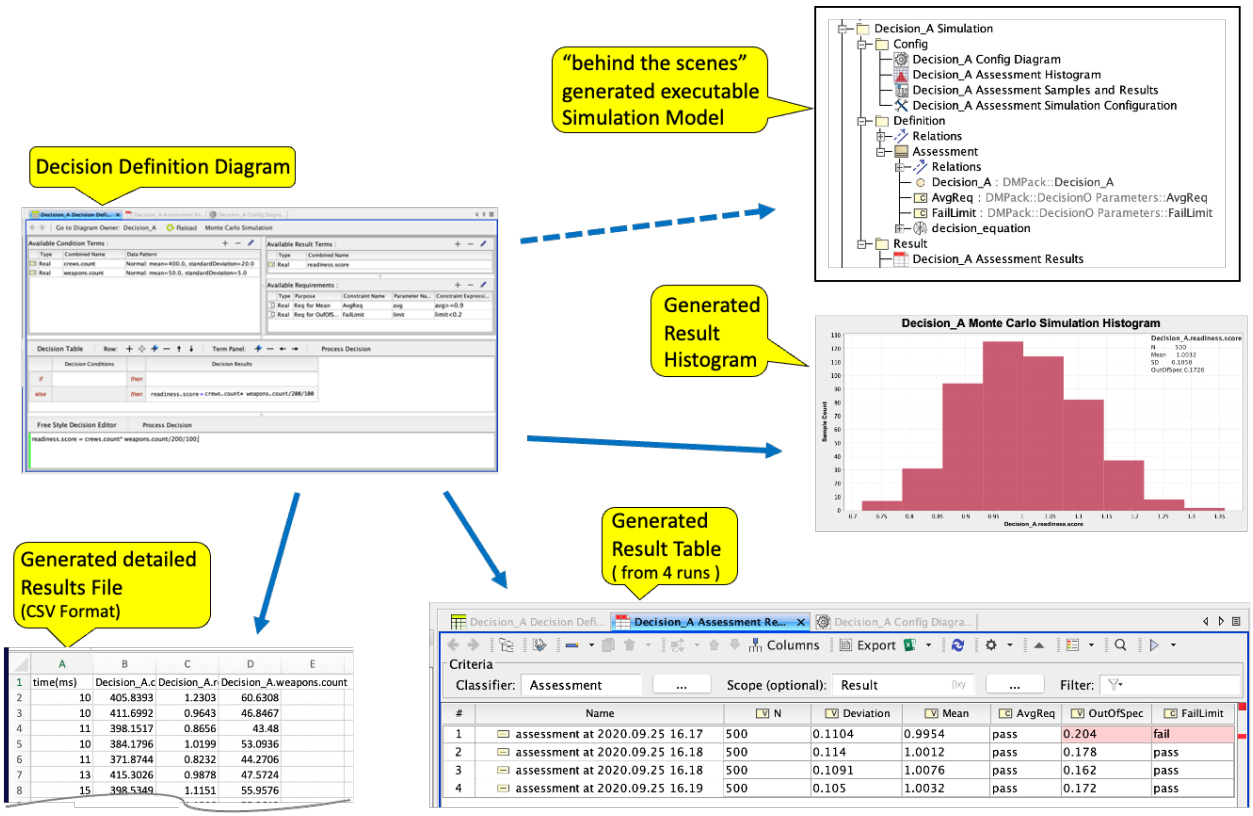

Figure 7 - The Monte Carlo Analysis Environment and Capability

The Decision Modeler utilizes the Monte Carlo Analysis program that is part of the SysML commercial plugin available for MagicDraw. This program is tightly associated with SysML modeling, corresponding to its original usage as a "systems engineering trade study" tool. To use it in the Decision Modeler context, the Decision Model must closely resemble SysML. This means model elements must be represented as SysML Blocks, properties and facts must be represented as SysML Value Types, etc. All these model adaptations are performed automatically as soon as the user selects the "Create/Update Analysis Configuration" option in the Monte Carlo Analysis Menu. The generated additional model elements are enclosed in a separate Package and do not alter the Decision Model in any way. The model augmentation algorithm is guided by auxiliary Stereotype elements in the Decision Model, the additional model elements are provided in a template-like form by a model library associated with the

Decision Modeler. This library (and the library containing the decision expression functions too) are installed during the Decision Modeler plugin installation.

The companion report "Project Technical Report" (CDRL A010) describes the Monte Carlo Analysis process in detail. The SysML origin of the Monte Carlo Analysis program imposes some constraints, the most important is that the input and result facts must be of the SysML type `Real`. Until SysML v1.7, SysML had its own primitive types, which are technically equivalent with the primitive types of UML, or even Java types, but have their own distinct type identifier. The created auxiliary model creates SysML-typed equivalents of input and output facts, and takes care of the binding between them and their original elements in the Decision Model. The same is true for the input fact distribution characteristics and the result fact constraints.

The Monte Carlo Analysis runs on the execution engine internal to MagicDraw. This engine was not originally designed for such data- and algorithmic-intensive applications. The execution times for a Monte Carlo Analysis of a Decision element are rather long. We measured execution times around 108 seconds for a run with 500 samples for an expression with two input standard-distributed facts. The execution time is linear to the sample count, we measured 1041 seconds for a run with 5000 samples. Examining the results showed that there is no significant improvement of results beyond 600-700 samples, with results from 500 samples already being very close to this threshold. Therefore, the default and recommendation are, to use a sample count of 500.

4.4 Model Annotation Capabilities

The Decision Modeler provides two separate model annotation capabilities. One, the Application-Driven Annotation, is static and completely under control of the user or of an external application. It is in particular independent of, and in no way influenced by, any model execution. The user has the ability to define seven distinct annotations with their markup color and text. These definitions are stored as MagicDraw Project Properties and are therefore global for the whole project. The visibility and visibility style for these annotations is controlled Project-wide and stored as a Project Property too. Every Decision Model element, except the connector elements and comments, may be annotated with one annotation. The ID of this annotation is stored in the model element as tag value. All these definitions and annotations become persistent after the Project is saved.

The second form of annotation, Execution State Annotation, is volatile, and only available during execution of the Decision Model. The only user control is to turn on, turn off and clear the annotation. The creation of the annotation is completely under control of the execution engine of the Cameo Simulation Toolkit. While the Execution State Annotation of the Decision Requirements Diagram follows the symbolic and color scheme of the native Cameo Simulation Toolkit annotations (as for Activity and State diagrams), all annotation graphics for the Decision Requirements Diagram are generated via Java programs, which are triggered by intercepting the annotation triggers from the Cameo Simulation Toolkit execution engine. These graphical annotations exist only during the existence of the Decision Requirements Diagram at or after the execution. They are never stored in any form.

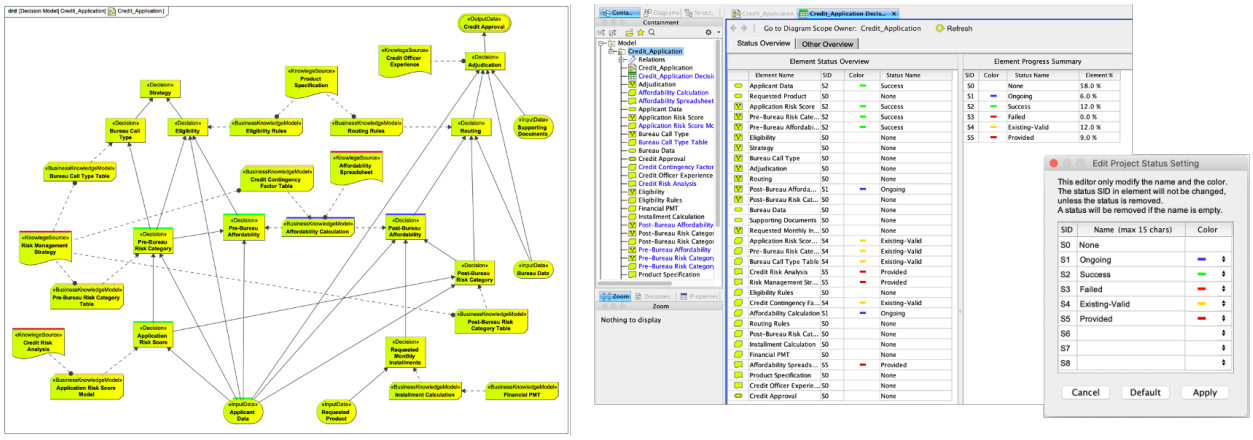

Figure 8 - Application-based Annotation Feature

4.5 Limitations and Complications

A Decision Model represents a graph, where the nodes are `Decision` and `InputData` elements, and the edges are `InformationRequirement` connectors. Since it is impossible to impose a strict order in which these model elements are created, the modeling process represents a proverbial "chicken-and-egg" situation: UML requires the types at the ends of a connector to be defined before it can be created, while knowledge about these types only exists after the decision expressions in the connected `Decision` elements are created. To allow creation of the Decision Model graph, the connectors are created with an initially empty container type `InformationItem`. But then one of these situations will arise:

- All created `Decision` elements and `InformationRequirement` connectors are still empty at creation time (meaning no facts or expressions are yet defined). This will allow unrestricted creation of a skeleton Decision Model graph, but some connectors may become invalid later when the fact types on the connector ends become incompatible with no available transformation, based on the then defined decision expressions within the connected `Decision` elements.
- All `Decision` elements are fully defined, including fact types and decision expressions, before they become connected to form the Decision Model graph. In this case, created `InformationRequirement` connectors become immediately invalid if the connected facts are type-incompatible and no transformation is possible or available.
- Likewise, in an already valid Decision Model, `InformationRequirement` connectors may become invalid if changes to the decision expression in any of the connected `Decision` elements makes the connected facts type-incompatible, and no transformation exists.

The equivalent situation exists also between `Decision` and `BusinessKnowledgeModel` elements, or between chains of `BusinessKnowledgeModel` elements.

Building a Decision Model graph from Decision elements and `InformationRequirement` connectors creates a skeleton model, in which the type(s) of the items (facts) flowing along

the `InformationRequirement` connectors are initially unknown. When the Decision Model is finally fully defined, meaning all connectors are in place, and all input facts, result facts and decision expressions are defined, then the fact types presented on both ends of an `InformationRequirement` connector may no longer agree. This requires the insertion of transformations into the item flow along the `InformationRequirement` connector, if the types are transformable. Otherwise, the modeler must be informed and asked to resolve the problem.

The Decision Model Profile associates the connectors with UML InformationFlow elements, which in turn use a thin anonymous container element called `InformationItem. This` container allows a common set of actions to perform the type transformations, or triggers the incompatibility message to the user. The real complication is to implement this in the MagicDraw diagram handling. There is no official way to do this, no documentation, and questions to No Magic were not answered. We resorted to an extensive and deep level reverse engineering of commercial plugins for MagicDraw, and finally found a single clue. However, this came late in the project when remaining funds were very limited, therefor this mechanism is only partially implemented.

The integration with the Monte Carlo Analysis package provided in the commercial MagicDraw SysML plugin, the programmatic interaction with the Alf compiler, and utilizing the JavaScript interpreter and execution engine embedded in MagicDraw were also areas without any support, besides intensive reverse engineering. Fortunately, the solutions were more obvious and therefore found earlier in the project timeline.

Two more fundamental limitations of UML are the lack of time as a first-class modeling element, and the static single-classification scheme throughout the language. The inability of UML to model things in *time and space* prevents a correct modeling of snapshot-based evaluations. There is no way to model the temporal progression from one snapshot at time *t* to the succession snapshot at time *t+1*. However, this ability is necessary to model the dataflow evaluation of the Decision Model graph along a progression of time. This behavior can only be approximated by cyclic re-evaluation of the whole Decision Model with a low-enough frequency to let everything in the model settle at the end of each cycle. The big difference of this cyclic evaluation compared to a temporal evaluation is that every cycle in the cyclic evaluation is totally isolated, while the progressions in the temporal evaluation are coordinated through a temporal context.

The consistent static single-classification scheme of UML makes handling of the item flows between Decision elements harder than it needs to be. Ironically, in the set of Object Actions, UML provides a `ReclassifyObjectAction` which supports dynamic classification changes and multiple classification of objects. While the MOF Support for Semantic Structures (SMOF) OMG specification picked this idea up, UML itself has no way to use this capability within its language.

4.6 Co-Simulation Capabilities

Executing a Decision Model on digital computer system actually means evaluating a snapshot of the Decision Model graph, with all the `InputData` values frozen for the moment of the snapshot. The only environment capable of a true continuous evaluation would be an

analog computer, but those are no longer in use. But in reality, a recurring evaluation with a sufficient short cycle time is as good as an analog evaluation, with the benefit of better value precision provided by the digital representation. In co-simulations performed across multiple participating systems, cyclic evaluations are the norm, and cycle times are typically longer than they would be in an isolated Decision Model graph evaluation.

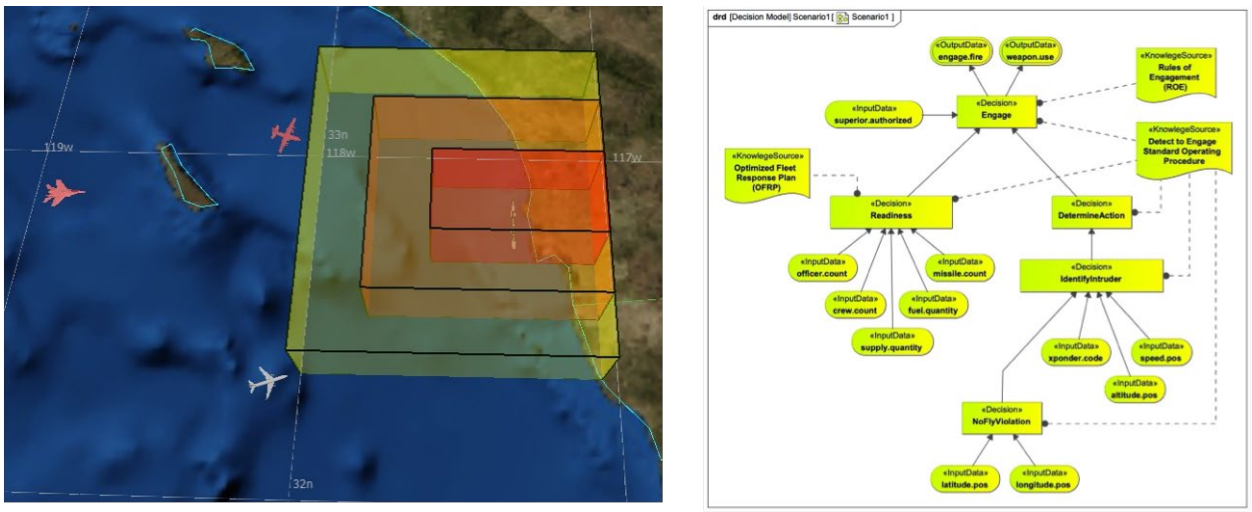

Figure 9 - The No-Fly Zone Co-Simulation Demonstration Scenario

We have demonstrated a co-simulation between the Decision Modeler executing inside the MagicDraw / Cameo Simulation Toolkit environment and AFSIM. The scenario has multiple aircraft approaching a fictive No-Fly Zone, and the Decision Model determining the course of actions based on the facts periodically delivered by AFSIM. The cycle time in this experiment was 1 second. The facts delivered by AFSIM were aircraft position, speed and altitude, and the transponder code emitted by the aircraft. The No-Fly Zone had three concentric areas. Depending on the reported position outside or inside these sub-areas, the speed, altitude and transponder code, the Decision Model produced one of three results: (1) ignore the aircraft, (2) track the aircraft, and (3) attack the aircraft. Also, the decision of weapon type for the attack was determined based on altitude and speed. See the Project Technical Report (CDRL A010) and a narrated video of the simulation for details about the No-Fly Zone Scenario.

The design goal for the No-Fly Zone Scenario was to create a research demonstrator to prove the power of Decision Modeling in a detect-to-engage mission context. While we collaborated with Navy Subject Matter Experts to ensure that the scenario was reasonably realistic, we restricted the information and methods used to be strictly unclassified. The scenario does not, and was not intended to, represent actual US Navy tactics, techniques, and procedures, it demonstrated, however, a successful integration of mission engineering with decision modeling.

Performing a co-simulation requires a real-time[9] capable communication between the participants of the co-simulation. We designed and implemented the SimCom Protocol for that purpose, a lightweight RPC-like[10] application-level communication protocol. For simplicity, we used a TCP[11] socket as the underlying network protocol. For details of SimCom, please refer to the Project Technical Report (CDRL A010).

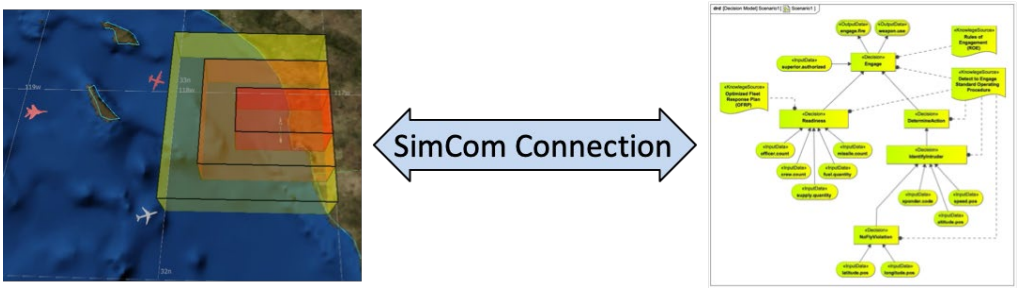

Figure 10 - Co-Simulation SimCom Communication

Our partner for the co-simulation experiment was the San Diego Office of the MITRE corporation. MITRE provided AFSIM and the simulated flight of the three aircraft. We provided the implementation of the Decision Modeler, the Decision Model, and the SimCom implementation for AFSIM: a library to be loaded into AFSIM, and for MagicDraw: a dedicated SimCom plugin. The internal communication between the Decision Modeler plugin and the SimCom plugin uses a socket-based memory channel.

The limited model execution resources provided by MagicDraw in combination with the Cameo Simulation Toolkit do not support full-scale Decision Model execution within MagicDraw (it is a *modeling* tool after all). However, the execution extension capabilities provided by SimCom, and the tight integration of SimCom with the Decision Modeler itself, allows the transparent insertion of (external) execution agents into the Decision Model execution scheme. We used this ability in the No-Fly Zone demonstration to link the individual decision steps performed in the Decision Model during model execution into a coherent execution flow. In order not to over-complicate the demonstration scenario, we "buried" these agents into the SimCom interface library we developed for AFSIM and provided to MITRE. These embedded agents (specified using the initial, UML-based, version of the Agent and Event Metamodel (AgEnt)) handled not only the Decision-to-Decision linkage in the Decision Model, but drove also the real-time log of events visible in the lower left corner of the screen during scenario execution. (See also the video recording of one full No-Fly Zone Scenario execution).

[9] "real-time" not necessarily means fast, but performing with strict timing and time synchronization.
[10] RPC : Remote Procedure Call.
[11] TCP : Transmission Control Protocol, part of the TCP-IP Networking Stack.

5 STANDARDIZATION ACTIVITIES

All research, development and implementation work during the FAST project was to some extent influenced by international standardization. For Decision Modeling, we were partially at the receiving end, taking input from the OMG Decision Model and Notation (DMN) specification, and from academic work around decision table technology world-wide.

But at the same time, we were actively involved in five standardization activities at the OMG, in some of them even as the task leader:

MOF to RDF Transformation - This task had started long before FAST and was completed in the early months of the FAST project. It defines a fully automatable transformation process from MOF (and UML) models to OWL ontologies. We were leading the standardization process and used this technology to create a FAST-compliant version of the W3C Sensor Ontology.

Metamodel Extension - This small task applies and extends the techniques of the "Semantic MOF" (SMOF) to UML Profiles, to make these profiles more capable and easier to implement. Since we had been task leader on SMOF, we were leading this task too.

Systems Engineering Modeling Language version 2 - We participated in this huge effort within the subtask for core language and metamodel development. The goals and achievements of this standardization effort were largely overlapping with the original plans for FAST, allowing us to achieve at least some of our intended work after being redirected to decision modeling. For the future of decision modeling, we discovered from our involvement in the SysML v2 development a way to implement a more capable and powerful decision modeling technique as extension to SysML v2.

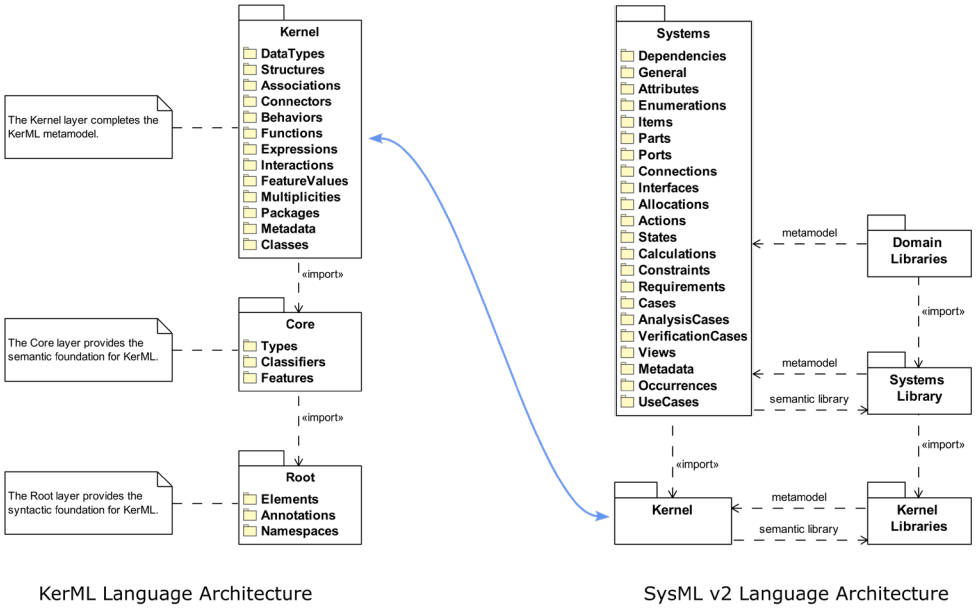

Figure 11 - SysML v2 Architecture

The SysML v2 architecture differs significantly from the UML architecture. On a macroscopic level, it consists of the Kernel Modeling Language (KerML) layer, and the SysML layer, which specializes KerML for Systems Engineering modeling. This is shown in Figure 11.

KerML is a neutral, general-purpose modeling language, similar to UML, but with significant differences:

- KerML implements 4D Modeling, all modeled elements are occurrences in time and space, all behaviors and interactions are strictly temporal coordinated
- The KerML Semantics are mathematically and ontologically grounded (see also the annotations in Figure 12).
- KerML provides a rich and extensible Expression Sub-language, expressions participants are not limited to mathematical elements, but can include any model element defined in KerML.
- KerML is unlimitedly extensible through model annotations and model libraries without adversely affecting tool implementations

The annotations in Figure 12 provide additional details.

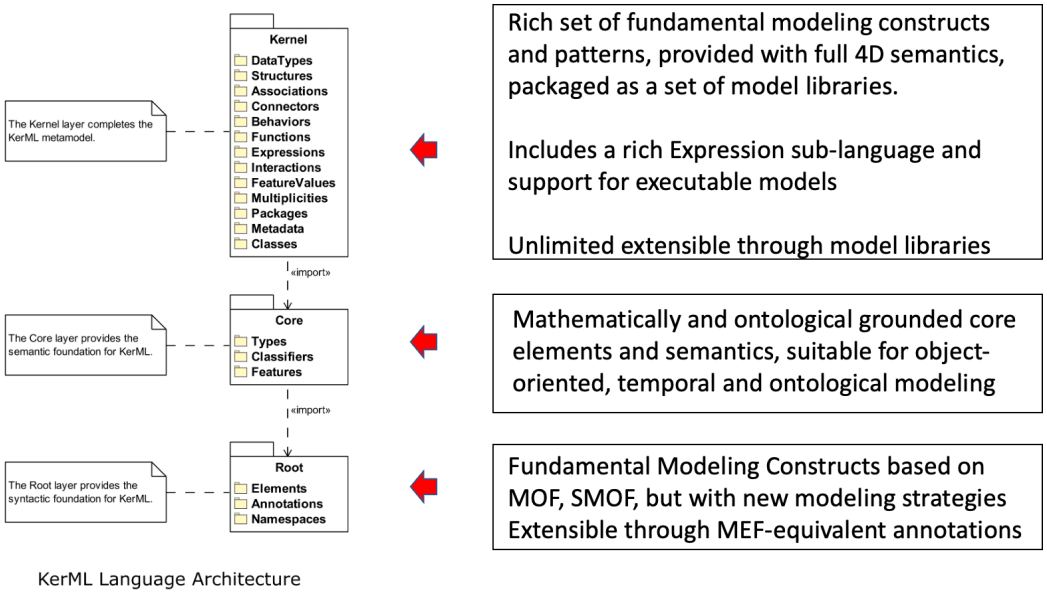

Figure 12 - KerML Architecture Details

Agent and Event Metamodel - This task has long lingered due to the inability of UML to precisely express temporal models, a topic that is core to the definition of autonomous and reactive systems like agents. We created a UML model in the attempt to model an existing agent core prototype built by 88solutions long before the FAST project, but the model could not express the most important behavioral features of that prototype. We restarted this work using SysML v2 technology (mostly KerML), which provides the temporal and functional modeling foundation required for the specification of reactive and possibly autonomous agents. This work is very promising, but fell time-wise in to the period of FAST with very limited funding availability.

Precise Semantics for Uncertainty Modeling - We participated in this task led by the Simula Research Institute in Norway. Uncertainty is a key factor in decision making, and therefore an important factor for future Decision Modeling. Our Decision Model (and DMN too) is not (yet) considering uncertainty. The results from this research and standardization work became only available after FAST was out of funding, but uncertainty should be a strong factor in future decision modeling work.

6 DISSEMINATION ACTIVITIES

Throughout the three years performance of the FAST project, we performed numerous presentations about our work, and demonstrations of our tool developments. These activities fall into four categories:

6.1 FAST Technology Presentations and Tutorials

At the kick-off meeting, and during the first nine months of the project, we made several presentations about our proposed approach to improve the efficiency of software and systems development using an advanced, pattern-based, modeling approach. This included also presentations and demonstrations of our distributed code analysis technology, which we had developed before FAST, but slightly refined in the first months of the project.

The most notable presentation venues were the Naval Information Warfare Center – Pacific in San Diego, California, the Naval Information Warfare Center – Atlantic in Charleston, South Carolina, the United States Army Special Operations Command at Fort Bragg, and the Army Research Lab in Durham, North Carolina.

We performed also tutorials about Systems Engineering using SysML for the Naval Information Warfare Center – Atlantic in Charleston, SC. A tutorial for Naval Information Warfare Center – Pacific in San Diego was unfortunately cancelled after one day due to the emerging COVID-19 lockdown.

6.2 Decision Modeling Presentations and Demonstrations

We performed a large number of presentations and demonstrations to interested audiences from the US Navy and US Air Force. Due to COVID-19 restrictions, all these presentations and demonstrations were performed using Internet collaboration techniques.

6.3 Co-Simulation Demonstrations

In collaboration with the San Diego Office of the MITRE Corporation, we developed a co-simulation demonstration between our Decision Modeler and AFSIM, using a fictive No-Fly Zone scenario. This was the first in-person collaboration after the COVID-19 lockdown was partially relaxed. It resulted in an in-person demonstration in July 2021 at the MITRE office to our Sponsor and other Navy participants. We created also an eight-minutes video covering the demonstration, which was then reused in an Internet-based remote demonstration of the co-simulation experiment to a Navy sponsor in the Pentagon.

6.4 Other Technology Presentations and Demonstrations

We have been, and are, regular participants at the “Air Force Futures” group, recently renamed to “Digital Engineering Forum”, a recurring virtual meeting organized by the US Air Force. During the last two years, we presented and demonstrated our Decision Modeling technology twice and made two presentations about our work on SysML v2.

At the recurring Quarterly OMG Technical Meetings, we presented and demonstrated our Decision Modeler, and provided updates on the Agent and Event Metamodel work.

7 CONCLUSIONS

As requested, we applied throughout most of the project performance period a focus on decision modeling, as applicable to mission engineering. However, the applicability of decision modeling is much more widespread. It can (and should) play a dominant role in the early conceptual modeling phase during systems and software development, when the consequences of requirements and their possible ways of satisfaction are evaluated. The availability of a dynamically executable modeling capability would be extremely helpful to support this difficult task. Such a capability could be combined with knowledge and artificial intelligence technology to perform an exhaustive solution search under user guidance.

In our development of the Decision Modeling technology and the Decision Modeler tool, we made significant progress on achieving and setting the foundations for reaching the above stated goal. However, UML, and its modeling tool implementations, showed their age and are not able to fully satisfy the requirements of such an advanced, decision-based, modeling environment. Despite these obstacles, we succeeded to design and implement a flexible decision table specification system and created the execution environment to evaluate these decision tables in real-time. The resources of the underlying Cameo Simulation Toolkit restrict parallel execution, which in turn restricts the concurrent execution of decision models as a whole. Even if these restrictions could be lifted, the lack of temporal coordinated behavior and limited event capabilities of UML would make dynamic decision model execution very difficult to realize. We have, however, implemented and demonstrated the ability to perform decision-making collaborations with external tools, connected through our SimCom capability. In the right setup of such a collaboration, a dynamic decision model becomes possible, as we demonstrated with the No-Fly Zone co-simulation with AFSIM.

SysML v2, and in particular its underlying application-neutral core modeling system KerML[12] provide now all the necessary methodologies and modeling capabilities to specify such an advanced decision modeling capability. KerML considers all modeled elements to exist in time and space (also called 4D modeling), and provides an extensive set of temporal modeling capabilities to define model behavior and behavior execution. In addition, the KerML (and SysML v2) architecture is based extensively on Model Libraries, which eases language extensions.

Decision Tables, and to some extent Decision Modeling, have been the subject of academic research for many years now. Despite this, there are surprisingly few implementations of this technology on the market. The FAST project provided us with the opportunity to research the integration of decision technology into main-stream modeling. We accumulated valuable experience, but finally reached a point where the quite dated UML technology became an obstacle for further progress. But the FAST project was blessed with the opportunity to engage deeply into a bottom-up, brand-new modeling system: SysML v2. It is not just a new version of SysML, or even a new language, it is a new modeling methodology. With this new technology, and our experience from being part of the core developer team, we see a way forward to introduce decision technology into a wide range of modeling tasks, not only mission engineering.

[12] KerML : Kernel Modeling System

The first step will be the completion of the second-generation Agent and Event Metamodel (AgEnt) using KerML technology, resulting in a loadable KerML Model Library, extending the KerML modeling system. This is a "Past-FAST" activity already going on.

With the AgEnt extenisons to KerML in place, a new Decision Modeling framework can be built, also as a KerML Model Library. The whole Decision Graph Modeling environment can be brought forward and transformed into the structures required by KerML. This should be a straight forward task as soon as commercial KerML and/or SysML v2 implementations become available. We recommend re-writing the Decision Table and expression execution part from scratch, carrying past experience forward, but utilize the very powerful KerML Expressions Sub-language. Active decision elements, like `Decision` and `BusinessKnowledgeModel` shall be implemented as (autonomous) agents, using the upcoming AgEnt KerML Model Library (see above). Decision Model execution and simulation could then be performed outside the modeling tool in an agent framework. The ubiquitous temporal coordination provided by KerML makes dynamic whole-model execution feasible.

REFERENCES

[A010] Manfred Koethe: *FAST Project Technical Report,* FAST project CDRL A010

[AgEnt] Manfred Koethe: *Agent and Event Metamodel*, an ongoing technology submission to the Object Management Group,
(not yet published)

[BPMN] *Business Process Model and Notation*, version 2.0.2, an OMG specification
https://www.omg.org/spec/BPMN/2.0.2/

[DMN] *Decision Model and Notation*, version 1.4, an OMG specification
https://www.omg.org/spec/DMN/1.4/

[MOF2RDF] MOF to RDF Transformation, version 1.0, an OMG specification
https://www.omg.org/spec/MOF2RDF/1.0/

[MEF] *Metamodel Extension Facility*, version 1.0, an OMG specification
https://www.omg.org/spec/MEF/1.0/

[PSUM] *Precise Semantics for Uncertainty Modeling*, version 1.0, an ongoing technology submission to the Object Management Group, adoption process initiated, expected URL will be
https://www.omg.org/spec/PSUM/1.0beta/

[SysML] *OMG Systems Engineering Modeling Language*, version 1.6, an OMG specification
https://www.omg.org/spec/SysML/1.6/

[SysMLv2] *OMG Systems Engineering Modeling Language*, version 2.0, an ongoing technology submission to the Object Management Group, current release 2022-10,
https://github.com/Systems-Modeling/SysML-v2-Release

[UAF] *Unified Architecture Framework*, version 1.2, an OMG specification
https://www.omg.org/spec/UAF/1.2/

[UML] *Unified Modeling Language*, version 2.5.2, an OMG specification
https://www.omg.org/spec/UML/2.5.2/