FUSED Implementation Summary

Submitted by: Adventium Enterprises

Contact Information: Mark Boddy (mark.boddy@adventiumenterprises.com)

FUSED is a language used to specify compositions of design engineering models written in a variety of other modeling languages. The implemented and delivered FUSED toolset is a tool integration framework that supports multiple engineers collaborating in the development of a diverse set of engineering models used for multiple purposes in multiple phases of design. The implementation consists of a set of translation and process management tools that manages the semantically-correct transfer of information among a set of models. All of the implemented software is open source, though the demonstration system does employ some COTS and freeware tools. Where it is permissible, we have included installers for these tools. Where that is not allowed, we provide pointers to the appropriate web-sites for tool purchase or authorization, as required.

FUSED is extensible to support a chosen set of modeling environments. Implemented examples in our software delivery include requirements (in SysML), solid geometry (in Creo), computational fluid dynamics (in AVL), dynamical systems (in OpenModelica), and vetronics/avionics (in AADL). An extensible language approach is used, so that many FUSED capabilities are presented to domain experts as minor additions to familiar languages and tools. We also provide a FUSED language to specify compositions of models. Compositions may be used for multiple purposes, for example to specify multiple views of a component, verify inter-model consistency, specify part/whole assemblies, or apply design operations to models. The delivered system includes support for nine domain-specific modeling environments, used to model various aspects of a small UAV design. A set of demonstrations are presented, both as a video and in document form, using this instantion of a FUSED integration.

FUSED uses the Silver higher-order attribute grammar language and tools to specify and implement tools that understand and extend the various modeling languages of interest. Developed as part of the *Minnesota Extensible Language Tools* research project at the University of Minnesota (http://melt.cs.umn.edu/), Silver is open-source software, available from the University. Silver has special support for defining and implementing sets of extensions to existing languages, not all of which will necessarily have formal grammars or semantics. A modeling toolset uses many more file formats than just the language written by users; there are typically a variety of intermediate and analysis result file formats. The delivered implementation includes tools that can parse Creo/ProE mass properties analysis files and AVL aerodynamics stability derivatives files.

In the delivered implementation, Silver is used to generate tools that can extract model elements from files written in supported modeling languages, convert them to a canonical internal representation (which inside the tool takes the form of a higher-order attributed abstract parse tree), and convert from a canonical representation to any other language representation in which elements of that type make any sense. One concrete representation that we have defined for every ontology type is an XML representation. Any collection of model elements in any hierarchical namespace structure can be written to a file in this format. The implementation also includes a set of basic operations on elements of the internal ontology. Examples of these operations include extracting a subset of elements from a collection, composing elements to form a new collection, adding type qualifiers, or checking simple properties for an element.

FUSED includes a graphical composition language, implemented via an editor and compiler using Eclipse GEMS. The compiler generates ant build scripts. Executing a composition means executing a target in one of these ant build scripts. The overall execution is actually a hierarchy of build scripts that call each other. These scripts can be divided into two kinds, those that are generated entirely by the compiler for a specified composition, and builds for a particular model developed in a particular modeling environments (which are the leaf builds in a tree of builds invoked for a particular purpose by the system engineer). The FUSED compiler proper generates build scripts , and passes paths to collections of FUSED elements in canonical representation between model builds, performing FUSED operations on these collections as specified to get the various models to talk to each other.