The Role of Modelica in a Robust Engineering Process

Peter Harman
deltatheta uk limited
The Technocentre, Puma Way, Coventry, CV1 2TT, UK
peter.harman@deltatheta.com

Abstract

An engineering process comprises stages of design, analysis and optimization, often with specialist tools and engineers in each of these areas. We look at how the interaction and communication between these stages can be tailored to maximize the robustness of the overall process, and what the role of Modelica is in achieving this. This is described for industrial examples. We then discuss the additional requirements of Modelica tools and libraries for such a process.

Keywords: Engineering process; simulation strategy; parameter management; Product Lifecycle Management (PLM)

1 Introduction

The requirements of physical modeling tools in terms of their overall structure and capabilities [1] and the history of their development [2] have been discussed with the common conclusion that a correctly developed tool can greatly simplify the job of the user by the use of features such as symbolic manipulation, drag-and-drop modeling and version control. These however prompt the further question of what can additionally be done for a physical modeling tool to have a greater influence on the engineering process as a whole. This paper attempts to study this question with industrial examples and the tool requirements these create.

2 Robust Engineering Processes

2.1 A Definition

Rather than robustness of the developed product we define a robust engineering process as one where:

- Errors due to different engineers using different sets of data are not made
- Engineers do not spend time performing calculations just to move a design from one toolset to another, such as CAD to mathematical model or vice versa
- The true optimized design is found and the fact that assumptions and objectives change during the process is both acknowledged and handled

Although this definition sounds like purely policy changes, it is really a matter of communication and data management. Figure 1 shows a comparison between two engineering processes. A serial process where stages of design and analysis follow each other, with data translations between the two, and a concurrent process where design and analysis engineers draw data from the same source, which is updated with any changes made by each, and where translation is automated. This concurrent process is the robust process we aim to achieve, and although in this case it may be a small improvement, considering a case where more areas, such as control systems development or test engineering, are involved in the project, the efficiency is greatly improved.

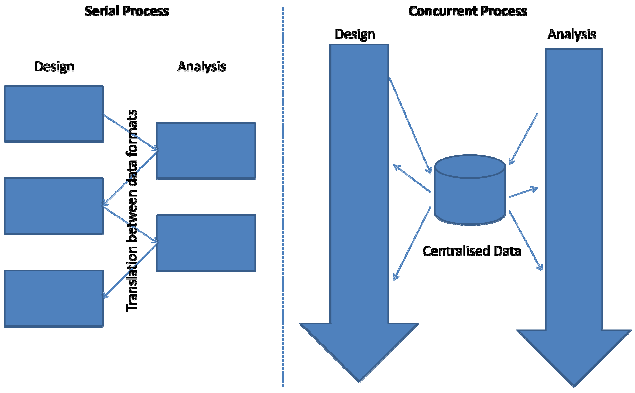

Figure 1: Serial and Concurrent Processes

It is important to stress that this process does not involve engineers blindly using whatever data is currently available, management of versions is critical and as with CAD data or software version control there must be a process of publishing / releasing fi-

nished work and monitoring dependencies between versions.

2.2 Modelica and PLM

An analogy can be drawn between successful introduction of Modelica into an engineering process and the introduction of Product Lifecycle Management [3] (PLM). Modelica aims to maximize model reuse and therefore Modelica users within a company will be working from the same set of libraries. In the same PLM allows all engineers in the company to work from the same data, whether in design, manufacturing or purchasing.

3 Industrial Examples

We will look at two industrial examples, one where speed of development is critical, and one where development is shared between different organizations.

3.1 Formula 1

In Formula 1 timescales are compressed and commercial success can only be gained from the most optimized design. While a small industry, it is a clear example of the benefits of efficient and robust design processes and has been an example often used in the introduction of PLM [4].

In this environment simulation is critical and is typically applied in different areas across the engineering departments within a team. This may include CFD of the aerodynamics, multibody modeling of the vehicle dynamics, lap simulation, powertrain simulation, hydraulic system simulation, control system development and driver training simulation.

Each of these areas might use different simulation tools and hence different models. There is a cost motivation to share models between departments in order to save work, and a technical motivation in order to ensure all departments are working from the same assumptions. The reason for the different tools is that the requirements of the models in these simulations are very different, and even where the domain of the model is related, such as between a vehicle dynamics simulation and a lap simulation, the detail level varies widely.

Modelica can be used to unify the modeling and simulation across the organization as long as:

1. The Modelica environment used can generate simulation code for all the required platforms.
2. The models make use of replaceable components to achieve different detail levels with the same parameter data.
3. The models can be parameterized directly from design data and car setup data.

Achieving this reduces the time from a design change or car setup change to having a running simulation, and hence the time for simulation engineers to feedback recommendations based on results. Since the start of 2009 track testing during the season has been banned making simulation more critical, and it is desirable that all design changes are successfully analyzed before racing so such a reduction in the required time to do this is important.

3.2 Automotive Tier 1

Tier 1 suppliers in the automotive industry have ever growing responsibility for the delivery of complete systems for the vehicle. Whilst this is fully within the engineering capabilities of the suppliers, defining the responsibilities for system integration often causes issues. This becomes especially important when a system integration issue arises during production creating a situation of assigning blame and responsibility.

By sharing systems engineering data, the Tier 1 and OEM can speed up system integration during development, and speed up location of faults, and hence responsibility, with failures during production. Virtual testing of the full vehicle can occur early in development using models developed by those responsible for the component development.

As an open-standard, Modelica is ideal for such sharing of models. This is achievable as long as:

1. Design versions are managed and a model can be selected that corresponds to a particular design of the system.
2. Models can easily be communicated, either by packaging models and resources in a single file, or by using standardized protocols e.g. a PLM system or a Subversion version control system.
3. Supplier and OEM intellectual property can be protected without hiding the intention of and operation of models.

Achieving this reduces duplication of modeling work in both parties, ensures models are developed by those closest to the design, and creates a collaborative development.

4 Requirements for Tools and Libraries

The concepts discussed here are entirely feasible within the existing Modelica tools available. However in order to achieve the levels of efficiency, data management and integration required to achieve the processes mentioned there are a number of recommendations for both Modelica tools and libraries.

4.1 Integration of tools

In the F1 example above one requirement was to parameterize models from design data and car setup data. This is a wider issue where many users require parameters sourced from spreadsheets, CAD software and company specific software.

Much data is stored in spreadsheets as a convenient form for viewing and performing calculations. Rather than copying data from the spreadsheet a tool should allow model parameters to be linked to spreadsheet cells, and updated as the spreadsheet file is updated.

Translators from CAD to Modelica multibody models have been written [5] however to successfully connect the two the Modelica tool would be able to monitor the CAD data for changes and the model would be continually updated accordingly. The CAD data contains more than just the structure of the assembly and mass and inertia data. The model should be able to request other parameters from CAD features e.g. the area of a piston for a hydraulic actuator.

Most engineering companies will have data stored in a company specific manner. Following the examples above, an F1 team will have data for setup of the car by race engineers which will ideally be possible to load into a simulation, an automotive tier 1 might supply multiple customers and have data sets for each customer which they want to keep separately to the model itself. Tools should have an API to allow parameterization from any data source.

4.2 Protection of models

In the Automotive Tier 1 example discussed above one issue was protection of IPR. There are three main solutions in this area, each with advantages, which one is preferable will always be an area of debate.

1. Encryption: The Modelica text is encrypted and the tool does not allow access to it. This has the advantage that the model structure is preserved and optimizations can be performed when it is simulated, however tracing bugs when the tool will not allow access to equations or variable names can cause problems, and the encryption is tool-specific and not part of the Modelica specification.
2. Compilation: The model is compiled to a dynamic linked library (dll) that can be used at simulation time. The MODELISAR [6] project will define a standard for this, so the issue of compatibility is reduced, however compiled code is platform specific, and no further optimizations or analysis of the model structure can be performed.
3. Obfuscation: This is a means of hiding the details of interpreted languages, most widely used examples being Javascript code in web pages, by changing the names of variables and methods and using complex syntax to make the code unreadable but still functional. This has not been widely explored with Modelica but may provide a tool-neutral alternative to encryption.

4.3 Communication of models

The Modelica specification defines a filesystem structure for storage of Modelica libraries and a mapping between files and classes. In addition to the Modelica code a library often requires resources such as image files and HTML documentation. There is a proposal for Modelica 3.2 to allow an entire library and associated resources to be stored within an archive file similar to .zip or .jar files. This would ease communication of models as only 1 file is required for an entire library.

An alternative approach for ensuring libraries can be shared and transmitted would be for Modelica libraries to be loaded directly from a central source e.g. a version control or PLM system.

4.4 Modelica compliance

Release of new features in the Modelica specification not surprisingly precedes the implementation of them in tools. Also some features may not be relevant to the focus of a particular tool and are not implemented. Communication of models between different companies who might use different software could be complicated by whether one Modelica tool supports the same Modelica features as another. At

the moment there is no standard description of compliance to the language specification. It would be very useful to companies using or considering using Modelica to be able to compare compliance to the standard, this could be illustrated with a table of features against whether the tool in question supports them.

4.5 Discretization of models

Parameterization of a model from CAD and other design data is made easier if the discretisation of the model, that is the division of the model into individual components, matches the physical system e.g. one physical part, and hence one CAD part, corresponds to one Modelica component.

5 Conclusions

Successful adoption of Modelica can influence more than just model development and generate improvements in the engineering process as a whole. There are actions required by engineering companies adopting Modelica in order to achieve this, but also actions on Modelica tool and library suppliers to aid this. Tool integration and communication of models are important features which must be considered. To ease compatibility of libraries to Modelica tools, a matrix of tool compliance to individual features of the Modelica specification should be published.

References

[1] Tiller, M. *"An Analysis of Physical Modeling Requirements and Techniques"*, SAE 2009

[2] Breitenecker, F. *"Software for Modelling and Simulation - History, Developments, Trends and Challenges"*, Simulation and Visualization 2006

[3] Stark, J. *"Product Lifecycle Management (PLM): 21st Century Paradigm for Product Realisation"*

[4] Jeffreys, M. *"The Challenge of Introducing Product Lifecycle Management in Formula 1 and lessons for other organizations and sectors"*. Matthew Jeffreys Consulting Ltd, 2007.

[5] Engelson, V. *"Tools for Design, Interactive-Simulation, and Visualization of Object-Oriented Models in Scientific Computing"*, Linköpings Universitet 2000

[6] *"MODELISAR Project Profile"* 2008 [online] http://www.itea2.org/public/project_leaflets/MODELISAR_profile_oct-08.pdf