DYNAMIC FUNCTION MODELING CONCEPT AND ITS SOFTWARE REALIZATION

L.S.Chechurin, H.M.Bakker, W.W.Wits

Key words: TRIZ, Function Modeling automation, CAD, Modelica, SolidWorks

Introduction

Engineers require various types of modeling when they try to improve a product or a process in a systematic manner. First, they express the situation in a certain language. This can be a drawing or a technical description in a natural language (for example a patent, where the engineering system is described by a set of drawings and a text). This type of modeling is relatively easy, but unfortunately, too "soft" to be a substitute of the reality. In other words, there is a big amount of "common sense" and/or uncertainties in these models.

The extreme alternative is the mathematical model (based on the understanding of physical processes). Mathematical model analysis can provide a very deep insight into system operation and, therefore, assist system improvement at a very non-trivial level: for either optimization of system parameters or analytical experiments on structural changes. That is why an attempt to derive a mathematical model of the situation usually takes place at the first stage of any engineering project. In other words, engineers try to describe the reality by the language of mathematics. However, this technique is difficult for most of the real cases. To choose the proper type of modeling and to work with these models analytically require a high level of professionalism and has been known rather as an art than an algorithm. Therefore, approaches of turning the art into routine (automation) are highly needed. And what we observe in modern computer-aided research is the growing amount of automation of mathematical modeling in design and analysis.

Examples of modeling automation are

1. Finite element modeling (FEM). In general, after the geometry of an object is input, the software automatically splits it into (typically millions of) elements by meshing. Then it applies the appropriate basic mathematical model of the element and assemblies the whole picture of the

system state or dynamics (meshing the time as well) in the form of a system of (millions of) elementary equations. There are no analytical benefits of this type of models because of their size, but the numerically obtained solutions of them are widely used as the system's state or dynamics simulation.
2. System kinematics can be derived from drawings if the user assigns joints, movable and fixed parts etc. Models of allowed (according the system geometry) movements can be generated in the form of explicit function or governing mathematical equation.
3. Some software can derive a system of differential equations of the dynamics of the system behavior. For example, MATLAB can derive the set of nonlinear differential equations governing the dynamics of a multi-joint manipulator given the system's geometry description. These equations might be complex but can be used for analytical manipulations (say, stability or limit cycle analysis etc).
4. Software can derive mathematical models out of a given set of data, for example system identification based on a sufficiently large amount of input/output signal records (MATLAB Identification Toolbox is an example).

Function modeling (FM) has been introduced a long time ago (see National Institute of Standards and Technology (NIST) approach [1], Function Cost Analysis, IDEF modeling etc.) as a structured standard method to describe (or to model) various types of (engineering) systems. It is more formal compared to the description by natural language and much less formal comparing to mathematical modeling. In most cases it represents a system by a graph where nodes are the system's elements and the links indicate interactions.

The contribution of TRIZ to FM (comparing to NIST, for example) is a more accurate function definition. In modern TRIZ, adopted by many TRIZ practitioners [2], a function is legitimate if (there are directly interacting material objects or fields and) it changes or maintains a parameter of the object it works on. This type of modeling the situation looks well-suited to the general idea of TRIZ since:

a) it helps to get rid of thinking inertia,

b) it provides a comfortable environment for further TRIZ instruments application like contradiction analysis, trimming, SuField analysis etc, and

c) it does not require (at least at surface level of analysis) high mathematical professionalism (compared to mathematical modeling), just a general understanding of system functioning or the physics of the interactions inside the system.

Thus we distinguish three main modeling paradigms: modeling by natural language, FM and mathematical modeling. While moving from one technique to

another the complexity of the modeling grows, but the amount of available instruments grows too. This composition immediately forces to ask several questions:

- Are there any modeling techniques of modeling "between" these paradigms, could we hybridize the modeling techniques at least? Could we smoothly increase / decrease the complexity of the chosen model? Could we develop a platform that can combine FM and mathematical modeling?
- If there are some successful attempts to automate the mathematical model design, then aren't we able to develop methods of Function model (FM) design automation?

The paper presents an approach that allows answering both questions positively.

Dynamic Function Model Concept

The practice of modeling the situation using FM reveals the following limitations and obvious places for improvement. First, a FM is known to be just a snapshot of a situation. In static system analysis it might be enough, but we have to deal with time-variant situations mostly in real world. The situation changes from one moment to another and the FM should reflect these changes: objects and functions may vary, appear and disappear. In the existing practice there are three methods to cope with this:

1. If changes are "small" then use one FM for multiple states. It means that all appearing/disappearing objects and functions will be placed into it. This can make the FM complex;
2. If changes are "significant" then use one FM for every system state. For each state the FM will reflect the situation perfectly, but there will be a need to hold a number of FMs;
3. If the changes are "too large" then use flow models, tracking the variation of an object or parameter (say, energy) during the whole process. In fact, flow models are not the models of object interaction anymore, they show the evolution of an object or parameter over time. This flow definition (introduced in Gen3ID

methodology for example) can also be generalized on space parameters as well. Then parameter variations (say, energy, stress or temperature) can be tracked from one object to another.

Some attempts to overcome the limitations of FM are interesting: A. Kashkarov [3] tried to depict various states of an element of FM explicitly, by "extended" element blocks. He also introduced quantitative analysis within FM by new definition of flows (of energy or material). Unfortunately, these attempts led to complicated FMs where the number of elements grows in accordance to the number of their states.

In this study, we try to define the FM as a *variable* graph and let the instant picture corresponds to only one moment of time. Then, varying the time from an initial moment to a final moment we can observe the whole family of FMs describing the changes of the situation in the analyzed process.

Example. Let us focus on the simplified FM of what is happening inside a boiler. A "heater" heats "water" until it boils. However, more careful analysis will reveal more details as the heating progresses. For example, by the end of heating water evaporates intensively. It means that a new element, "steam", should appear in the FM. Thus, if we analyze the process that evolves in the time interval [t_0, t_{fin}] it is suggested to define a dynamic FM, as the variable FM(t) where t belongs to [t_0, t_{fin}]. Now we have to be ready to deal with the time-variant graph, but when we design and manipulate with FM with the help of computer, it does not seem to be difficult.

From Dynamic FM to Mathematical Model

To come to these dynamic FMs, we would like to suggest the concept of FM assisted by mathematics. If there is a function between two elements (material objects or fields in being in direct interaction) of the FM, then a parameter of function recipient is maintained or changed. In some cases it is possible to apply a mathematical equation that governs the parameter (it is better to call it variable in such a case). For example, let us assume that there is the function "heat" between the "object" and "subject", and the variable "temperature" of the "subject" changes. The heating is known to be decomposed into convection, conduction and radiation. All these mechanisms can be expressed by mathematical equations or functions (more or less precise) in which the temperature of the "subject" is a

function of certain parameters. Obviously, this quantitative information about the heating could be very useful at least for the evaluation of the performance of the function. If the FM is the only model we have at a certain moment, it is necessary to perform some research in order to provide the adequate mathematical model. *But for instance in redesign where we can start from a CAD assembly, this information can be obtained in an automated manner by simulation, not analytically.* Now, FM is assisted (where possible) by mathematical modeling and provides not only quantitative information about function performance but it can even correct the FM itself.

Example. Let us again consider the boiling process in the time interval [t_0, t_{fin}]. The mathematical model of the heating process can be used to assess if the water is hot enough by the time t_{fin} to generate the required amount of steam. If yes, then the mathematical model can advise to introduce the new element "steam" into the initial FM. The mathematical simulation can also show the value of the temperature gradient in the water. Because if they are large, it is reasonable to introduce two (or more) elements named "water", say, "low temperature water" and "high temperature water". The properties (and location) of these two "waters" are so different, that it is reasonable to treat them as different elements in the FM.

Practical implementation

Husig and Kohn [10] have described an extensive bibliography about the paradigm of Computer Aided Innovations (CAI). Also, the state of the art for CAI and CAD software was discussed in [4] and [11]. This paper focuses on introducing the concept of integrating Dynamic FM and mathematical modeling. To demonstrate this concept, the software prototype from [4] was taken as a basis. This tool already provides integration of CAD software and FM. For the mathematical side, the tool was extended with support for Modelica.

Note. The development of this software is part of an ongoing research project [4, 11]. In the past, Dassault Systèmes provided a TechOptimizer add-in for CATIA v5 to define Function Models, but the functionality of that software was limited: according to its user guide, the user had to manually define the Function Model and manually link the CAD parts to the components in that Function Model. Our ongoing work provides software that enables integration, automation, user assistance and other advantages.

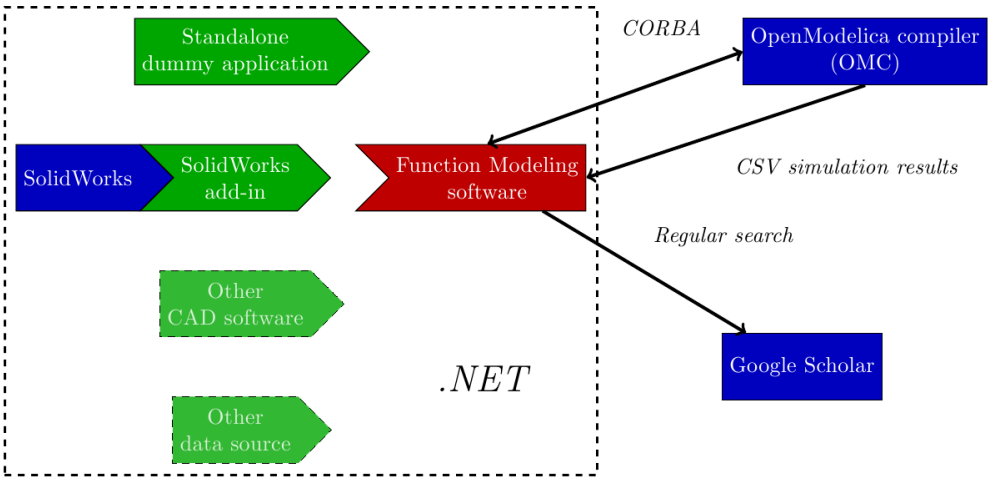


Figure 1. Software architecture

Modelica. The Modelica standard [5] describes a modeling language that allows mathematical modeling of both simple draft models and complex models with a physical context. Compared to Simulink, it provides support for non-causal equations (action-reaction) whereas Simulink only supports one-way equations (visualized by arrows in Simulink's block diagrams). It also provides support for SI-units and their derived units and it provides a library of basic components (models) for several physical and technical domains. Modelica is an object-oriented language where basic models can be instanced as objects to compose a complex system and where basic models can be extended to create models with more complex properties (inheritance [6]).

Example. An ideal DC-motor can be modeled using the models of a resistor, a coil, a rotational inertia component and a component that accounts for the EMF-effects [9]. This ideal DC-motor model can then be inherited and extended to account for heat production and friction effects to make a more realistic model.

Architecture. The FM part is implemented as a software component that can be loaded from other software. Currently there are a SolidWorks add-in and a dummy application hosting the FM component (see Fig.1). The dummy application is used to test the FM component independently from SolidWorks. Development of other host applications like add-ins for other CAD software is also possible. The FM component can open Google Scholar URLs to find literature about physical phenomena.

There are several computer programs that allow for graphical and textual editing of Modelica models and simulation of the models by using numerical integration. Out of the various implementations of Modelica, OpenModelica was chosen because of its free availability and its compiler component (OMC) that can be controlled via CORBA [7].

OpenModelica is a software collection that consists of an editor, a compiler (OMC) and an optimization program. The OMC takes a Modelica model as input and compiles it to a simulation program: an executable file which can calculate the variable values for each time step. Besides simulating it performs model checking and it can provide information about the loaded model. The communication through CORBA between the FM component and the OMC is bi-directional: commands are given to the OMC and the results are read back into the FM component.

After a model's simulation by the OMC, an CSV-file containing the results is read into a System.Data.DataTable using [8]. This DataTable is used as a cache to smoothly show the transient results while moving the time frame as will be illustrated later on.

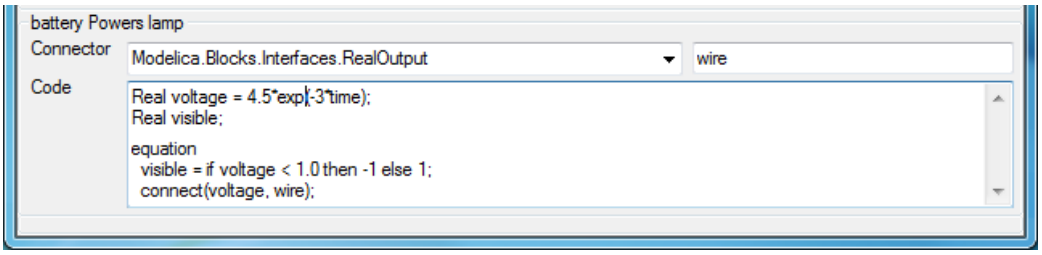
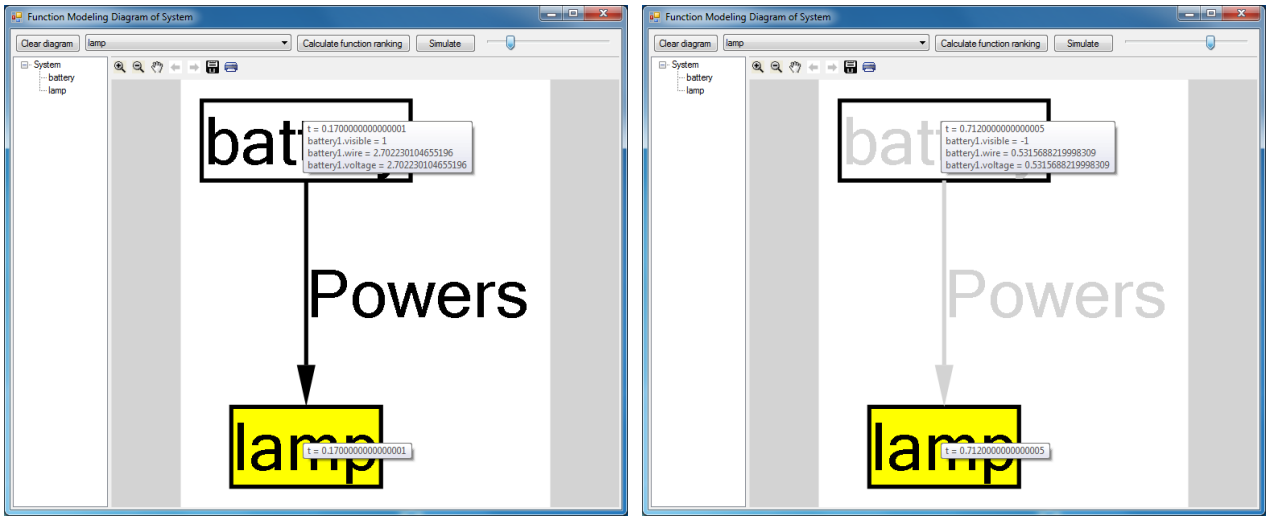


Figure 2. Code editor for component



a) b)

Figure 3. Showing simulation results and visibility changing

Workflow. The idea is to add more detail (both quantitative and qualitative) to 'normal functions' by adding Modelica code (example: the function 'to heat' can be detailed with a model about how the heating is performed and how much heat energy is transmitted per second). A simple right-click on a function allows editing its Modelica code (see Fig.2). The code for a component is separated between functions. However, it is not needed to write code for all functions – it is up to the user how much detail and realism he wants to add.

Example. The code in Fig.2 lets the battery voltage go down as a function of time. The visibility of the battery and its function is then made dependent on this voltage: when the voltage is too low, the "powers" function is not performed anymore and therefore dimmed in the FM. This is a very simple model to show that it is possible to make quick mock-ups of dynamic FMs with a few lines of code. A more advanced model would make the voltage dependent on the charge of the battery, etc. N.B.: In reality there might be a power cord that connects the battery and lamp, but for demonstration purposes this example is kept as simple as possible. The goal is to demonstrate the principle of the integration of simulation results.

When closing the editor, the OMC will check the Modelica code for errors. Storing erroneous code is impossible. Clicking the 'Simulate'-button lets the OMC calculate values for all model variables. Each variable is coupled to a FM component through its Modelica model. When sliding the time slider (top right of the FM diagram window), the values of the variables are shown at the component they belong to. Components can be hidden based on their special variable 'visibility' (see Fig.3), which can be related to any variable. This is another example of integrating mathematics with FM.

Ideas. When a system's objective would be quantified (for example, in terms of an error function that would need to be minimized), the contribution of each function to that error function could be evaluated. This way, functions could be ranked objectively on their usefulness or harmfulness. A comparison between this way of ranking and 'traditional' function ranking would be a subject for another study.

Another possibility that rises from adding physics calculations to FM, is the detection of changes within materials. Unfortunately, Modelica does not yet do this automatically: Modelica supports the modeling of two-phase fluids (gas and liquid) through the Modelica.Media package, but it is up to the user to put support for this in his model. There are several models for e.g. water, and the simplest models only support one phase. However, software could look up the material phase from phase tables using the pressure and temperature simulation results.

Conclusions

The paper continues earlier works on Function Modeling automation. We introduce the concept of dynamic function modeling and the way it can be assisted by mathematical modeling. The concept is illustrated by the software prototype. It is shown **how** Function Modeling can be integrated in SolidWorks and be assisted by Modelica in this paper. The full size example of the practical implementation of this approach as well as the comparative research of the suggested approach and "human built solution" are suggested for future research.

Reference List

1. J. Hirtz, R.B. Stone, D.A. McAdams, S. Szykman, and K.L. Wood. A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts. NIST Technical Note 1447, February, 2002 (available at http://www.mel.nist.gov/msidlibrary/doc/szykman_RED.pdf, visited 20 Apr 2012)
2. S.Litvin, N.Feygenson, O.Feygenson. Advanced Function Approach. Proceedings of the 10th ETRIA World TRIZ Future Conference 2010 – 3-5 November, Bergamo-Italy. p. 79-85
3. A.G. Kashkarov. Substance-Energy Conversions in Engineering Systems. Methodology for the Construction and Analysis of Models. Abstract of a thesis for the "TRIZ Master" degree, 2009. available at http://www.triz-summit.ru/ru/section.php?docId=4490, visited 20 Apr. 2012)
4. L.S.Chechurin, W.W.Wits, H.M.Bakker, T.H.J.Vaneker. Introducing trimming and function ranking to SolidWorks based on function analysis. In Proceedings of the TRIZ Future Conference, Dublin, Ireland, pp. 215-225, 2-4 November 2011
5. Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling (available at https://modelica.org/documents/ModelicaSpec32Revision1.pdf, visited 15 Apr. 2012)
6. John C. Mitchell. Concepts in Programming Languages. Cambridge University Press, 2003, ISBN 9780521780988 (available at http://books.google.com/books?id=7Uh8XGfJbEIC visited 15 Apr. 2012)
7. Documents Associated with CORBA, 3.2. Release date: November 2011 (available at http://www.omg.org/spec/CORBA/Current/ visited 15 Apr. 2012)

8. Sebastien Lorion, “A Fast CSV Reader”, last update 9-nov-2011, http://www.codeproject.com/Articles/9258/A-Fast-CSV-Reader, visited 15 Apr. 2012
9. A Pop, P. Fritzson. The Modelica Standard Library as an Ontology for Modeling and Simulation of Physical Systems, Internal Report, August 2004. Available at http://www.ida.liu.se/~adrpo/reports/adrpo-petfr-WS-OSEA.pdf, visited 15 Apr. 2012
10. S. Husig and S. Kohn. Computer Aided Innovation – State of the Art from a New Product Development Perspective, *Computers in Industry* 60 (2009) 551–562
11. H.M.Bakker, L.S.Chechurin, W.W.Wits. Integrating TRIZ function modeling in CAD software, In Proceedings of the TRIZfest 2011 Conference, St. Petersburg, Russia. 21-23 July 2011, pp 18-29.