2. Scientific Information on the Development of a General Modeling Methodology for Variable Structure Systems.

2.1. Summary

The demands on physical systems modeling and simulation (M&S) environments have grown dramatically in recent years. Ever more complex engineering systems are being designed that can only be analyzed by means of modeling and simulation, as their highly nonlinear behavior excludes the use of more analytical closed-form techniques. System designers demand M&S environments that allow them to plug models of system components together in the same way, as the real components are being assembled in the manufacturing plant. Component models ought to be delivered by the component manufacturers together with the components themselves. Models generated by different manufacturers are supposed to be able to be connected seamlessly and simulated reliably and efficiently. The object-oriented physical systems modeling paradigm makes this possible.

An international standard committee, the Modelica Association, designed in the late 1990s a new physical systems modeling language, called Modelica. Modelica is truly object-oriented, i.e., it enables modelers to connect sub-models topologically and hierarchically. A number of companies have meanwhile adopted the Modelica technology. Large-scale system models, e.g. describing the dynamics of cars, have been encoded in Modelica that consist of several hundreds of thousands of lines of code. These models have proven to be as run-time efficient as the best manually encoded models of the past.

Many contemporary models contain changes in their structure. Such systems are called variable structure systems. To express structural changes, a corresponding modeling language has to meet certain requirements. The language must support discrete events and hence support hybrid modeling, since structural changes clearly represent a discrete event. Furthermore, relations between variables or sub-models must be allowed to be stated in a conditional form, so that the structure can change depending on time and state. In addition, variables and sub-models should be dynamically declarable, so that the corresponding instances can be created, handled, and deleted at run time.

Unfortunately, Modelica meets these requirements only partly and provides only very limited means for the description of such models. All variables are fully declared statically, and there are no tools for adding or removing variables or even sub-model instances at runtime. Equations are allowed to change, but only in a certain fashion due to a number of technical restrictions that result from the translation process and the underlying simulator.

It is our goal to enable the convenient modeling of variable structure systems within a general modeling environment, comparable to Modelica. The environment should allow describing structural changes in a declarative manner that is compatible with the other declarative model elements. The longer term goal is to significantly extend Modelica's expressiveness and range of application. This includes the conceptual development of a variable structure modeling language, and furthermore the development of new methods and techniques for a corresponding translator program and simulation environment.

The proposed research effort is multi-faceted, as the design of an M&S environment for the successful simulation of variable structure systems touches upon a number of separate topics. On the one hand, the proposed research will include aspects that concern the field of language design. There will also be more technical parts that demand typical engineering skills for the corresponding software development. Finally, the results of the proposed research effort need to be promoted within the scientific community.

The proposed research will be undertaken by Dipl. Inf.-Ing. Dirk Zimmer as part of his Ph.D. dissertation research. Mr. Zimmer is being co-advised by Prof. Dr. Walter Gander and Prof. Dr. François E. Cellier. The proposed research effort is to be conducted at the Institute of Computational Science, Department of Computer Science of ETH Zurich.

2.2 Research Plan

2.2.1 Account of the State of the Research Field

Introduction

The demands on physical systems modeling and simulation (M&S) environments have grown dramatically in recent years. Ever more complex engineering systems are being designed that can only be analyzed by means of modeling and simulation, as their highly nonlinear behavior excludes the use of more analytical closed-form techniques. Whereas such models were encoded in the past frequently using general-purpose programming languages, such as Fortran or C, this approach is no longer sufficient, as the design lifecycle for new products becomes shorter and shorter. System designers demand M&S environments that allow them to plug models of system components together in the same way, as the real components are being assembled in the manufacturing plant. Component models ought to be delivered by the component manufacturers together with the components themselves. Models generated by different manufacturers are supposed to be able to be connected seamlessly and simulated reliably and efficiently. The object-oriented physical systems modeling paradigm [8-10,23] makes this possible.

The most prevalent physical systems M&S software on the market today is Simulink [32]. Simulink has replaced earlier M&S systems, such as ACSL [1], due to its tight coupling to MATLAB [31], the most widely used engineering systems design tool world-wide. Parameterized models can be encoded in Simulink. Their parameter values can then be fine-tuned in MATLAB, using either its Optimization Toolbox or its System Identification Toolbox. Controllers for these physical systems can then be designed also in MATLAB using e.g. the Control System Toolbox or the Fuzzy Logic Toolbox.

Yet, the modeling capabilities that are offered in Simulink are fairly primitive. Simulink offers a graphical interface based on block diagrams. Unfortunately, block diagrams are far from being object-oriented. Simulink models can thus not be assembled from sub-models in an object-oriented fashion. Consequently, Simulink is useless for large-scale systems modeling.

In order to overcome these deficiencies, an international standard committee, the Modelica Association [33], designed in the late 1990s a new physical systems modeling language, called Modelica [28,44]. The design of Modelica is heavily influenced by the visionary and pioneering work of Elmqvist [23]. Modelica is truly object-oriented, i.e., it enables modelers to connect sub-models topologically and hierarchically. The Modelica language itself is in the public domain. Several implementations of Modelica in M&S environments have meanwhile become available, most advanced and successful among them the commercial Dymola M&S environment [6,20]. There is also a decent implementation available that is in the public domain [39].

A number of companies have meanwhile adopted the Modelica technology. Large-scale system models, e.g. describing the dynamics of cars, have been encoded in Modelica that consist of several hundreds of thousands of lines of code. These models have proven to be as run-time efficient as the best manually encoded models of the past. Dynasim, the producer of Dymola, was recently bought by Dassault. Dassault plans on integrating Dymola into its line of CAD tools, and make Modelica the central modeling engine for all of its physical systems modeling needs.

General-purpose Physical Systems Modeling Language: Modelica

In the field of modeling and simulation, general modeling languages have become increasingly more accepted and widespread. In contrast to handcoded simulation programs (e.g. based on MATLAB, Fortran or C++), these languages and their corresponding environments typically offer a number of advantages:

- A well specified modeling language drastically eases the actual modeling process. The modeler can focus his or her energy on the actual model creation, and does not need to worry about the underlying simulation engine.

- The modeling language does not only enable a simulation of a given system. It also supports the organization of knowledge. Complex systems can be decomposed into simple, easily readable and understandable sub-models. This knowledge can be shared, promoted, and reused.
- Within a modeling language, certain components can be checked for consistency. This allows modeling errors to be discovered early. The modeling process is far less error prone, and the model validation becomes a feasible task.
- A general modeling language based upon abstract concepts of mathematics and computer science supports the creation of sub-models from different domains that can be coded by domain-area specialists. These sub-models can interact with each other, thereby enabling the creating of multi-domain models.
- Within a truly open modeling language and environment, models that have been created once, can be reused for a longer time period, because the model is not strictly bound to a certain programming environment.

The increasing interest in a solid standard for a general modeling language has led to the foundation of the Modelica Association in 1997. The Modelica Association [33] is a non-profit organization with the primary purpose of supporting the development of a generally accepted physical systems modeling language that forms a standard within industry and science. The association therefore provides a general modeling language that shares its name with the organization and hence is also called Modelica. It is an open language, freely available to everyone who is interested.

A recurrent Modelica conference [29] offers a platform for scientist, users and developers to present new models, exchange knowledge, and discuss ongoing developments. Regular design meetings contribute to the further refinement of the language.

The Modelica language is a general modeling language, primarily intended for modeling physical systems. At its time of introduction, the Modelica language inherited several ideas and concepts from various other modeling languages that have formerly been of relevance, including Dymola, Omola, Allan, Smile, and SIDOPS+. The language is mainly based on differential-algebraic equation (DAE) systems. Modelica offers a declarative and equation-based approach to modeling that enables the convenient formulation of models of many different kinds of continuous processes. In addition, Modelica offers means for event-driven discrete processes that adequately enable the modeling of certain hybrid systems.

The object-oriented modeling technique embraced by Modelica enables the modeler to cope with the increasing amount of complexity that characterizes contemporary models for industrial applications. The knowledge can be effectively organized, reused, and disseminated. Complexity can be portioned out on different modeling layers and can be hidden from the end user.

Major applications are found in the mechanical, electrical, and thermo-dynamical domains. Also control systems form a significant application class. The Modelica Standard Library provides model packages for all these different domains, and hence makes the corresponding knowledge practically available for all users. The Modelica language has successfully been applied to a number of industrial problems, e.g. from the automotive industry [2,19,41], power plants [21,40,42], and thermal building simulations [7,47,48]. Modelica is extensively used by several research centers, as well as by a number of important industrial enterprises. In addition, Modelica, and primarily its graphical modeling environment Dymola, have proven to be useful in teaching the concepts of physical systems modeling to engineering and science students. Dymola, and especially its graphical bond-graph libraries [11,14,15,17,50-52], are taught at ETH Zurich to MS-level students in computer science. Similar classes are offered by other universities, e.g. at TU Munich.

Different modeling and simulation environments for Modelica are on the market. Best known and most widely used among them is the commercial product Dymola [6,20] from Dynasim. The Group of Prof. P. Fritzson at the University of Linköping is currently developing the OpenModelica package [39] that contains a free open-source compiler and simulator. This still growing project becomes an increasingly powerful platform for Modelica. Another noticeable project that concerns Modelica is MOSILAB [36],

developed by the Frauenhofer Institute. MOSILAB will be a semi-commercial product. It takes first steps to handling systems with a variable structure.

It is important to note that there are also some noticeable languages and modeling environments outside the Modelica world that can handle DAEs. This concerns mainly the academic approaches HYBRSIM, ABACUSS II, and Chi, as well as the commercial products 20-Sim and gPROMS. Also VHDL-AMS deserves to be mentioned in this context.

- ABACUSS II [4,18] is an open modeling environment and simulator for mixed discrete / continuous processes.
- HYBRSIM [35] is an experimental modeling and simulation environment based on bond graphs. Its interpretative approach allows also the handling of certain causalization and index reduction mechanisms at simulation-time.
- Chi (actually denoted by χ) [27,45,46] is a highly formal language for the modeling of hybrid systems. It offers various means to control discrete and continuous processes.
- 20-Sim (pronounced Twente-Sim) [49] has been originally developed at the University in Twente (Netherlands). It originates in the modeling of control processes, and has then been further extended to a commercial product.
- gPROMS [5] represents also a commercial product based on ABACUSS.
- VHDL-AMS [3] was designed as an extension to the VHDL electronic systems hardware design language. It is capable of being used for multi-domain modeling, although its orientation towards electronic circuits is still noticeable.

Variable Structure Systems

Many contemporary models contain changes in their structure. The simulation changes from one structure to another at run time. Such systems are denoted by the term: variable structure systems [16,24,34]. There are many different motivations that lead to the generation of variable structure systems:

- The variable structure results from ideal switching processes:
  Classic examples of such structural variations include ideal switching processes in electric circuits and rigid mechanical elements that can break apart, e.g. a breaking pendulum. In these examples, the number of degrees of freedom (and in consequence the number of state variables) may change at run time.

- The model features a variable number of variables:
  This issue typically concerns social or economic simulations as well as traffic simulations that feature a variable number of agents or entities, respectively.

- The variability in structure is to be used for reasons of efficiency:
  A bended beam should be modeled in more detail at the point of the buckling and more sparsely in the remaining regions.

- The variability in structure results from user interaction:
  When the user is allowed to create or connect certain components at run time, this usually reflects a structural change. The most popular examples for this kind of simulations are computer games.

The term variable structure systems turns out to be a rather general term that applies to a number of different modeling paradigms, such as adaptive meshes in finite elements, pure discrete communication models of flexible computer networks, etc. We focus on the paradigm that is represented by Modelica: declarative models that are based on DAEs with hybrid extensions. Within such a paradigm, a structural change is typically reflected by a change in the set of variables, and by a change in the set of relations (i.e., equations) between these time-dependent variables. These changes may lead to severe changes in the model structure. This concerns the causalization of the equation system, as well as the perturbation index of the DAE system.

To express such structural changes, a corresponding modeling language has to meet certain requirements. The language must support discrete events and hence support hybrid modeling, since structural changes clearly represent a discrete event. Furthermore, relations between variables or sub-models must be allowed to be stated in a conditional form, so that the structure can change depending on time and state. In addition, variables and sub-models should be dynamically declarable, so that the corresponding instances can be created, handled, and deleted at run time.

Unfortunately, Modelica meets these requirements only partly and provides only very limited means for the description of such models. All variables are fully declared statically, and there are no tools for adding or removing variables or even sub-model instances at runtime. Equations are allowed to change, but only in a certain fashion due to a number of technical restrictions that result from the translation process and the underlying simulator.

MOSILAB offers a first approach to handling variable structure systems in a more general sense. It combines an extensive subset of Modelica with a description language for State Graphs to handle the switching between different modeling modes. MOSILAB features the dynamic creation of sub-model instances, although it does so in a very limited way. Unfortunately, the description of the dynamics of variable structure systems using state graphs represents a feasible solution for small-scale problems only. Besides, the beauty and clarity of the original Modelica language severely suffered in the process of extending the language. Furthermore, MOSILAB is currently not able to handle higher index problems in a satisfactory way.

Also other hybrid modeling languages like Chi, ABACUSS II, etc. do not provide a convenient and general solution for variable structure systems. All these languages lack sufficient means for creating models dynamically and for organizing their data in an appropriate way. Whereas such features are available in most general-purpose programming languages, an adequate implementation of these methods within a declarative modeling environment is still missing.

Dynamic Handling of DAEs

It is not only a lack of expressiveness that severely limits the modeling of structural changes in Modelica. Many limitations originate from technical restrictions of the translation process and the simulator. To understand these restrictions, it is important to take a look at the processing mechanisms for DAEs.

Differential algebraic systems of equations of object-oriented models typically represent very sparse systems. Hence various methods have been developed during the last decades to treat such sparse systems in an effective way. This includes essentially two interrelated parts: First the causalization of equations and the extraction of algebraic loops, and second the handling of structural singularities (so-called higher index problems).

The causalization enables an efficient simulation, and the corresponding extraction of algebraic loops the application of appropriate solvers. The typical causalization of DAEs corresponds to the BLT-transformation of sparse matrices. The algorithm that is mostly used for this purpose is an algorithm proposed by Tarjan [43] that is based on edge coloring of undirected bipartite graphs, where the coloring denotes a valid causalization. When algebraic loops are encountered, elaborated heuristics are applied to select a set of tearing variables for breaking the loop [25]. These heuristics were subsequently improved and have proven to be powerful enough to cope with even very large systems of equations effectively and efficiently.

However, very little research has conducted on how to treat the causalization processes (or the graph coloring, respectively), if the system cannot be assumed to be constant any longer. It is therefore necessary to review the existing methods and adopt them to the new demands.

The term “structural singularity” denotes an algebraic constraint between state variables. To handle these constraints, various methods have been developed. In particular, an algorithm proposed by Pantelides [38] for symbolic index reduction has been found to be very powerful. It is used by Dymola [6,20] and OpenModelica [39]. Unfortunately, the Pantelides algorithm fails in the case of systems with variable index [13,30], and is therefore inappropriate for dealing with variable structure systems. Other methods are more appropriate for handling such cases. These include inline integration [12,13,26,30], and projective integration methods [22].

Research Goal

Although variable structure systems form a highly important and interesting class of models, the modeling and simulation of such systems is currently only poorly supported in a general form. Higher index models occur frequently in mechanical systems through the coupling of different sub-systems. Hence in an object-oriented modeling approach, they cannot be avoided. If the system furthermore undergoes structural changes, the modeler almost invariably ends up with a variable index problem. Consequently, an M&S environment that is unable to cope with such problems will fail on a large number of industrial engineering applications. For example, let us consider the ejection seat of a military aircraft. At the time the dome cover is blown the aircraft undergoes a structural change. Once the ejection seat is released, the system undergoes a second structural change. A third structural change occurs, when the ejection seat leaves the guiding rail. Even such a simple system cannot be modeled easily and simulated reliably without a methodology capable of dealing with variable structure systems in a general fashion.

It is our goal to enable the convenient modeling of variable structure systems within a general modeling environment, comparable to Modelica. The environment should allow describing structural changes in a declarative manner that is compatible with the other declarative model elements. The longer term goal is to significantly extend Modelica’s expressiveness and range of application. This includes the conceptual development of a variable structure modeling language, and furthermore the development of new methods and techniques for a corresponding translator program and simulation environment.

It is not our target to immediately change the Modelica standard or to establish an alternative modeling language. Our scientific work is intended to merely offer suggestions and guidance for future development. This will primarily benefit the future development of Modelica, but our results may also prove useful to other modeling communities and researchers. In particular, the technical parts that concern the dynamic treatment of DAEs is expected to form a valuable contribution to the state-of-the-art of object-oriented physical systems modeling. Our independence enables us to focus on the conceptual work without being restricted by practical demands of an urgent realization. We are but a small research group, and consequently, our contributions must be fairly well focused in order to be useful. Hence it is not our intention to develop a fully workable and functional product as part of this research proposal. The final implementation will have a prototypical character only with the purpose of demonstrating the principal functionality of our newly developed algorithms and techniques. A full integration of these new algorithms and techniques into the framework of the Modelica language may be pursued at a later time.

2.2.2 Account of One's Own Research in the Field

The proposed research will be undertaken by Dipl. Inf.-Ing. Dirk Zimmer as part of his Ph.D. dissertation research. Mr. Zimmer is being co-advised by Prof. Dr. Walter Gander and Prof. Dr. François E. Cellier. The proposed research effort is to be conducted at the Institute of Computational Science, Department of Computer Science of ETH Zurich.

Walter Gander was born May 24, 1944. He is a citizen of Saanen/BE. He received his Diploma degree in Mathematics from ETH Zurich in 1968. He stayed on as Assistant to Prof. H. Rutishauser at the newly founded Institute of Computer Science, and received his Ph.D. degree in Mathematics in 1973 under the guidance of Prof. P. Henrici. From 1973 to 1987, Dr. Gander was Professor of Numerical Analysis and Computer Science at the Engineering College of the Neu-Technikum Buchs. In 1977/78, he was a Visiting Scientist at Stanford University. Dr. Gander completed his Habilitation in 1979 and became Privatdozent of Numerical Analysis at ETH Zurich. From 1987 to 1991, he served as Associate Professor and since 1991 as Full Professor of Computer Science at ETH Zürich. From 1989 to 1997, Dr. Gander was Head of the Institute of Scientific Computing. From 1990 to 1992, he served as Chairmen of the Faculty of Computer Science. From 1989 to 1991, he was Head of the Swiss Supercomputer Initiative for acquiring the national supercomputer in Manno. From 1997 to 2001, he served as Chairman and Director of Studies of the Department of Computer Science. Dr. Gander was granted three sabbatical leaves to Stanford University in 1984, 1993, and 2006.

Dr. Gander's primary research interests concern scientific computing, i.e., studying the solution of problems with the use of mathematics and computation, numerical linear algebra, and parallel computing. Dr. Gander was a member of the SIAM Working Group on Computational Science and Engineering Education and helped define the curriculum for that field. He is also one of the fathers of the new curriculum for CSE at ETH. The book *Solving Problems in Scientific Computing using Maple and MATLAB*, Springer-Verlag, by W. Gander and J. Hrebicek became a bestseller. It has been re-printed in 2005 in its fifth edition and has been translated to Chinese, Portuguese, and Russian.

Dr. Gander's five most important publications are:

1. Gander, W. (1981), "Least Squares with a Quadratic Constraint," *Numerische Mathematik*, **36**, pp.291-307.

2. Gander, W. (1990), "Algorithms for the Polar Decomposition," *SIAM J. on Sci. and Stat. Comp.*, **11**(6), pp.1102-1115.

3. Gander, W., G.H. Golub, and R. Strebel (1994), "Least-Squares Fitting of Circles and Ellipses," *BIT*, **34**, pp.558-578.

4. Gander, W. and D. Gruntz (1999), "Derivation of Numerical Methods Using Computer Algebra," *SIAM Review*, **41**(3), pp.577-593.

5. Gander, W. and W. Gautschi (2000), "Adaptive Quadrature Revisited," *BIT*, **40**(10), pp.84–101.

François E. Cellier was born July 30, 1948. He is citizen of Dübendorf/ZH and La Neuveville/BE. He received his Diploma degree in Electrical Engineering in 1972, his Postdiploma degree in Automatic Control in 1973, and his Ph.D. degree in Technical Sciences in 1979, all from ETH Zurich. Dr. Cellier stayed at ETH Zurich until 1983 as Lecturer of Simulation. He then moved to the United States, where he worked at the University of Arizona as Professor of Electrical and Computer Engineering from 1984 until 2005. He served as Head of Computer Engineering from 1997 until 2000. He was Director of Undergraduate Studies of the Electrical and Computer Engineering Department from 2001 until 2004. He requested early retirement in 2005, and became Emeritus Professor of the University of Arizona. Dr.

Cellier then returned to his home country of Switzerland, where he accepted a position with the Computer Science Department of ETH Zurich.

Dr. Cellier's main scientific interests concern modeling and simulation methodologies, and the design of advanced software systems for simulation, computer-aided modeling, and computer-aided design. Dr. Cellier has authored or co-authored more than 200 technical publications, and he has edited several books. He published a textbook on *Continuous System Modeling* in 1991 with Springer-Verlag, New York. This book became soon the standard textbook for physical systems modeling world-wide. In this book, Dr. Cellier advocated the use of object-oriented modeling and popularized the Dymola language for object-oriented modeling. The success of this book led to the creation of Dynasim AB, a Swedish company devoted to the design of object-oriented modeling software, and more indirectly, to the later creation of the Modelica language. Dr. Cellier published his second textbook on *Continuous System Simulation* in 2006 also with Springer-Verlag, New York.

Dr. Cellier served as General Chair or Program Chair of many international conferences, and served from 2004 until 2006 as President of the Society for Modeling and Simulation International. He was keynoter at numerous international conferences, most recently at the 2005 Asia Simulation Conference in Beijing, at the 2006 European Simulation Multi-conference in Bonn, at the 2006 Maplesoft Conference in Waterloo (Canada), and at the 2007 Eurosim Conference in Ljubljana. He received in 1979 the *Silver Medal of ETH* for his Ph.D. Dissertation entitled *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools*. He received in 1996 the *Silver Medal of the City of Lille* for his scientific contributions to the fields of continuous system modeling and simulation. In 2004, he was elected *Fellow of SCS*. He received in 2005 an award from the Modelica Association for his *bond graph library* (1st rank). In 2006, he was elected life member of the *Gelehrte Gesellschaft von Zürich*.

Dr. Cellier's five most important publications are:

1. Cellier, F.E. (1991), *Continuous System Modeling*, Springer-Verlag, New York, 755p.

2. Cellier, F.E. and E. Kofman (2006), *Continuous System Simulation*, Springer-Verlag, New York, 643p.

3. Cellier, F.E., H. Elmqvist, M. Otter (1995), "Modeling from Physical Principles," *The Control Handbook* (W.S. Levine, ed.), CRC Press, Boca Raton, FL, pp.99-108.

4. Elmqvist, H., M. Otter, and F.E. Cellier (1995), "Inline Integration: A New Mixed Symbolic/Numeric Approach for Solving Differential-algebraic Equation Systems," keynote presentation, *Proc. ESM'95, SCS European Simulation Multi-conference*, Prague, Czech Republic, pp.xxiii-xxxiv.

5. Otter, M., H. Elmqvist, and F.E. Cellier (1996), "Relaxing: A Symbolic Sparse Matrix Method Exploiting the Model Structure in Generating Efficient Simulation Code," keynote presentation, *Proc. Symposium on Modelling, Analysis, and Simulation*, CESA'96, IMACS Multi-conference on Computational Engineering in Systems Applications, Lille, France, vol.1, pp.1-12.

2.2.3 Detailed Research Plan

The proposed research effort is multi-faceted, as the design of an M&S environment for the successful simulation of variable structure systems touches upon a number of separate topics. On the one hand, the proposed research will include aspects that concern the field of language design. There will also be more technical parts that demand typical engineering skills for the corresponding software development. Finally, the results of the proposed research effort need to be promoted within the scientific community. Thus, the concrete research plan includes three major parts:

- conception and full definition of a modeling language,
- development of a corresponding translator and simulation system, and
- evaluation and promotion of the new modeling methodology.

The following sections treat each of these parts in more detail.

Conception and Full Definition of a Modeling Language

The conceptualization of dynamic processes appears to be a difficult task for the human brain (at least for its conscious part). Hence methods have been developed that enable a modeler to describe a dynamic process by offering a static description. DAEs are such a static description and represent one of these methods. Such methods for the mapping of dynamic processes onto static descriptions have proven very useful, since they lead to a better conceptualization of the model. The model becomes more self-contained, because it represents a valuable semantic entity, even without being interpreted by a computer program. In fact, the mapping of a dynamic process onto a static description is exactly what modeling is essentially all about.

Hence it is in the nature of a good modeling language that it is has strong declarative character, if it is not even based entirely upon declarative principles. The declarative character of a modeling language enables the modeler to concentrate on *what* should be modeled, rather than forcing him or her to consider, *how* precisely the model is to be simulated.

The modeling of variable structure systems generates the need for declarative methods to express the creation and removal of variables and even complete model instances during the simulation. Currently such methods have not been sufficiently developed for use in object-oriented modeling languages. Previous approaches to variable structure systems modeling are more in line with imperative techniques of programming. Unfortunately, they do not get along well with the synchronous data flow principles [37] embraced by declarative modeling languages, such as Modelica.

It is our goal to develop declarative methods for the static description of variable structure systems, and integrate them into the framework of a declarative modeling language. Like in Modelica, we shall provide means for declaring synchronous, non-causal relations between variables (i.e., equations). As an extension to Modelica, we furthermore offer a convenient way for declaring asynchronous, causal transmissions from one variable (or sub-model) to another. All of these declarations can be grouped in an almost arbitrary fashion. These groups of declarations may be activated or deactivated in accordance with conditions, events or predetermined sequences.

Unlike in Modelica, also the declaration of variables and sub-models can occur at the beginning of each group or subgroup. Since these groups can be stated in a conditional form, variables and sub-models may be dynamically created and deleted at run time. Hence instance creation and deletion does not need to be stated in the (typical) imperative form. It results from the activation and deactivation of declarative groups. The dynamically created objects can be handled in an unambiguous way by the declaration of asynchronous transmissions.

The new language is intended to be similar in character to Modelica. The similarity will ease the understanding of our work and help the acceptance of our approach. On the other hand, it is not meaningful

to emphasize too much on the similarity, i.e., to make the new language backward compatible to Modelica. A number of fundamental assumptions of Modelica cannot be upheld anymore by the introduction of variability in structure, and therefore a direct extension of Modelica does not represent a feasible approach.

Like Modelica, the new language should be well-structured, easily readable, and intuitive to understand. The language should provide various object-oriented tools that enable the efficient handling of complex systems. Although the language may not be fully implemented within the framework of the proposed research effort, we shall provide a full and extensive definition of the language, both in terms of its grammar and its semantics. This will enable other researchers to reproduce our concepts, even in detail. Whereas the definition of the grammar will clearly be formal, the definition of the semantics will only be describable in a semiformal way. Special emphasis will be placed on properly and precisely defining and describing the complex interactions between the discrete and continuous sub-processes.

Development of a Corresponding Translator and Simulation System

The proposed implementation of the modeling language is of an exploratory character only. It intends to concentrate on the fundamental use and functionality of the language. To keep the implementation effort within feasible limits, considering that the proposed research effort is to be completed by a single Ph.D. student as part of his dissertation research, we must shun away from a complete implementation at this time, and focus the development effort on the most essential core parts only. Even though such an implementation won't be able to support the language in its entirety, it will nevertheless be able to demonstrate and document the principal functionality of the proposed methodology. It is important to notice that the development of such a simulation system demands more than just good engineering practice. It requires quite a bit of scientific research. Restricting the implementation effort will help us focus on the more important scientific aspects of the proposed research.

A structural change represents a discrete and causal process. Hence the corresponding discrete elements of our language are planned to be implemented in the first place. Since these processes are all discrete and causal, the development of a corresponding simulation system implementing such processes causes no fundamental difficulty. However the mapping from the declarative statements describing the events, variables, and sub-models into the imperative framework of the underlying simulation system is expected to be a non-trivial, but manageable, task.

The synchronous processing mechanisms for the algebraic equation systems (especially continuous DAEs) are then implemented on top of this flexible framework. This generates the need for an efficient and dynamic treatment of DAEs, a topic that has not been researched extensively until now.

As stated previously, the standard causalization mechanisms are based on Tarjan's graph-coloring algorithm [43]. Since equations can be modified, added or removed at run time, the coloring needs to be dynamically updated, and also the tearing variables for the algebraic loops [25] need to be re-determined at run time. In the worst case scenario, the update may affect the entire equation system, but in most cases, only a small subpart of the equation system needs to be refreshed.

Since the worst-case scenario invariably results in a slow update, it is important that the update algorithm refrains from performing unnecessary or redundant operations. Only those updates that are inevitable should be executed. The algorithm should therefore exhibit a behavior that preserves the existing causalizations as long as they can be assumed to be appropriate, even if some prerequisites of the causalization are temporarily not fulfilled during the updating mechanism.

An optimistic algorithm has been sketched for this purpose that seems to offer a promising approach. The algorithm introduces a hysteretic mode into the coloring process that prevents existing causalizations from being deleted in an overhasty fashion if they intermediately lose their causal root. The new algorithm succeeds in avoiding unnecessary updates and can handle various changes simultaneously.

Also structural singularities need to be handled in a dynamic fashion. Inline integration [12,13,26,30] has shown to offer a feasible approach. By partially inlining certain integrator equations, we expect to obtain a feasible solution for up to index-2 problems. Higher index problems, especially index-3 problems that occur frequently in mechanical systems, can principally be solved by inline integration as well. However, we expect to run into numerical difficulties if the system itself is not stable.

The implementation of the underlying simulation system brings about further problems, such as the efficient solving of linear and non-linear systems of equation, as well as the efficient and precise triggering of state-events. However, we shall refrain from putting considerable effort into these topics and stick to simple and robust solutions. An extensive number of elaborated solutions have already been developed for such purposes, and most of them are only marginally affected by the new set of demands that are caused by the variability in structure.

Evaluation and Promotion of the New Modeling Methodology

The resulting modeling language and its corresponding software environment together promote a modeling methodology. In the final stage, this new methodology shall be carefully validated and compared to other established modeling methodologies. This validation and comparison process shall address the following topics:

- **Readability of the language:**
  - What are the mechanisms offered by the language providing means for structuring the input?
  - Does the new modeling language resemble our natural language and/or common mathematical nomenclature and is this resemblance really helpful for formulating models or does it rather mislead?
  - Does the new modeling language lend itself to an embedding into a graphical modeling and simulation environment?
- **Complexity of the grammar:**
  - How many terminal symbols are used within the language?
  - What is the number of grammar rules and how many EBNF operators have to be used?
- **Expressiveness of the language:**
  - What are the prerequisites that a system has to fulfill so that it can be modeled in our language?
  - What are the underlying assumptions made by the language?
  - In which situations does the modeling language generate awkward models due to a lack of expressiveness?
- **Support for knowledge organization:**
  - How effectively can code be reused?
  - What kinds of polymorphisms are available?
  - How effectively can complexity be hidden from the user?
- **Support for error prevention and recovery:**
  - Can understandable syntax errors be generated?
  - Can understandable run-time error messages be generated?
  - Is it possible to foresee potential run-time errors at compile time and generate meaningful warning messages?
- **Requirements of the translator and simulation systems:**
  - Which methods need to be provided by the translator?
  - Which methods need to be provided by the simulator?
  - Which methods should be shared by translator and simulator?
  - Does the language facilitate the generation of efficient simulation run-time code?

Experience tells us that it is always hard to promote a new modeling methodology to the scientific community, because it requires that researchers accept and get themselves acquainted with a foreign mindset. This represents a time-consuming and often difficult task. Hence it is of major importance that we provide enough material of a didactic nature that eases and clarifies the access to our new concepts.

To this end, a number of application examples are needed to present and demonstrate the modeling concepts that can be expressed in our new modeling language. Whereas most of these application codes will be of an academic nature only, it is important to also offer demonstration models of industrial size and realism. Their purpose is to motivate also people from industry to consider our approach.

The resemblance of our approach to the widely accepted Modelica language will be of further help in this respect. Our models will look similar to already familiar Modelica models, and the adoption to our methodology will thus be much easier for people within the Modelica community.

2.2.4 Timetable, Milestones, and Deliverables

Phase 1: April 1, 2006 – September 30, 2006.

This is the preparation phase. The available literature pertinent to the proposed research has been studied.

Phase 2: Oct 1, 2006 – September 30 2007

This is the phase of the language design of our new modeling language.

Milestone 1: Full definition of the new modeling language. The grammar is formally defined by the corresponding EBNF definition, and a parser is being provided. The semantics are precisely defined in textual form. The resulting demands on the simulation system are specified and analyzed, and a sketch of the compilation process is being provided, albeit only on paper. A set of examples demonstrates the new modeling methodology and aids the understanding of the semantics.

Phase 3: Oct 1 2007 – September 30 2008

This is the phase of the language implementation. A prototypical simulation environment and a corresponding prototypical translator are being created.

Milestone 2: The partial implementation of the modeling language offers sufficient support for the discrete modeling parts. The synchronous update of the equations is partially supported at this time. The causalization of equations is dynamic, and the system is already able to cope with linear algebraic loops. A method addressing the dynamic handling of variable-index systems takes care of index-2 problems and preferably also some classes of higher-index problems. The implemented algorithms are well documented and presented in scientific papers.

Phase 4: Oct 1 2008- September 30 2009

Since the proposed research is primarily of a conceptual nature, it is highly important that a significant effort be put into the promotion of the obtained results and concepts among the scientific community. It is therefore necessary to deliver more than just technical descriptions in form of scientific articles. Additional material of a more didactic character as well as a number of demonstration examples clarifying and concretizing our new modeling methodology are of equal importance. For people working in industry it is also important to recognize that the advocated concepts scale up conveniently to realistic industrial problem sets.

Milestone 3: A number of academic examples that present the major capabilities of our modeling methodology are being developed and implemented. Some larger and more realistic examples demonstrate the applicability of the new modeling methodology to dealing with real-world problems. The prototypical implementation is being enhanced sufficiently to support the demonstration examples. In addition, the modeling methodology is being promoted on the world-wide web by suitable documentation of a tutorial nature.

During the second half of the project, we plan on disseminating our results. Scientific publications in the form of journal articles and conference papers shall be written and presented within the Modelica community and in various journals in the field of modeling and simulation (e.g. Simulation, ACM Transactions on Modeling and Computer Simulation, Mathematical & Computer Modelling of Dynamical Systems, Simulation Modelling Practice and Theory), as well as at conferences relating to object-oriented computer languages (e.g. ECOOP, OOPSLA). Furthermore the complete source code as well as the executables will be distributed freely on the Internet. Finally, the complete grammar and semantics will be released and made freely available on the Internet.

2.2.5 Significance of the Planned Research to the Scientific Community and to Eventual Potential Users

Having available a general modeling methodology for variable structure systems will result in a number of important advantages. Such a general modeling methodology enables the convenient capture of knowledge concerning variable structure systems, and provides means for organizing and sharing that knowledge both by industry and science. Furthermore, a general modeling language is a valuable tool for engineering and science education. Within the Computer Science Department of ETH Zurich, we already offer an MS-level class on physical system modeling that uses Modelica [28,44] and especially its three bond graph libraries [11,14,15,17,50-52] as pivotal tools for knowledge communication.

In concrete terms, our methodology is intended to aid the further extension of the Modelica framework. This benefits primarily the prevalent application areas of mechanics and electronics.

- Ideal switching processes in electronic circuits (resulting from ideal switches, diodes, and thyristors) can be more generally modeled. Occurring structural singularities can be handled at run time.
- The modeling of ideal transitions in mechanical models, like breaking processes or the transition from friction to stiction, become a more amenable task.

In addition, our methodology may also be applicable to domains that are currently foreign to Modelica. Hence our work might help widen the field of applications and offer a solid modeling methodology for a larger number of domains. This could concern for example:

- Hybrid economic or social simulations that contain a variable number of entities or agents, respectively.
- Traffic simulations.

Finally, more elaborate modeling techniques become feasible. For instance, multi-level models can be developed, whereby the appropriate level of detail is chosen at simulation run time in response to computational demands and/or level of interest. For example, a flight tower simulator could be developed that uses crude models for aircraft that are far away, and more refined models for aircraft that are either on the runway or in the vicinity of the airport.

Although the proposed research will be conducted at ETH Zurich exclusively, the researchers are in close and frequent contact with the M&S community in general and with the Modelica community in particular. A frequent exchange of ideas, concepts, methods, and tools will take place to ensure optimal cross-fertilization between our research group and the M&S community at large.

References

[1] Aegis Simulation, Inc. (1999), *ACSL: Advanced Continuous Simulation Language, Reference* Manual, Aegis Simulation, Inc.

[2] Andreasson, J. and M. Gäfvert (2006), "The Vehicle Dynamics Library: Overview and Applications," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.43-51.

[3] Ashenden, P.J., G.D. Peterson, and D.A. Teegarden (2002), *The System Designer's Guide to VHDL-AMS,* Morgan Kaufmann Publishers.

[4] Barton, P.I. (1991), *The Modeling and Simulation of Combined Discrete Continuous Processes,* PhD Dissertation, Imperial College, London, U.K.

[5] Barton, P.I. and C.C. Pantelides (1994), "Modeling of Combined Discrete/Continuous Processes," *AIChE J.*, **40**, pp.966-979.

[6] Brück, D., H. Elmqvist, H. Olsson, and S.E. Mattsson (2002), "Dymola for Multi-engineering Modeling and Simulation," *Proc. 2nd Intl. Modelica Conference*, Oberpfaffenhofen, Germany, pp.55:1-8.

[7] Casas, W., K. Prölß, and G. Schmitz (2005), "Modeling of Desiccant-Assisted Air Conditioning Systems," *Proc. 4th Intl. Modelica Conference*, Hamburg, Germany, Vol.2, pp.487-496.

[8] **Cellier, F.E.** (1991), *Continuous System Modeling*, Springer-Verlag, New York, 755p.

[9] **Cellier, F.E.**, and H. Elmqvist (1993), "Automated Formula Manipulation Supports Object-oriented Continuous-system Modeling," *IEEE Control Systems*, **13**(2), pp.28-38.

[10] **Cellier, F.E.**, H. Elmqvist, M. Otter (1995), "Modeling from Physical Principles," *The Control Handbook* (W.S. Levine, ed.), CRC Press, Boca Raton, FL, pp.99-108.

[11] **Cellier, F.E.** and J. Greifeneder (2003), "Object-oriented Modeling of Convective Flows Using the Dymola Thermo-bond-graph Library," *Proc. ICBGM'03, 6th SCS Intl. Conf. on Bond Graph Modeling and Simulation*, Orlando, Florida, pp.198-204.

[12] **Cellier, F.E.** and E. Kofman (2006), *Continuous System Simulation*, Springer-Verlag, New York, 643p.

[13] **Cellier, F.E.** and M. Krebs (2007), "Analysis and Simulation of Variable Structure Systems Using Bond Graphs and Inline Integration," *Proc. ICBGM'07, 8th SCS Intl. Conf. on Bond Graph Modeling and Simulation*, San Diego, CA, pp.29-34.

[14] **Cellier, F.E.** and R.T. McBride (2003), "Object-oriented Modeling of Complex Physical Systems Using the Dymola Bond-graph Library," *Proc. ICBGM'03, 6th SCS Intl. Conf. on Bond Graph Modeling and Simulation*, Orlando, Florida, pp.157-162.

[15] **Cellier, F.E.** and A. Nebot (2005), "The Modelica Bond Graph Library," *Proc. 4th Intl. Modelica Conference*, Hamburg, Germany, Vol.1, pp.57-65[1].

[16] **Cellier, F.E.**, M. Otter, and H. Elmqvist (1995), "Bond Graph Modeling of Variable Structure Systems," *Proc. ICBGM'95, 2nd SCS Intl. Conf. on Bond Graph Modeling and Simulation*, Las Vegas, NV, pp.49-55.

[1] Dr. Cellier received an award of the Modelica Association for his library (1st rank 2005).

[17] **Cellier, F.E.** and **D. Zimmer** (2006), "Wrapping Multi-bond Graphs: A Structured Approach to Modeling Complex Multi-body Dynamics," *Proc. 20th European Conference on Modeling and Simulation*, Bonn, Germany, pp.7-13.

[18] Clabaugh, J.A. (2001), *ABACUSS II Syntax Manual*, Technical Report, Massachusetts Institute of Technology, http://yoric.mit.edu/abacuss2/syntax.html.

[19] Dempsey, M., M. Gäfvert, P. Harman, C. Kral, M. Otter, and P. Treffinger (2006), "Coordinated Automotive Libraries for Vehicle System Modelling," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.33-41.

[20] Dynasim AB (2006), *Dymola Users' Manual*, Version 6.0, Lund, Sweden.

[21] Eborn, J., F. Selimovic, and B. Sundén (2006), "Modeling and Dynamic Analysis of CO2-Emission Free Power Processes in Modelica Using the CombiPlant Library," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.17-22.

[22] Eich, E. (1993), "Convergence Results for a Coordinate Projection Method Applied to Mechanical Systems with Algebraic Constraints," *SIAM J. Numerical Analysis*, **30**(5), pp.1467-1482.

[23] Elmqvist, H. (1978), *A Structured Model Language for Large Continuous Systems*, Ph.D. Dissertation, Lund Institute of Technology, Lund, Sweden.

[24] Elmqvist, H., **F.E. Cellier**, and M. Otter (1993), "Object-Oriented Modeling of Hybrid Systems," *Proc. ESS'93, SCS European Simulation Symposium*, Delft, The Netherlands, pp.xxxi-xli.

[25] Elmqvist H. and M. Otter (1994), "Methods for Tearing Systems of Equations in Object-oriented Modeling," *Proc. ESM'94, SCS European Simulation Multi-conference*, Barcelona, Spain, pp.326-332.

[26] Elmqvist, H., M. Otter, and **F.E. Cellier** (1995), "Inline Integration: A New Mixed Symbolic/Numeric Approach for Solving Differential-Algebraic Equation Systems," *Proc. ESM'95, SCS European Simulation Multi-conference*, Prague, Czech Republic, pp.xxiii-xxxiv.

[27] Fábián, G. (1999), *A Language and Simulator for Hybrid Systems*, Ph.D. Dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands.

[28] Fritzson, P. (2004), *Principles of Object-oriented Modeling and Simulation with Modelica 2.1*, John Wiley & Sons, 897p.

[29] Kral, C. and A. Haumer (2006), *Proc. 5th Intl. Modelica Conference*, *Vols.1 and 2*, Modelica Association.

[30] Krebs, M. (1997), *Modeling of Conditional Index Changes*, MS Thesis, Dept. of Electr. & Comp. Engr., University of Arizona, Tucson, AZ.

[31] Mathworks, Inc. (2006), *MATLAB User's Guide*, *Version 7.2*, Mathworks, Inc.

[32] Mathworks, Inc. (2006), *Simulink User's Guide*, Mathworks, Inc.

[33] Modelica Association (2007), http://www.modelica.org/.

[34] Mosterman, P.J. (1999), "An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages," In: *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Springer-Verlag, Vol.1569, pp.165-177.

[35] Mosterman, P.J. (2002), "HYBRSIM - A Modeling and Simulation Environment for Hybrid Bond Graphs," *J. Systems and Control Engineering*, **216**, Part I, pp.35-46.

[36] Nytsch-Geusen, C., T. Ernst, A. Nordwig, P. Schwarz, P. Schneider, M. Vetter, C. Wittwer, A. Holm, T. Nouidui, J. Leopold, G. Schmidt, and A. Mates (2006), "Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.63-71.

[37] Otter, M., H. Elmqvist, and S.E. Mattsson (1999), "Hybrid Modeling in Modelica Based on the Synchronous Data Flow Principle," *Proc. IEEE International Symposium on Computer Aided Control System Design*, Hawaii, pp.151-157.

[38] Pantelides, C. C. (1988), "The Consistent Initialization of Differential-algebraic Systems," *SIAM J. Sci. Comput*., **9**(2), pp.213-231.

[39] Pop, A., P. Fritzson, A. Remar, E. Jagudin, and D. Akhvlediani (2006), "OpenModelica Development Environment with Eclipse Integration for Browsing, Modeling, and Debugging," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.2, pp.459-465.

[40] Schimon, R., D. Simic, A. Haumer, C. Kral, and M. Plainer (2006), "Simulation of Components of a Thermal Power Plant," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.119-125.

[41] Simic, D., H. Giuliani, C. Kral, and J.V. Gragger (2006), "Simulation of Hybrid Electric Vehicles," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.25-31.

[42] Souyri, A., D. Bouskela, B. Pentori, and N. Kerkar (2006), "Pressurized Water Reactor Modelling with Modelica," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.1, pp.127-133.

[43] Tarjan, R.E. (1972), "Depth First Search and Linear Graph Algorithms," *SIAM J. on Comp*., **1**(2), pp.146-160.

[44] Tiller, M.M. (2001), *Introduction to Physical Modeling with Modelica*, Kluwer Academic Publishers, 344p.

[45] van Beek, D.A. (2001), "Variables and Equations in Hybrid Systems with Structural Changes," *Proc. 13th European Simulation Symposium*, Marseille, pp.30-34.

[46] van Beek, D.A. and J.E. Rooda (2000), "Languages and Applications in Hybrid Modelling and Simulation: Positioning of Chi," *Control Engineering Practice*, **8**(1), pp.81-91.

[47] Weiner, M., and **F.E. Cellier** (1993), "Modeling and Simulation of a Solar Energy System by Use of Bond Graphs," *Proc. 1st SCS Intl. Conf. on Bond Graph Modeling and Simulation*, San Diego, CA, pp.301-306.

[48] Wetter, M. (2006), "Multi-zone Building Model for Thermal Building Simulation in Modelica," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.2, pp.517-526.

[49] Weustink, P.B.T., T.J.A. de Vries, and P.C. Breedveld (1998), "Object-Oriented Modeling and Simulation of Mechatronic Systems with 20-sim 3.0," *Proc. Mechatronics 98*, pp.873-878.

[50] **Zimmer, D.** (2006), *A Modelica Library for Multi-bond Graphs and its Application in 3D-Mechanics*. MS Thesis, ETH Zurich, Switzerland[2].

[2] Mr. Zimmer received the *Medal of ETH* for his MS Thesis.

[51] **Zimmer, D.** and **F.E. Cellier** (2006), "The Modelica Multi-bond Graph Library," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol. 2, pp. 559-568[3].

[52] **Zimmer, D.** and **F.E. Cellier** (2007), "Impulse Bond Graphs," *Proc. ICBGM'07, 8th SCS Intl. Conf. on Bond Graph Modeling and Simulation*, San Diego, CA, pp. 3-11.

[3] Mr. Zimmer received an award of the Modelica Association for his library (1st rank 2006).