TO AUTOMATIC MODEL ABSTRACTION: A TECHNICAL REVIEW

Wilhelm Dangelmaier, Daniel Huber, Christoph Laroque, Mark Aufenanger
Chair of Business Computing, esp. CIM
Heinz Nixdorf Institute, University of Paderborn
whd|huber|laro|marka@hni.upb.de

KEYWORDS

Model Abstraction, Complexity, Validity

ABSTRACT

Discrete event models in material flow simulation are growing constantly in scope and resolution. Model abstraction is necessary to allow simulation experiments of efficient runtime. Automatic model abstraction is able to make the work of simulation experts easier. Thus only the models with highest complexity have to be created and maintained. In this paper techniques necessary for automatic model abstraction are reviewed. At the beginning, definitions complexity and validity are discussed. Following are methods to measure these model characteristics. And finally methods for abstraction and some practices are presented. Concluding the lack of a unified modeling framework and the lack of quantitative measures of abstraction results is asserted.

MOTIVATION

The complexity, size and resolution of simulation models are growing constantly in the last years, for several reasons [CPBP00]. Firstly, there is the possibility to create such huge models. The performance-cost-ratio of computers is improving constantly and the simulation tools are getting better. Modern Tools are user friendly applications which incorporate model reuse, quality visualization and object orientation. Secondly, there is the desire to include everything in one model. The vision of the "Digital Factory" leads modelers to build models of every single aspect of a factory and to combine everything in one single model. With the possibility to build huge models simulation experts include lots of detail in a model to feel safe not to miss anything important. Thirdly, high quality virtual reality animation induces much complexity because many details have to be included in the model to generate "smooth" animations.

The runtime of these huge models is bad, i.e., the simulation of a certain amount of simulation time takes a lot of real time. Simulation experiments with several replications are hardly accomplishable. The time needed to run a simulation, to get all needed input data and evaluate the results is expensive. There are several attempts to allow an efficient simulation experiment when having a huge model. One possibility is to increase the computational power by running the simulations in parallel. This can be done by parallelizing one simulation or by parallelizing the replications [DHL^{+}06].

Another possibility is to reduce the complexity of the model. In a huge model (e.g., one plant with shops A, B and C) not every shop or facility is of high importance for every specific experiment. If one experiment is done to analyze the performance of shop A, there can be other shops (B, C) with minor or negligible influence on the results of the experiment. By using submodels of lower complexity (B', C'), the overall complexity of the composed model can be reduced with few consequences to validity. To use this method there have to be at least two models for every facility, one of high and one of low complexity. The composition of a model before starting a simulation experiment is normally done manually, but Mueck [Mue05, DM04] elaborated a simulation technique that can switch submodels of different complexity during simulation on the basis of significance for the current experiment. Such a system was envisioned by Zeigler and en [Z86] imagined a long time ago.

One major drawback of this method is the need for creating and maintaining at least two models per facility. Normally models are under permanent modification. Thus these modifications have to be done repeatedly, once for every submodel and level of complexity. Automatic model abstraction is a possibility to overcome this drawback. The modeler has to create and maintain only the model of highest complexity, whereas models of lower complexity are created automatically.

When parts of model are to be abstracted, a partitioning algorithm is needed. This algorithm should group areas of the model with logical interrelation and similar geometrical position like shops, manufacturing cells, etc. Partitioning algorithms should not be discussed in this paper, refer to Fjallstrom [Fja98] as a starting point.

To benchmark the abstraction of one model, a metric is needed to compare the complexity of the original model with the created one. This metric should tell the abstraction algorithm, if the complexity was reduced in the desired amount. It is used in combination with a validity metric as a regulation. The validity metric garantees that the model behavior is not altered beyond tolerance during abstraction. The abstraction itself should be automated and flexible concerning applicability, achievable complexity and validity. In the next sections we present research works done in the field of model abstraction and supporting methods.

COMPLEXITY, LEVEL OF DETAIL, VALIDITY

In the last section the terms complexity, detail and validity were used. In this section, their definitions and importance in the field of simulation model abstraction are presented. To begin with, the complexity definition of the complexity theory is not applicative because simulation is no algorithmic problem.

Zeigler et al. [ZPK00] defines complexity and detail as the product of size and resolution of a model. Whereas the size is the number of components and the resolution is the number of states per component. Three different kinds of complexity are separated. The analytic and exploratory complexities are not relevant here, only the computational or simulation complexity is. It measures the computational resources necessary to execute the model.

Chwif et al. [CPBP00] define scope and level of detail as two components of complexity whereas scope is the same as size and level of detail is the same as resolution.

Sisti and Farr [SF98] give a detailed description for these terms. Validity or accuracy defines, how well the model behavior fits the real system. Resolution describes, in what depth minute aspects are modeled. Complexity is a term for computational aspects of model execution. Four assertions are built, which describe the relationship between these terms. Firstly, higher resolution does not necessarily imply more accuracy, secondly, neither does complexity. Thirdly, higher resolution does usually imply increased complexity and last but not least complexity is directly related to computer runtime.

Sargent [Sar05] defines three types of validity. The conceptual model validity determines, that the assumptions and theories underlying the conceptual model are correct and that the model represents the real word system reasonably in the experimental frame. Operational validity determines, that the output behavior of the model is of sufficient accuracy. Data validity determines, if the data available for model building and testing is correct and adequate. In the context of model abstraction the validity of the original model in relation to the real world system is of no interest. Therefor only operational validity has to be achieved between original and abstracted model.

For automatic model abstraction, in order to improve simulation runtime, complexity must be reduced, because complexity defines the demand for computational resources. The reduction of complexity can have negative effect on validity, i.e., an abstracted model can have a differing behavior.

Recapitulating: *Model abstraction is defined as the reduction of complexity while preserving the validity of the model in an experimental frame* [ZPK00].

MEASURING COMPLEXITY

Only a rather intuitive definition for the term complexity was given. To evaluate the results of an abstraction algorithm a complexity metric has to be defined.

Wallace [Wal87] as well as Schruben and Ycesan [SY93] developed metrics operating on graphs. Wallace used a Action Cluster Incident Graph to represent models, which are related to the better known Simulation Graphs of Schruben.

Wallace defines a Control and Transformation Metric as (1).

$$C_{W1} = \sum_{i=1}^{n} (2 * RW_i + 1.5 * W_i + R_i) * \frac{A_i}{N} \qquad (1)$$

Where the transformation complexity of one node i is defined as the weighted product of the number of variables with read and write access (RW), with read access (R) and with write access (W). The control complexity, the complexity of the interconnection of nodes, is defined as the quotient of the number of entering and leaving edges (A) and the total number of edges (N). In some tests Wallace has shown that this metric has good response in comparison to other software metrics but he did not test computer runtime.

Schruben and Ycesan developed several metrics. The metric (2), as the most sophisticated, showed best regression with computer runtime.

$$C_3 = \sum_{v \in V(G)} C(v) + \sum_{v,w \in V(G)} I(v,w) \qquad (2)$$

This metric summarizes for all nodes v the complexity of the Nodes C(v) and the complexity of the arcs I(v,w). Analog to Wallace, C(v) is the transformation complexity and I(v,w) is the control complexity. In (2) C(v) can represent the number of state variables of a vertex and I(v,w) the number of edges between nodes v and w.

These metrics are promising, because they work strictly on models and no simulation is necessary to acquire data. But it for complexity reduction a metric is necessary with a very good correlation with simulation runtime. Runtime analyses of and complexity measurements of several simulation models should be accomplished to validate the metrics. With runtime analyses the event complexity $C(v)$ could be refined, which seems profitable.

MEASURING VALIDITY

There are various techniques for model validation (see Sargent [Sar05] for an overview), which can be objective or subjective, and with or without formal statistical procedures [LK00]. Often applied validation techniques are animation, graphical comparison of data, confidence intervals or hypothesis test.

Validation in model abstraction is used to measure the differences between the behavior of original and simplified model. The Classification of Sargent sorts this validation problem into the class "Comparison using statistical tests and procedures", because the systems are observable and a objective approach is desired. Further on, having both models and their input and output data, trace-driven validation is possible, case (iii) of Kleijnen [Kle99].

Automatic model abstraction requires additionally an automatic validation procedure. Martens et al. [MPK06] introduces a approach based on fuzzy set theory. Methods of artificial intelligence are used to measure a degree of similarity between two models. The disadvantages of the approach, the necessity to run both models with the same inputs and the computational effort, are of low impact here, because the data is available and long runtime for abstraction is no problem.

When abstracting submodels, i.e., parts of one model, the interfaces between these submodels define the data type of input and output data. The above described methods use numerical data, but objects or token are also possible in- and outputs. Comparing the in- and output trajectories can be used to compare these models. If the tokens are uniform, numerical methods are applicable, but if classified or unique tokens are used, it is much more complicated, like in Sarjoughian and Zeigler [SZ96]. The trajectories can differ in time of occupance or quantity of token, but also in type.

METHODS FOR MODEL ABSTRACTION

In this section methods for abstraction should be presented. Starting with some basic thoughts about how complexity could be tackled and continuing with some developed practices. Finally some efforts in metamodeling are presented.

Basics

Zeigler et al. [ZPK00] proposes several possibilities to reduce complexity.

Aggregation Combining a partition of components into a single object. The single object represents the input/ output behavior of the partition. The number of components in the model is reduced.

Omission Leaving out objects, variables or interactions thus the number of components in the model is reduced.

Linearization Representing behavior around an operating point as a linear system.

Deterministic/ stochastic replacement
Changing deterministic variables to stochastic is effective, if the computation of deterministic value is more complex than the computation of the stochastic distribution. Vice versa, the complex calculation of a stochastic value can be avoided when using a deterministic value, e.g., the mean.

Formalism transformation Using a a different formalism which is more suitable for the problem. E.g., using differential equations or metamodels instead of discrete event simulation.

When using these methods, the effect on validity must always be kept in mind. While Aggregation seems quite complicated to implement, it has little effect on validity. On the other side all the other methods can have enormous effect on it.

Practices

Sevinc [Sev90] uses homomorphisms to abstract models. Homomorphisms for model abstraction are introduced by Foo and Zeigler [Foo74, ZPK00]. A homomorphism is a mapping from one model M to another M' such the behavior of these two models is equivalent. A weakened homomorphism can be defined to determine the accepted behavioral tolerance in simplification.

The main idea is to transform coupled models to atomic models. A coupled model consists of several atomic models or other coupled models. A actual simulation model can be viewed as a tree, where the leafs are atomic models. When simplifying the model the size of the tree is reduced. To do this atomic models have to be created with behavior similar to the substituted coupled one. While simulating a model, the inputs, outputs and transitions are observed. For every element a homomorphism is defined to lump the observed states. Additional functions can be defined to lump the in- and output. The lumped data is stored and a code generator creates the new models.

Another approach is done by Sevinc and Foo [SF90]. In this work, states are being clustered and these clusters form new aggregated states. The state set is reduced and the model abstracted. All occurred states during simulation are stored in a database. States with similar characteristics are defined to have a high proximity and low distance. A distance function is defined and applied to every pair of states. States with low distances are clustered and the cluster is represented by the state with the highest frequency of occurrence. In an improvement phase the built clusters are optimized by moving states from one cluster to another. The method of clustering and improvement is similar to standard graph partitioning. The output, time advance and transition functions are created by using probabilities calculated using the observed simulation data.

More recently Brooks and Tobias [BT00], Rose[Ros99, Ros00] and Johnson et. al.[JFM05] presented works on model abstraction in production systems. The approach is similar in all three references: After identifying the bottleneck of the system, all elements predecessors and successors of the bottleneck are abstracted, but the bottleneck itself is preserved. Abstraction was accomplished by substituting all elements by simple delays. Rose asserts that this technique preserves validity quite good, as long as no dispatching rules more complex than FIFO are used in the model. Johnson et. al. identified the bottleneck as the machine with highest utilization and uses a correlation coefficient for validity measurement. Instead of abstracting all elements but the bottleneck, the method of Johnson et. al. also contains a iterative process by only abstracting the elements with lowest utilization until validity is critical. This method seems suitable for the automation of abstraction, but this was not explicitly examined by the author. Another drawback is necessity to run sample simulations and to freeze the production program, thus no universally valid abstraction is achieved.

Metamodels

In the two above described methods the structure and formalism of the simulation models is not changed. When using metamodels the structure of the discrete event model ist lost. A metamodel is a surrogate model for the original model with a similar input-output behavior. There are many methods or building metamodels [PCG00] like polynomial regression, surface response, methods of artificial intelligence and some more.

Sarjoughian and Zeigler [SZ96] introduce one interesting approach in the field of artificial intelligence. The input and output trajectories of several test simulation of the system are stored. A reasoner is used to predict segments of an output trajectory to a given segment of an input trajectory. If the given input segment is not in the stored data, the algorithm can predict an equivalent output segment. To achieve this a framework, well-defined assumptions and a non-monotonic reasoner are created.

CONCLUSION

The purpose of automatic model abstraction was defined as a technique to create lumped models with better runtime than the original ones. Therefor the complexity of one model has to be reduced. The validity of the abstraction should not be reduced beyond a certain tolerance level. In this paper, the necessary techniques to achieve a regulated abstraction were presented. These techniques are complexity metrics, validity measurement and model abstraction.

One identified general problem is the use of different modeling formalisms. Complexity metrics were defined on simulation graphs; practices in model abstraction used DEVS. Thus it is necessary to unify all necessary techniques in one formalism.

The presented complexity metrics were tested by their authors and showed good correlation to computer runtime. The evaluation of event complexity can be improved by analyzing the program code of events in a more detailed way. The impact of program loops and properties of variables, like data types and random distributions, on computer runtime should be investigated and, if necessary, integrated in a complexity metric.

Fully automated model validation techniques exist, but if abstracting submodels, the interfaces between submodels could be a problem. The use of colored tokens or similar objects requires the development of new validation techniques. The use of methods of artificial intelligence seems to be the most capable approach.

The basic approaches for model abstraction are exhaustive. But the older parts of the practices are badly described and results are not satisfactorily tested. The newer practices described are promising. A regulation of validity was implemented, but a regulation of complexity is missing. Metamodeling is a powerful technique for abstraction but the structure of the model is lost. The complexity of a metamodel can usually not be measured with a metric specialized on one modeling formalism. The interfaces between submodels limit the applicable metamodeling techniques. The use of methods of artificial intelligence is the most interface independent ones and are not restrained on numerical data.

Metamodeling as well as the presented practices in abstraction need training data. Because of the stochastic nature of simulation, many simulations have to be run to collect good training data. Abstraction techniques, which operate only on the static model, seem worth studying. Using such techniques would reduce the number of simulation runs for abstraction significantly. The stochastic behavior of a simulation is defined in its model. When analyzing the model for abstraction, the reasons for this behavior can be found and handelnd.

No satisfying technique for regulated automated model abstraction could be found and further research in this field is necessary.

REFERENCES

[BT00] Roger J. Brooks and Andrew M. Tobias. Simplification in the simulation manufacturing system. *International Journal of Production Research*, 38(5):1009–1027, 2000.

[CPBP00] Leonardo Chwif, Marcos Ribeiro Pereira Barretto, and Ray J. Paul. On simulation model complexity. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 449–455, 2000.

[DHL$^+$06] Wilhelm Dangelmaier, Daniel Huber, Christoph Laroque, Mark Aufenanger, Matthias Fischer, Jens Krokowski, and Michael Kortenjan. d^{3}fact insight goes parallel - aggregation of multiple simulations. In Thomas Schulze, Graham Horton, Bernhard Preim, and Stefan Schlechtweg, editors, *Simulation und Visualisierung 2006*, pages 79–88. SCS European Publishing House, 2006.

[DM04] Wilhelm Dangelmaier and Bengt Mueck. Using dynamic multiresolution modelling to analyse large material flow system. In R.G. Ingalls, M.D. Rossetti, J.S. Smith, and B.A. Peters, editors, *Proceedings of the 2004 Winter Simulation Conference*, pages 1720–1727, 2004.

[Fja98] P. Fjallstrom. Algorithms for graph partitioning: A survey, 1998.

[Foo74] Norman Yeow-Khean Foo. *Homomorphic Simplification of Systems.* PhD thesis, University of Michigan, 1974.

[JFM05] Rachel T. Johnson, John W. Fowler, and Gerald T. Mackulak. A discrete event simulation model simplification technique. In *Proceedings of the 2005 Winter Simulation Conference*, pages 2172–2176, 2005.

[Kle99] Jack P. C. Kleijnen. Validation of models: Statistical techniques and data availability. In *Proceedings of the 1999 Winter Simulation Conference*, pages 647–654, 1999.

[LK00] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 2000.

[MPK06] Jurgen Martens, Ferdi Put, and Entienne Kerre. A fuzzy set theoretic approach to validate simulation models. *ACM Transactions on Modeling and Computer Simulation*, 16(4):375–398, October 2006.

[Mue05] Bengt Mueck. *Eine Methode zur benutzerstimulierten detaillierungsvarianten Berechnung von diskreten Simulationen von Materialflssen.* PhD thesis, University of Paderborn, 2005.

[PCG00] Christos Panayiotou, Christos Cassandras, and Wei-Bo Gong. Model abstraction for discrete event systems using neural networks and sensitivity information. In *Proceedings of the 2000 Winter Simulation Conference*, pages 335–341, 2000.

[Ros99] Oliver Rose. Estimation of the cycle time distribution of a wafer fab by a simple simulation model. In *Proceedings of the SMOMS 1999*, pages 133–138, 1999.

[Ros00] Oliver Rose. Why do simple wafer fab models fail in certain scenarios? In *Proceedings of the 2000 Winter Simulation Conference*, pages 1481–1490, 2000.

[Sar05] Robert G. Sargent. Verification and validation of simulation models. In *Proceedings of the 2005 Winter Simulation Conference*, pages 130–143, 2005.

[Sev90] Suleyman Sevinc. Automation of simplification in discrete event modelling and simulation. *International Journal of General Systems*, 18(2):125–142, 1990.

[SF90] Suleyman Sevinc and Norman Y. Foo. Discrete event model simplification via state classification. *AI and Simulation Theory and Application*, 22(3):211–216, 1990.

[SF98] Alex F. Sisti and Steven D. Farr. Model abstraction techniques: An intuitive overview. In *Proceedings of the SCSC 1998*, 1998.

[SY93] Lee Schruben and Enver Ycesan. Complexity of simulation models: A graph theoretic approach. In G. W. Evans, M. Mollaghasemi, E. C. Russel, and W. E. Biles, editors, *Proceedings of the 1993 Winter Simulation Conference*, 1993.

[SZ96] Hessam S. Sarjoughian and Bernhard P. Zeigler. Abstraction mechanisms in discrete-event inductive modeling. In J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 748–755, 1996.

[Wal87] Jack C. Wallace. The control and transformation metric: toward the measurement of simulation model complexity. In *WSC '87: Proceedings of the 19th conference on Winter simulation conference*, pages 597–603. ACM Press, 1987.

[ZPK00] Bernhard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation.* Academic Press, 2nd edition edition, 2000.

[Z86] Bernard P. Zeigler and Tuncer I. ren. Multifacetted, multiparadigm modelling perspectives: Tools for the 90's. In *Proceedings of the 1986 Winter Simulation Conference*, pages 708–712, 1986.

AUTHOR BIOGRAPHY

Wilhelm Dangelmaier studied Mechanical Engineering at the University of Stuttgart, Germany. In 1981, he became director and head of the Department for Corporate Planning and Control at the Fraunhofer Institute for Manufacturing. In 1991, Dr. Dangelmaier became Professor for Business Computing at the Heinz Nixdorf Institute. In 1996, he founded the Fraunhofer Center for Applied Logistics (Fraunhofer ALB). His e-mail address is dangelmaier@hni.upb.de and his Web address is http://wwwhni.upb.de/cim.

Daniel Huber studied industrial engineering at the University of Paderborn, Germany. Since 2005 he is a research assistant at the group of Prof. Dangelmaier, Business Computing, esp. CIM at the Heinz Nixdorf Institute. His main interests are material flow simulation, modeling methodology and automatic model abstraction. His e-mail address is huber@hni.upb.de.

Christoph Laroque studied business computing at the University of Paderborn, Germany. Since 2003 he has been a Ph.D. student at the graduate school of dynamic intelligent systems and, in 2007, received his Ph.D. for his work on multi-user simulation. Since then he is research assistant in the group of Prof. Dangelmaier, Business Computing, esp. CIM. He is mainly interested in material flow simulation models and the "'digital factory"'. His e-mail address is laro@hni.upb.de.

Mark Aufenanger studied business computing at the University of Paderborn. Since 2005, he is a research assistant at the group of Prof. Dangelmaier, Business Computing, esp. CIM. He is mainly interested in material flow simulation in general, simulation of logistics systems and logics in simulation models. His e-mail address is marka@hni.upb.de.