INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

ARCHITECTING MODELS OF TECHNICAL SYSTEMS FOR NON-ROUTINE SIMULATIONS

Ulf Sellgren

Abstract

Complex physical behavior and contradictory requirements, such as performance and reliability, frequently have to be addressed by engineering design in a competitive environment where speed and agility are essential. Models are very important tools in performing such complex cognitive activities. In engineering of high performance artifacts, numerical modeling and simulation (i.e., experimenting with computer-based models) are increasingly important problem solving activities. The complex nature of engineering design and the time and cost constraints on the process require highly efficient and flexible procedures to configure models of technical systems for non-routine simulations. To support efficient configuration of complex models and to enable navigation in systems models, an *architecting tool* based on the model structure matrix (MSM) has been developed. The MSM is based on the design structure matrix concept. The advantages of a modular model architecture and the usability of the developed tool are illustrated in relation to an engineering modeling challenge.

Keywords: Complexity, interface, modularity, model structure matrix, physical behavior

1 Introduction

Most manufacturing companies operate in a global business environment that is characterized by high rates of change in the market and in technology and by intense competition. Companies thus face challenges and opportunities to develop their business by providing a variety of customized products of high quality, i.e., delivering what a customer needs, when the customer needs it, and at an attractive price.

Multiple functions and multiple side effects, such as vibration, friction, wear, heat, fatigue, and crack growth, are fundamental characteristics of many technical products [1]. The expectation of high performance from a technical component or system implies that it is heavily loaded, which affects the reliability and the operational survival time. Thus complex physical behavior and contradictory requirements such as product performance and reliability frequently have to be addressed by engineering design in a competitive environment where speed and agility are essential.

Different types of models, including mental, physical, analytical, and numerical representations, are important tools in complex cognitive activities such as engineering design. Numerical modeling and simulation (i.e., experimenting with computer-based models) has become increasingly important in the development of high-performance artifacts. A computer-based numerical model is an aggregate of structured and codified knowledge [2]. Simulation is the process of extracting information from a model by performing experiments with it. The aim of a simulation, from the engineering design process viewpoint, is to produce information that improves knowledge and thus supports product-related analyses and decisions made in the design process.

Many design questions are related to the complex dependencies between shape, topological structure, and physical behavior. The purpose of a behavior model, that is, a model designed to simulate physical behavior, is to serve as a tool to find an answer to design question. A routine simulation can be defined as a simulation of a well-defined event that can be performed with a standard model (e.g. a standard FE-discretization of a single component and a known static load) and where the analysis of the results can be performed as a standard procedure. *Non-routine simulations* address problems that are more open-ended in nature and usually involve several interacting components and physical phenomena.

Model performance can be defined as how well a model fulfills its intended functions. Typical characteristics for determining performance are the accuracy, speed, and flexibility of the model. These characteristics must be judged in the context of both the purpose of the model and its intended lifecycle. Non-routine simulations, which have a tendency to be more explorative and often also more qualitative than routine simulations, are facilitated by flexible models, that is, models that are relatively easy to configure and reconfigure for a slightly different purpose.

The flexibility of a model is mainly determined by its *architecture*, which can be defined as the scheme by which the features of the model are arranged into submodels and the interface features by which the submodels interact (see figure 1). A submodel is a model of a subsystem or a component, and an interface feature is a physical relation between two mating features that are parts of different submodels. One of the most important characteristics of a model's architecture is its modularity. The modeling challenge of configuring a model that is as simple as possible and as complex as necessary for a given simulation task may be addressed with a modular or integrative model architecture. A modular submodel has mating features that are well defined and interface with only a few other submodels [3]. An integrative model has mating features that may be more complex and that are distributed across the systems model. The architecture implicitly constrains how a model can be changed. A model with a modular architecture allows changes to be made to a few isolated submodels and interface features without affecting the design of other submodels and the interactions between them. Adding, removing, or modifying features of a model with a modular architecture is, in general, a less complex task than modifying an integral model. Reduced modeling complexity also reduces the risk of introducing model errors.

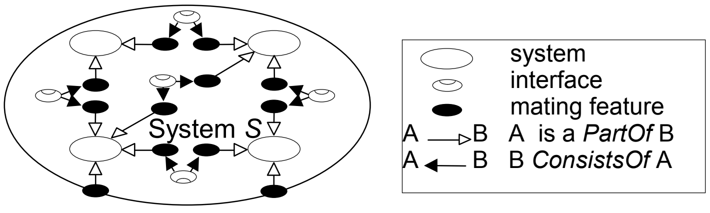
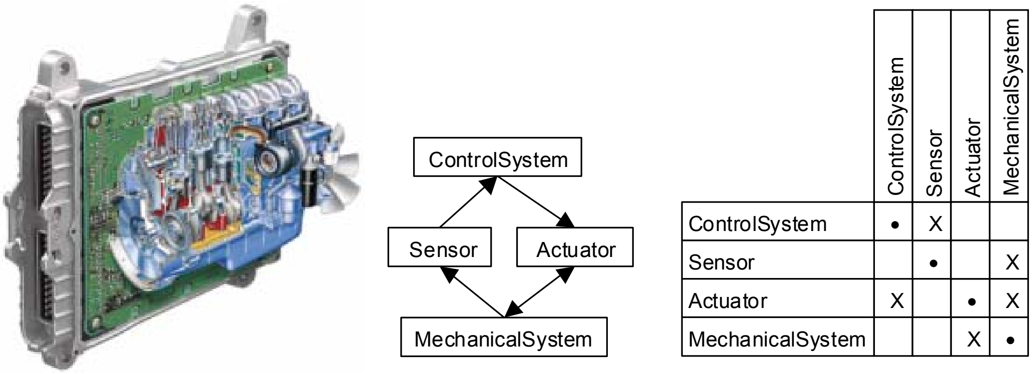


Figure 1. System as an aggregation of subsystems and interfaces (from [3])

The architecture of a system or a model of that system can be represented in several complementary ways. A virtual reality (VR) representation is an attractive option for communication and information exchange between domain-experts and non-experts. A graph-based representation (e.g. IDEF0, which is derived from SADT [4], and the Bond-graph [5]), allows the features, their characteristic properties, and their relations to be captured in a formal and logically complete way. A matrix-based representation, such as a product-based design structure matrix (DSM) [6], provides a compact, clear representation of a complex technical system and captures interactions between system elements (i.e., subsystems and modules). One of the general strengths of DSM is its applicability to large problems, for it provides a relatively good overview of the model without rapidly developing the complexity of a graph representation [7]. Figure 2 shows an artistic VR representation, a graph, and a DSM representation of an adaptive system [8]. Both the graph and the DSM clearly show a causal relation (i.e., a directed relation) between the sensor and the control system and a non-causal relation between the actuator and the mechanical system.



	ControlSystem	Sensor	Actuator	MechanicalSystem
ControlSystem	•	X		
Sensor		•		X
Actuator	X		•	X
MechanicalSystem			X	•

Figure 2. VR (from [9]) graph-based and DSM representations of an adaptive system

To support efficient configuration of complex models and to enable navigation in systems models, an *architecting tool*, referred to as the model structure matrix (MSM), has been developed. The MSM, which can be viewed as a model-based DSM, provides a compact representation of a complex model and its building blocks (i.e. submodels and interface features). The information model behind the MSM and a modular modeling method are conceptually described below. An engineering modeling challenge is used to illustrate the advantages given by the modular model architecture and the usability of the developed architecting tool.

2 Architecting models of systems

2.1 An information model

Linking design and behavior models is fundamentally different from typical data exchange tasks in that it requires heterogeneous transformations, that is, transformations of one or more types of information into a different type of information [10]. The integration challenge is further complicated by the fact that the apparent shape of a behavior model is usually idealized compared to the design model, while the required mating relations are frequently more detailed. Not only that, but the implementation of a mating relation varies between different disciplines and simulation tools. The modeling complexity is further increased in integrated product development where many variants and revisions of the design model are frequently floating around concurrently. This complexity can be addressed by using an information model with a three-layered architecture, that is, a design layer, a generic behavior layer, and an application layer [3]. The *design layer* presents design models, which represent design features such as shape, material, and orientation in space. The interactions between design submodels take place at interfaces, where an interface is a pair of mating faces. Interfaces can be classified as attachment, constraint, or contact (see figure 3). A behavior model, which is treated in the *generic behavior layer*, is designed to represent a specific behavior of a design model. Mating features are representations of mating faces. An interface feature is a representation of an interface at a generic behavior level. For example, it may represent the contact between two mating features that may be discrete representations of two mating faces. A contact can, for example, be represented by an attachment interface feature and vice versa. In the *application layer* we have source files that are self-contained models in proprietary format. The actual connection of two submodels is defined with an application- and implementation-dependent connect-feature, such as a set of FE software-specific contact elements. Node numbers, property ranges and the like for an FE based submodel or connect feature are stored as source file metadata. Architectural information stored in the generic behavior layer can be used to aggregate a system model in the application layer from application submodels and connect features by using number offsetting. Figure 4 shows the interface objects as building blocks of a more complete information model.

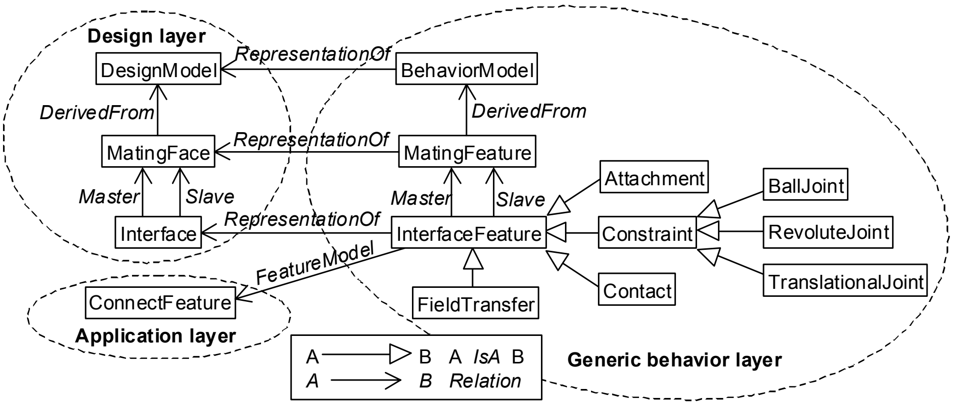


Figure 3. Interface modeling as a three-layered architecture – design layer, behavior layer and application layer

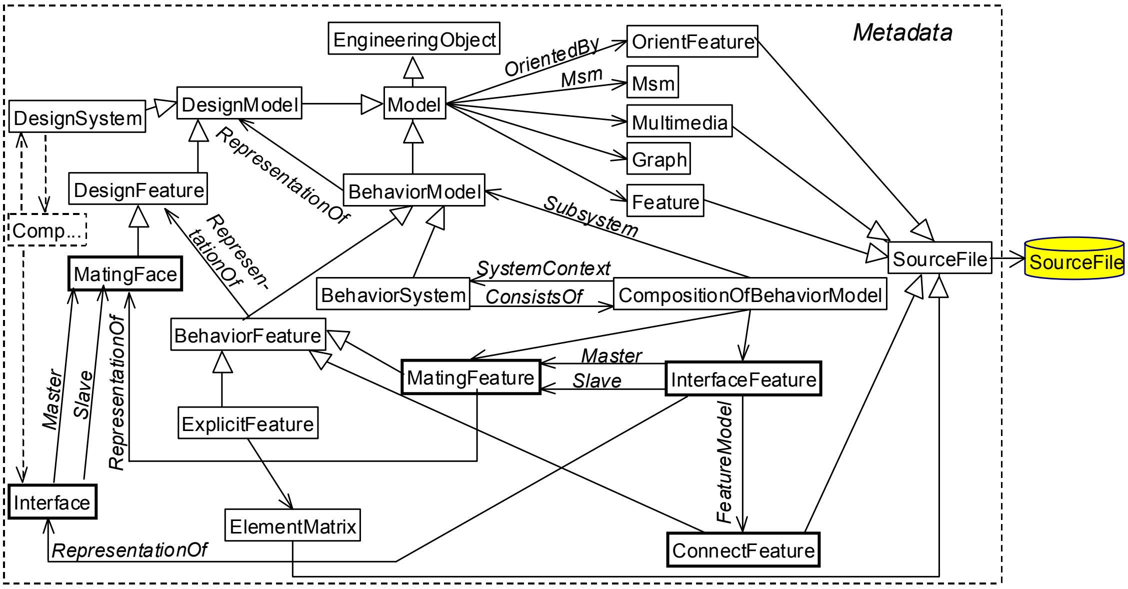


Figure 4. Interface model objects put into the context of a larger information model

2.2 A modular modeling method

A framework for product modeling, based on commercial CAD, CAE, and PDM-technology and the information model conceptually described above was proposed in [3]. A modular method for behavior modeling of complex technical systems was elaborated on in [11]. Modeling, according to this method (see figure 5), starts with an analysis of the technical specification with its different constraints. Then the system is divided in components and the mating faces of the components are defined. After that, a system model for a particular configuration is generated (structuring). The pairs of mating faces and their connections are then identified. Finally the system model is divided into convenient modules with mating faces (modularization), which gives a modularized behavior model for reuse, modification and further development. This method can be seen as a general modularization method that is useful for both design and behavior modeling.

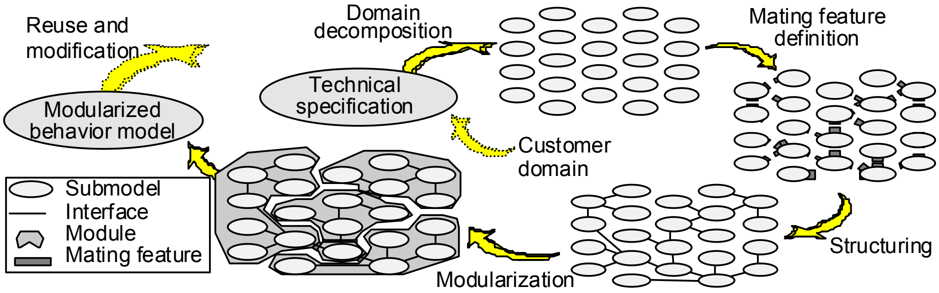
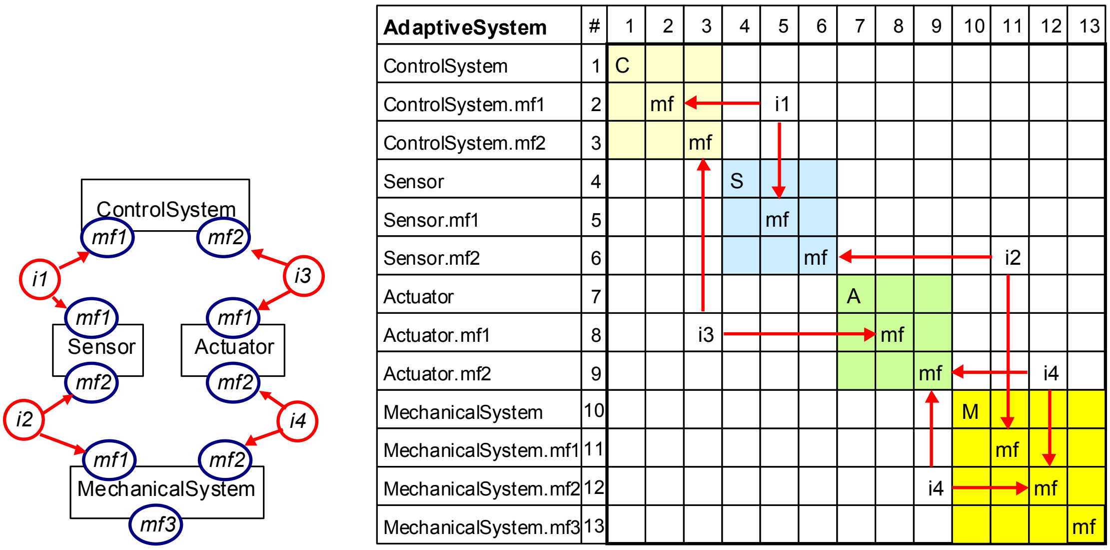


Figure 5. A modularized approach to behavior modeling [11]

2.3 The MSM architecting tool

DSM is a very general technique that has many applications. The three most common choices of variables in a DSM are components [6], parameters [12], and tasks [13]. A component-based or product DSM documents interactions between elements in a complex system architecture. Because it enables analyses based on clustering techniques, the product DSM has come to be used as an architecting tool in engineering design (e.g. Blackenfelt [14]). A newly developed model DSM research prototype, referred to as the model structure matrix (MSM), is described below.



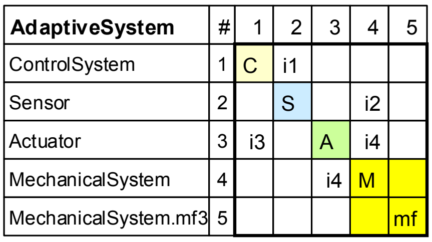
AdaptiveSystem	#	1	2	3	4	5	6	7	8	9	10	11	12	13
ControlSystem	1	C												
ControlSystem.mf1	2		mf			i1								
ControlSystem.mf2	3			mf										
Sensor	4				S									
Sensor.mf1	5					mf								
Sensor.mf2	6						mf					i2		
Actuator	7							A						
Actuator.mf1	8			i3					mf					
Actuator.mf2	9									mf			i4	
MechanicalSystem	10										M			
MechanicalSystem.mf1	11											mf		
MechanicalSystem.mf2	12									i4			mf	
MechanicalSystem.mf3	13													mf

Figure 6. A graph- (left) and an MSM-representation (right) of the architecture of a simple adaptive system

The left portion of figure 6 shows an expanded graph of the adaptive system from figure 2. Each subsystem has mating features (e.g. *mf1*) that are referenced by four interface features labeled *i1* to *i4*. The causality is an internal property of each interface feature that is not explicity shown in the graph. On the right side of figure 6, the system is represented by an MSM. The name of the current systems model is *AdaptiveSystem*. The four submodels, *ControlSystem*, *Sensor*, *Actuator* and *MechanicalSystem*, and the nine mating features are labeled C, S, A, M, and mf, respectively in the diagonal of the MSM. The four interface features are off-diagonal terms in the matrix. Non-causality is easily observed as a symmetric relationship, as in the two symmetric instances of i4, in the MSM. The elements of the MSM are pointers to submodels, mating features, and interface features, that is, to instances of classes in the information model. The relations such as *RepresentationOf* defined in the information model can then be used to find related objects. The MSM is thus a *model navigation tool*.

The MSM can also be integrated with a tool to change the model state. An MSM that shows only submodels and mating features (i.e. only diagonal elements) represents a model that is in a *collected* state. Interface features can be created by clicking empty off-diagonal MSM elements that represent relating pairs of mating features (e.g. the row 2 and column 5 element for i1 in figure 6). When all interface features have been created, the system model is *mated*. A model is *connected* when connect features have been generated for all interface features. The MSM is thus not merely a navigation tool but also a *model architecting tool*.

A mating feature that is referenced by an interface feature in an MSM can be viewed as an internal property of the parent submodel from which the mating feature derives. Hiding the internal properties gives a condensed and less complex MSM. Figure 7 shows a condensed MSM of the *AdaptiveSystem*. The MSM shows that the third mating feature of the *MechanicalSystem* is an unmated and it is thus an external mating feature that can be used to connect the *AdaptiveSystem* to another system (i.e. to configure a higher order system) or to an environment feature.



AdaptiveSystem	#	1	2	3	4	5
ControlSystem	1	C	i1			
Sensor	2		S		i2	
Actuator	3	i3		A	i4	
MechanicalSystem	4			i4	M	
MechanicalSystem.mf3	5					mf

Figure 7. A condensed MSM with one external mating feature

3 A modeling challenge

Volvo Construction Equipment (Volvo CE) is one of the world's leading manufacturers of construction machines, with a product range encompassing wheel loaders (see figure 8), excavators, articulated haulers, motor graders, and more.

Figure 8. A high performance Volvo wheel loader

The new range of Volvo wheel loaders utilizes modern technology such as load-sensing hydraulics, service accessibility, TP linkage, care cab, automatic power shift, and high performance and low emission engines to combine increased performance with improved operator comfort and reduced environmental impact [15]. A wheel loader is heavily loaded during normal operation, which gives strength and fatigue predications high priority. A complication is that different customers operate under significantly different conditions and that large differences can be observed in the behavior of the individual operators. Engineering predictions of product performance, reliability, and quality must thus be based on many different considerations and also on contradictory requirements.

The mechanical CAD software SolidEdge from EDS and several CAE tools, including the finite element (FE) software Ansys from Ansys Inc. and the multi-body systems software Adams from MSCSoftware, are currently used at Volvo CE. Ansys is mainly used for strength and fatigue analyses and Adams is used for dynamic and comfort simulations. Systematic and novel methods are used for strength and fatigue assessment of individual components. Idealized FE model geometries are created with the CAD software and transferred as Parasolids files to the CAE domain for further processing. Figure 9 shows two Ansys models of wheel loader frames. The size of an Ansys model of a symmetric half of a single frame component, with the level of detail required for fatigue analysis, is slightly less than one million degrees of freedom (DOFs). The size of an integrated rear frame and front frame model would thus be in the order of three to four million DOFs, which is an unrealistically large model even for static simulations.

Figure 9. Rear fame (left) and front frame (right) FE models

In the context of the product realization process, a major modeling and simulation challenge identified at Volvo CE has been to enable event simulations of complete vehicle dynamics for analyses of product behavior and performance. A working hypothesis is that this challenge can be addressed efficiently by combining a modularized model architecture, tools and modeling methods for flexible configuration of system models with an information model that extends current PDM technology and enables it to integrate the CAE and CAD domains.

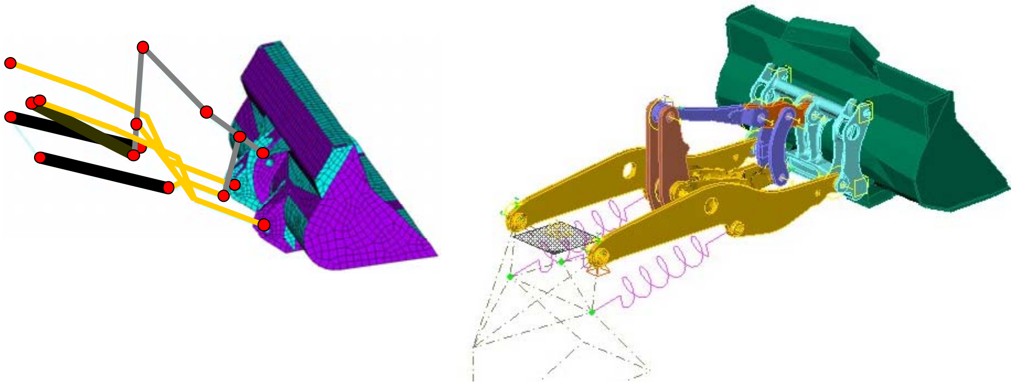
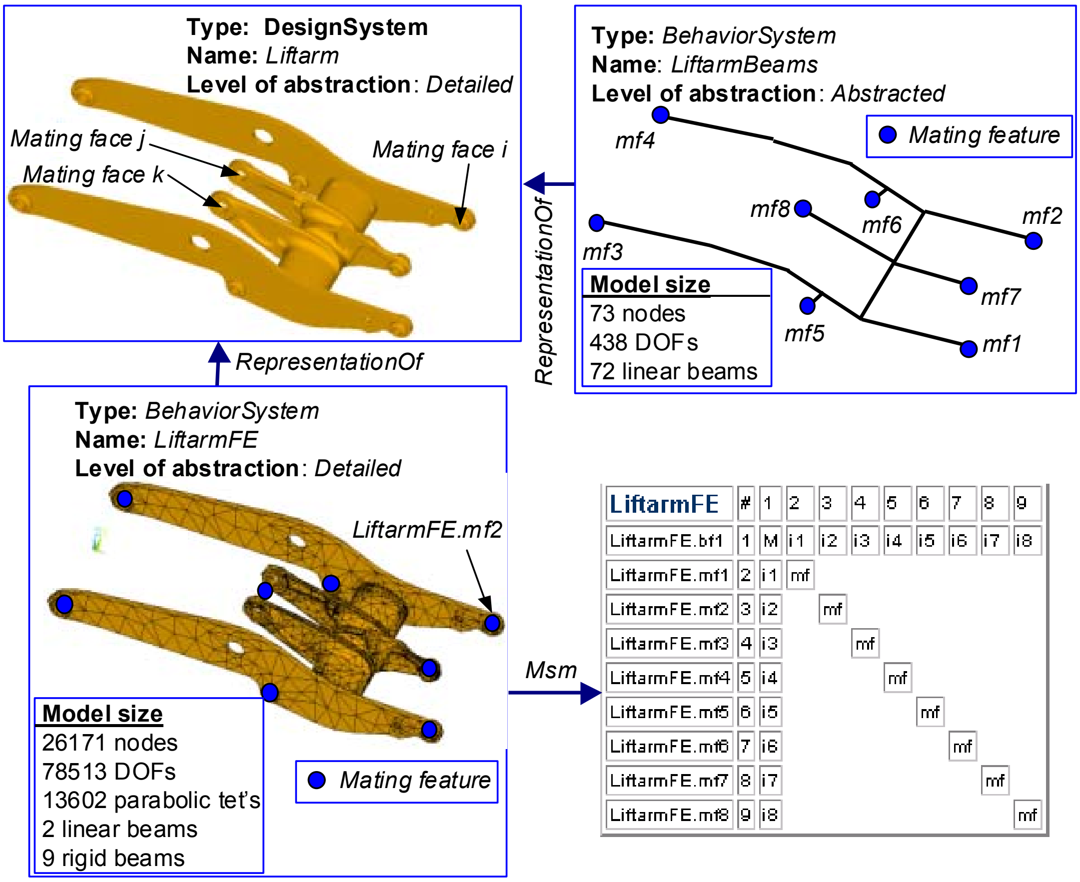


Figure 10. Two FE models of a lifting unit, Ansys (left) and Fedem (right)

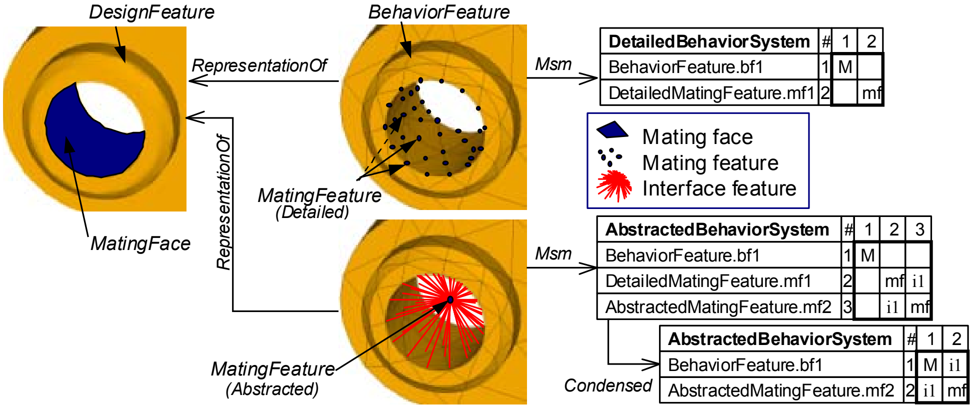
A set of design features and FE-based behavior features at different levels of abstraction have been created with the aim of enabling explicit experiments on how to successfully address the identified challenge. The FE features were created by decomposing two existing FE models in different formats (see figure 10) of a lifting unit. The lifting unit is a complex subsystem mounted on the wheel loader front frame. The Ansys model is an integrated 64000 DOFs FE model with the main links and the hydraulic components modeled as beams and the bucket and bucket link modeled with shell elements. The Fedem [16] model in figure 10 integrates an FE assembly with a control system. The simulation technology used in Fedem is based on component mode synthesis (CMS) as proposed by Craigh and Bampton [17]. Building blocks in the mechanical domain are FE component models in the Nastran format, which is widely used in the automotive and aerospace industries. Before reduction to component modes, the total number of DOFs in the assembled Fedem model is 256000. The Nastran models have all been translated to Ansys submodels, meaning that each behavior submodel in the generic behavior layer is referencing two feature models in the application layer. So far, only the Ansys models have been used in the configuration experiments.



LiftarmFE	#	1	2	3	4	5	6	7	8	9
LiftarmFE.bf1	1	M	i1	i2	i3	i4	i5	i6	i7	i8
LiftarmFE.mf1	2	i1	mf							
LiftarmFE.mf2	3	i2		mf						
LiftarmFE.mf3	4	i3			mf					
LiftarmFE.mf4	5	i4				mf				
LiftarmFE.mf5	6	i5					mf			
LiftarmFE.mf6	7	i6						mf		
LiftarmFE.mf7	8	i7							mf	
LiftarmFE.mf8	9	i8								mf

Figure 11. A design model and two FE-based behavior representations of a lift arm component of a lifting unit

The model variants of the lifting unit lift arm component and their relations are shown in figure 11. We can see two behavior model variants that are at different levels of abstraction. One model is a detailed discretization of a solid geometry into parabolic tetrahedrons. The other variant is a discretization into linear beams of an abstracted or idealized geometry - in this case a geometry with a reduced dimensionality. Both representations show eight mating features. Each of the eight mating features in the abstracted model references one single nodal point with three translational and three rotational DOFs in the behavior feature. The detailed FE model of the liftarm also has eight mating features, but each mating feature comes in two variants. A detailed mating feature variant references a set of nodes, each with three translational DOFs, that are located on the mating face of the design model (see figure 12). A detailed mating feature can be referenced by a contact type of interface feature and used in a detailed contact simulation. An abstracted mating feature references a nodal point with six DOFs located at the center of the mating face. In the application domain, this mating feature variant is realized with a set of beam elements or multi-point constraint (MPC) equations that relate the DOFs of the mating feature to the DOFs of the detailed mating feature. The abstracted mating feature is thus mated with the detailed mating feature with an internal interface feature, which is shown by the MSM in figure 12.



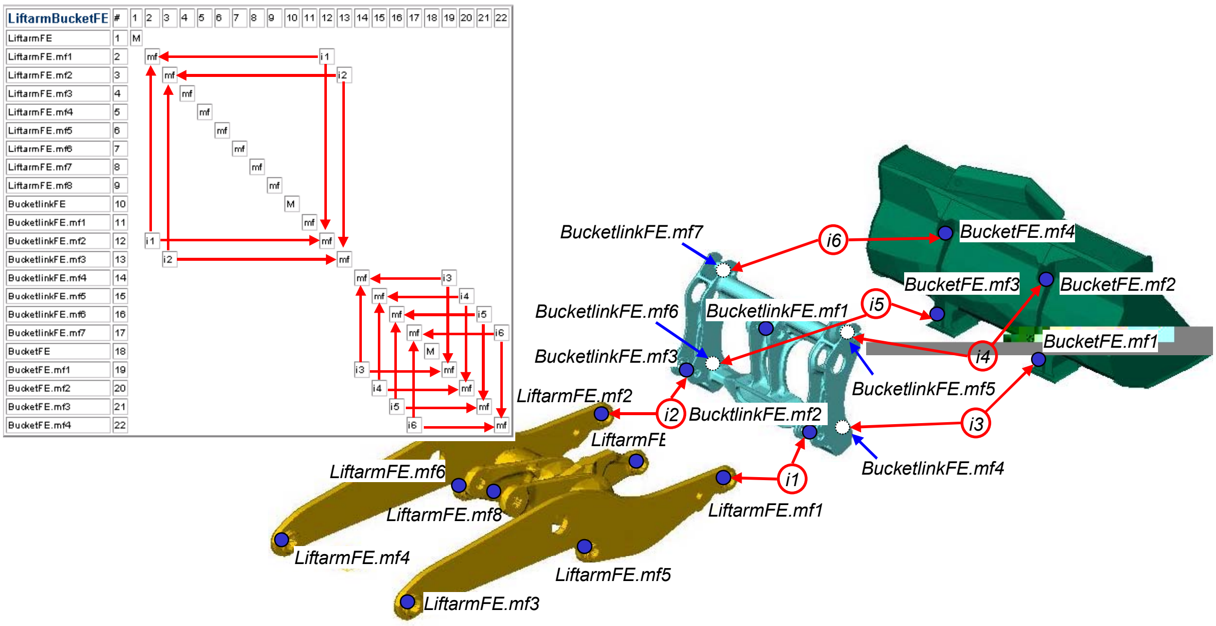
DetailedBehaviorSystem	#	1	2
BehaviorFeature.bf1	1	M	
DetailedMatingFeature.mf1	2		mf

AbstractedBehaviorSystem	#	1	2	3
BehaviorFeature.bf1	1	M		
DetailedMatingFeature.mf1	2		mf	i1
AbstractedMatingFeature.mf2	3		i1	mf

AbstractedBehaviorSystem	#	1	2
BehaviorFeature.bf1	1	M	i1
AbstractedMatingFeature.mf2	2	i1	mf

Figure 12. A design model and an FE-based behavior representation showing two mating feature variants

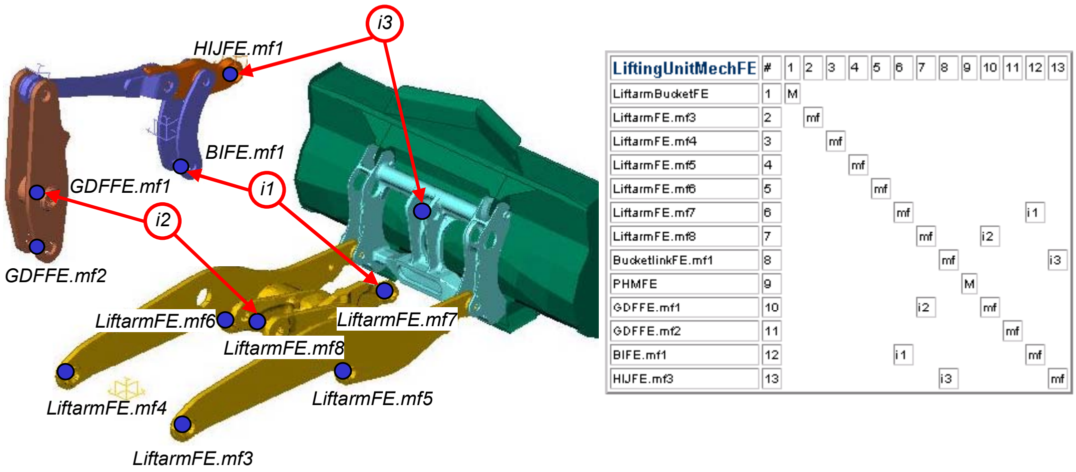
Figure 13 shows a collected behavior system, with the name *LiftarmBucketFE*, that is an aggregation of a lift arm, a bucket link, and a bucket submodel. The behavior system is a representation of a design system, which holds references to the orientation of the design subsystems and thus to its representations. By defining six revolute joint interface features in the MSM, the three submodels can be mated.



LiftarmBucketFE	#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LiftarmFE	1	M																					
LiftarmFE.mf1	2		mf										i1										
LiftarmFE.mf2	3			mf										i2									
LiftarmFE.mf3	4				mf																		
LiftarmFE.mf4	5					mf																	
LiftarmFE.mf5	6						mf																
LiftarmFE.mf6	7							mf															
LiftarmFE.mf7	8								mf														
LiftarmFE.mf8	9									mf													
BucketlinkFE	10										M												
BucketlinkFE.mf1	11											mf											
BucketlinkFE.mf2	12		i1										mf										
BucketlinkFE.mf3	13			i2										mf									
BucketlinkFE.mf4	14														mf					i3			
BucketlinkFE.mf5	15															mf					i4		
BucketlinkFE.mf6	16																mf					i5	
BucketlinkFE.mf7	17																	mf					i6
BucketFE	18																		M				
BucketFE.mf1	19														i3					mf			
BucketFE.mf2	20															i4					mf		
BucketFE.mf3	21																i5					mf	
BucketFE.mf4	22																	i6					mf

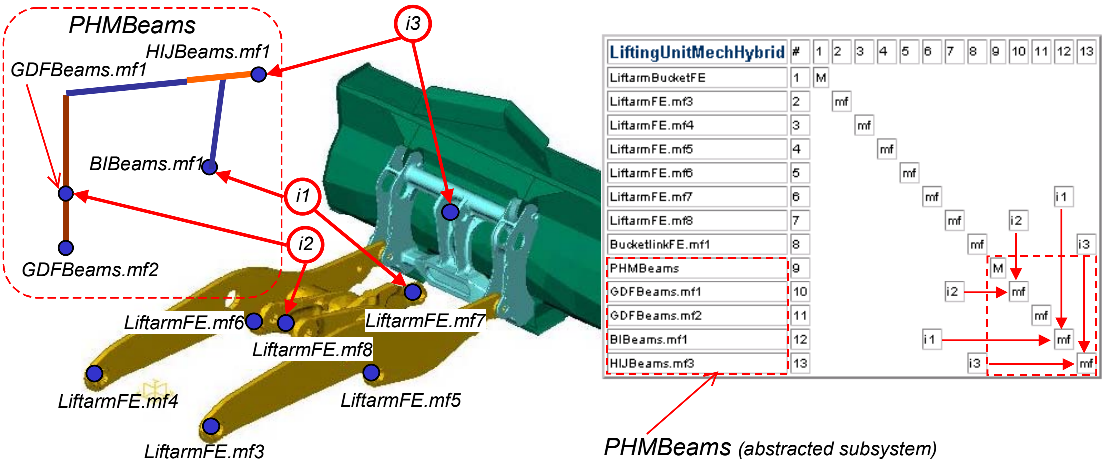
Figure 13. Configuration of a behavior model of the mechanical subsystem of the lifting unit

The behavior subsystem of the parallell holding mechanism, *PHMFE*, is configured in a similar way and aggregated with the *LiftarmBucketFE* subsystem to form a higher order model configuration. This model, referred to as *LiftingUnitMechFE* in figure 14, is a behavior representation of the complete mechanical subsystem of the lifting unit. In figure 15, the *PHMFE* subsystem is replaced with an abstracted variant referred to as *PHMBeams* and remated with the *LiftarmBucketFE* subsystem (i.e., the three interface features are redefined). This simple reconfiguration reduced the size of the simulation model from 256000 DOFs to 165000 DOFs. Yet with three detailed submodels, the model may still be considered unfocused.



LiftingUnitMechFE	#	1	2	3	4	5	6	7	8	9	10	11	12	13
LiftarmBucketFE	1	M												
LiftarmFE.mf3	2		mf											
LiftarmFE.mf4	3			mf										
LiftarmFE.mf5	4				mf									
LiftarmFE.mf6	5					mf								
LiftarmFE.mf7	6						mf						i1	
LiftarmFE.mf8	7							mf			i2			
BucketlinkFE.mf1	8								mf					i3
PHMFE	9									M				
GDFFE.mf1	10							i2			mf			
GDFFE.mf2	11											mf		
BIFE.mf1	12						i1						mf	
HIJFE.mf3	13								i3					mf

Figure 14. A detailed behavior model of the mechanical subsystem of the lifting unit



LiftingUnitMechHybrid	#	1	2	3	4	5	6	7	8	9	10	11	12	13
LiftarmBucketFE	1	M												
LiftarmFE.mf3	2		mf											
LiftarmFE.mf4	3			mf										
LiftarmFE.mf5	4				mf									
LiftarmFE.mf6	5					mf								
LiftarmFE.mf7	6						mf						i1	
LiftarmFE.mf8	7							mf			i2			
BucketlinkFE.mf1	8								mf					i3
PHMBeams	9									M				
GDFBeams.mf1	10							i2			mf			
GDFBeams.mf2	11											mf		
BIBeams.mf1	12						i1						mf	
HIJBeams.mf3	13								i3					mf

Figure 15. An hybrid behavior model of the mechanical subsystem of the lifting unit

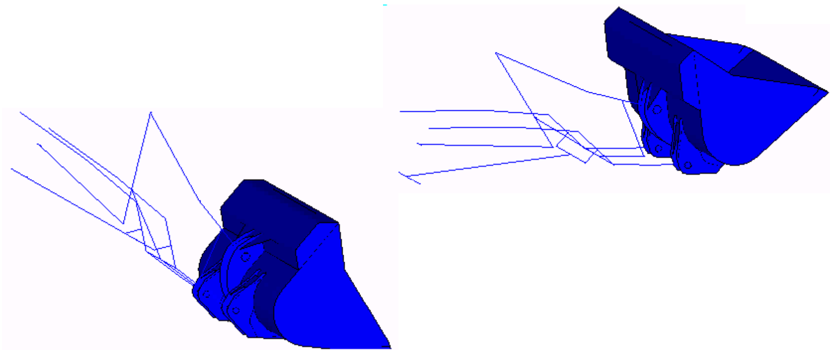


Figure 16. Event simulation with an abstracted behavior model of the lifting unit

If we replace all detailed submodels in the behavior model with their abstracted variants, we end up with the initial 64000 DOFs Ansys model shown in the left portion of figure 10. A further replacement of the bucket link and bucket submodels with an explicit model (i.e. a superelement) means that we end up with a model as small as 556 DOFs. This light-weight model was connected to a behavior model of an actuator subsystem (two lift cylinders and a tilt cylinder) and an abstracted front frame submodel. Figure 16 shows two snapshots from an FE event simulation performed with this model. If we exchange the abstracted front frame submodel for a detailed representation, we have a system model that is well suited to a simulation task with a detailed focus on the behavior of the front frame, as in fatigue analysis.

The model configuration examples presented above demonstrate how a modular model architecture enables configuration of new system models from existing submodels and interface features. Modeling experiments indicate that the ability to synthesize a large and complex model from small and self-contained submodels and related interface features means that most practical system models can be created in a time linear to the number of submodels and interface features. The modeling time is thus independent of the size of the submodels.

The MSM has proved to be an efficient and compact representation of a complex model. Experiments have shown that the MSM can be used as an efficient user interface for an integrated model navigation and configuration tool.

4 Conclusions and discussion

Engineering design is a complex cognitive process of solving open-ended problems. Modeling and experimenting with computer-based models enables engineers to deal with and understand complex relations between physical objects and between interrelated physical phenomena. The open-ended character of engineering problems and the time and cost constraints on the design process require highly efficient and flexible procedures to configure behavior models of technical systems for non-routine simulations. This challenge is preferrably addressed with a *modular model architecture* that enables submodels to be reused as building blocks of new system configurations.

Modeling and simulation reduce engineering complexity but add two significant types of complexity to the process. The internal modeling complexity is a measure of the multitude of relations between all the submodels that make up a system model. The external modeling complexity is a measure of the multitude of relations between different types of model representations and variants. Modeling experiments clearly show that an *information model* with a layered architecture - design layer, generic behavior layer, and application layer – enables the external model management challenge to be addressed successfully. A matrix-based tool, the *model structure matrix* (MSM), has been developed as a user interface to an integrated model navigation and architecting tool. The MSM is thus a tool that addresses the challenge from the internal model complexity and enables integrity assessment of complex models.

Modeling studies indicate that by combining the modular model architecture presented here, the layered information model, and the MSM-based architecting tool most practical system models to be created in a time linear to the number of submodels and interface features. The fact that the time to configure a system model is almost independent of the size of the submodels indicates that the presented modeling approach reduces modeling complexity.

For performance reasons, each model should be developed for a specific purpose. Reuse of a model, which most likely was developed by another engineer, as a submodel in a new context, requires that the appropriateness of the model can be assessed. A future research challenge is to define explicit and implicit characteristic parameters that can be used to determine the appropriateness of a stored submodel or interface feature for an intended simulation task and to extend the presented information model with these parameters.

5 Acknowledgements

The Swedish Engineering Design Research Agenda (ENDREA) and the Swedish Agency for Innovation Systems (VINNOVA) financially supported the work presented here. The active support of Stefan Forsberg, Volvo Wheel Loaders AB, is gratefully acknowledged.

References

[1] Whitney, D.E., 1996, "Why mechanical design cannot be like VLSI design", Research in Engineering Design, Vol. 8, 1996, pp.125-128.

[2] Boisot, M.H., "Is your firm a creative destroyer? Competitive learning and knowledge flows in the technological strategies of firms", Research Policy, Vol. 24, 1995, pp.489-506.

[3] Sellgren, U., "Simulation-driven design: motives, means, and opportunities", Doctoral thesis, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden, 1999.

[4] Ross, D.T., " "Structured analysis (SA): A language for comunicating ideas", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, 1977, pp.16-34.

[5] Breedveld, P.C., "Physical systems theory in therms of bond graphs", Ph.D. thesis, University of Twente, Enschede, Netherlands, 1984.

[6] Pimmler, T.U. and Eppinger, S.D., "Integration analysis of product decompositions", ASME Design Theory and Methodology Conference, DE Vol. 68, 1994.

[7] Malmström, J, Pikosz, P. and Malmqvist, J., "The complementary roles of IDEF0 and DSM for the modeling of information management processes", Advances in Concurrent Engineering – CE'98, Tokyo, Japan, 1998.

[8] Grohmann, B., Konstanzer, P., Lay, A., Wallmersperger, T., Kröplin, B., "On the principle of adaptive structure systems", NAFEMS Seminar: Finite Element Application for Adaptive Structural Systems, Nov 11-12, Ref. R0068, 1998, NAFEMS, UK.

[9] http://www.Scania.se, 2000.

[10] Peak, R.S., Scholand, A., et al., "On the routinization of analysis for physical design", ASME International Engineering Congress and Exposition, 1996, pp.73-82.

[11] Sellgren, U. and Drogou, R., "Behavior modeling in mechanical engineering: A modular approach", Engineering with Computers, Vol. 14, 1985, pp.185-196.

[12] Steward, DV, "The design structure system: A method for managing the design of complex systems", IEEE Transactions on Engineering Management, Vol. EM-28, No. 3,1981, pp.71-74.

[13] Smith, RP and Eppinger, SD, "Identifying controlling features of engineering design iteration", Management Science, Vol. 43, No. 3, 1997, pp.276-293.

[14] Blackenfelt M., "Managing complexity by product modularisation", Doctoral thesis, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden, 2001.

[15] http://www.volvoce.com, 2003.

[16] http://www.fedem.com, 2003.

[17] Craig, R.R. and Bampton, M.C.C., "Coupling of substructures for dynamic analysis", AAIA Journal, Vol. 6, No. 7, 1968, pp.1313-1319.

Corresponding author:
Ulf Sellgren
Department of Machine Design, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden
Tel: + 46 8 790 73 87, E-mail: ulfs@md.kth.se, URL: http://www.md.kth.se/people/mme/ulfs.shtml