Proceedings of the 19th CIRP Design Conference – Competitive Design, Cranfield University, 30-31 March 2009, pp412

On the Potential of Function-Behavior-State (FBS) Methodology for the Integration of Modeling Tools

A. A. Alvarez Cabrera, M. S. Erden, T. Tomiyama
Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands
{a.a.alvarezcabrera, m.s.erden, t.tomiyama}@tudelft.nl

Abstract
Current mechatronic products tend to be very complex systems. A design team is necessary to develop such products, and appropriate modeling and design support tools are essential to aid the design team. The Automatic Generation of Control Software for Mechatronic Systems project aims to develop a set of prototype tools and a framework to integrate available modeling tools, aiming to support the generation of control software for mechatronic machines. The project contemplates functional modeling as part of this framework. This paper considers the Function-Behavior-State (FBS) model as a base for the functional model, and discusses its potential regarding integration of modeling tools.

Keywords:
Function modeling, function behavior state, model integration, mechatronic systems design

1 INTRODUCTION

Development of mechatronic products brings new challenges for design because modern mechatronic systems tend to be complex by nature. The design of such systems requires the participation of experts from several domains that cooperate to solve problems from the point of view of their specialties. Appropriate modeling and design support tools are essential to deal with system complexity, and one alternative for support is to accomplish modeling tool integration.

The project of Automatic Generation of Control Software for Mechatronic Systems aims to develop a set of prototype tools and a framework (see Figure 1) to allow seamless integration among available modeling tools, so that an interdisciplinary product development team can (almost) automatically generate control software for mechatronic machines. The project considers functional modeling and reasoning from model information (i.e., qualitative reasoning [1]) as mechanisms to reach the goal of model integration (encircled in clear dash-dot lines in the figure), and to endow the set of models with the necessary information to generate control software. Implementing these aspects seeks to cope with complexity by providing a base for a complete system model in the most abstract levels, where attaining common understanding is more practical.

Use of functional models can be advantageous for several reasons. First, they provide a way of representing the intention of the designers of the system, both for design and for use. Secondly, but not less important, functions can represent a system at several levels of detail, which allows to change the level of abstraction in which the model is seen while preserving, what we could call, the consistency of the model (i.e., the model can still represent the whole system while showing more detail where required). Additionally, functions can model indistinctively hardware, software, and systems from different domains. In a sense, functional models get very close to represent the architecture of a system. The importance of modeling functions for machine and process design was already recognized in works of Rodenacker [2] and Pahl and Beitz [3]. There, design is seen as a process of transformation and mapping of information from abstract concepts (i.e., functions and requirements) to concrete descriptions of physical systems, that later will allow manufacturing a system. Thus, design cannot be done without the existence of these abstract concepts that specify what the system is expected to do. Careful documentation and modeling of the functional description is then as necessary as it is for any other information related to the design.

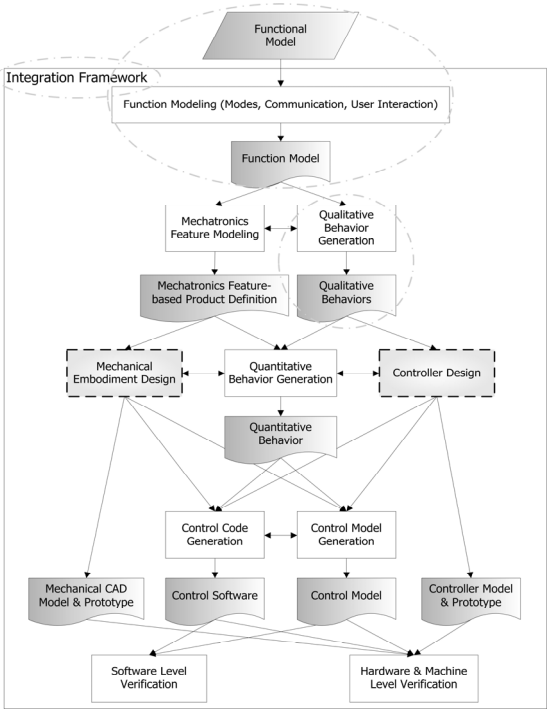


Figure 1: Architecture of the proposed control software generation framework. Black dash-lined blocks correspond to existing, commercial modeling tools [4].

This paper considers the Function-Behavior-State (FBS) model [5] as a base for the functional model description in the proposed framework, and discusses the potential of such model regarding integration of modeling tools.

The FBS model was designed to be part of an integrated framework but it was not intended to be the backbone for the integration activity, and thus, some adaptation is necessary. Some advantages that lead to the choice of FBS are that it:

- Clearly separates design intention and objective relations between components.
- Is built to support qualitative reasoning activities.
- Has been already implemented in a software tool and tested to some extent (cf. FBS modeler in [6]).

Another important reason to support the choice of FBS is that FBS differs from most system models developed at an early stage of design which are not aimed to prescribe how the systems actually behave [7]. Instead, FBS also pretends to simulate the behavior of the system from an objective point of view.

Section 2 exposes some basic concepts regarding model integration. Sections 3 and 4 recapitulate the literature about the FBS modeler and other tools that appear in its implementations. The discussion about the potential applications of FBS for model integration and proposals to do this can be found under section 5. Section 6 presents the general integration approach using FBS. Finally, section 7 describes the current progress of this research with the help of a practical example and mentions the next steps to work at. Section 8 presents the conclusions.

2 MODEL INTEGRATION REQUIRMENTS

An integrated modeling paradigm that gives the designers a proper view of the system as a whole in several levels of abstraction, and that keeps track of the current state of design is fundamental to attain an integrated design that can cope with the problems brought by complexity [4].

To establish some common grounds for the integration of models, literature proposes some basic requirements:

1. It is necessary to separate the modeler from the solver in order to deal with the definitional integration (i.e. of the models) and the procedural integration (i.e. integration of the solvers) processes separately [8].
2. Definitional integration becomes possible as models can be represented in a common language. A conversion of external models to a common language is necessary [8].
3. Procedural integration may be more suitable for situations where the models and their associated solvers are of diverse nature [8].
4. It is necessary to detect correspondence of variables between models. This seeks to minimize necessary human intervention in the detailed levels of the model integration process. Typing schemes offer an alternative to aid in this process [8].
5. Graphical user interfaces and views are crucial to provide model integration support [8].
6. One shared database that contains *all* the data of the integrated models quickly becomes a bottleneck [9].
7. Modularity, from the point of view of reusability, and the use of model libraries helps to speed up the modeling and verification processes [10].

3 FBS FUNCTION MODELING

FBS is a function modeling scheme created to support conceptual design in computer aided design (CAD) systems [5]. FBS aims to build a functional concept ontology [11]. Most components of the FBS model are based on a process ontology known as Qualitative Process Theory (QPT) [12].

As specified in [11], process ontologies focus on the effects of processes over the attributes of entities, and functional concept ontologies look to develop models of devices from the subjective perspective of humans.

An FBS model (see Figure 2) can be divided in three parts: (1) the functions layer, (2) the behaviors layer, and (3) the states layer. Each layer is connected to the next one to form a framework that describes the functionality of a system and how to attain such functionality. Behavior and state representations are based on QPT. All the objects are stored in a knowledge base, which is briefly described in section 4.2. The next part of this section contains a brief description of the concepts and main ideas of FBS [5], [13]-[15].

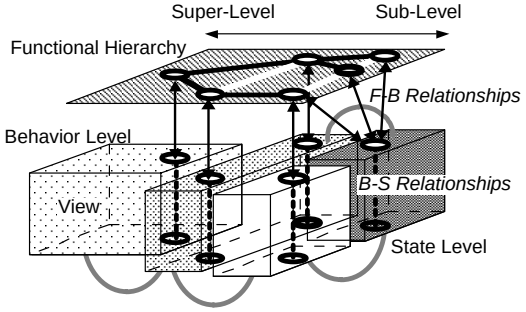


Figure 2: Scheme of FBS model [14]

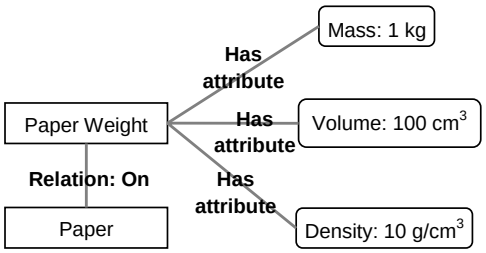


Figure 3: State of paper weight (adapted from [14])

3.1 State

To define state, first the concept of entity must be introduced. An entity corresponds to an object like a solid, a gear, or a single tooth of a gear. The choice for an entity depends on the level of detail being modeled. Entities possess attributes that describe them. Lastly, entities are connected to other entities by relations.

For modeling proposes, in FBS states and entities are treated simultaneously. A state is defined as "*a set of attributes and relations between entities*", and thus a state cannot be described without the use of entities. Figure 3 depicts a state, showing several attributes of the entity "Paper Weight" and how it relates to the entity "Paper".

3.2 Behavior

First it is necessary to define physical phenomena in order to ease the explanation of behavior in FBS. Physical phenomena link a group of entities and their relations to physical laws (e.g., first law of Newton) that regulate the changes of attributes and states. These changes are called state transitions. An example of a physical phenomenon is "linear motion", which connects an entity (e.g., a solid body) and its attributes to a law (e.g., $F = m\,a$). Physical phenomena are knowledge elements that contain the Behavior-State (B-S) connections among the classes of the objects. Physical phenomena become active or inactive according to a set of enabling conditions specified by the presence of a set of entities, attributes, and relations.

Behaviors constitute objective representations of what a system does. A behavior is defined in FBS as "*a sequence of state transitions over time*".

To model behavior it is possible to directly instantiate physical phenomena or groups of them. These instantiations are called physical features. Causality between involved physical phenomena can also be specified inside a physical feature. Another modeling option is to specify a behavior as a state transition table. Then an additional tool (described in section 4.1) searches and proposes candidate physical features that are able to obtain such state transitions.

3.3 Function

The definition of function tends to vary in the field of functional modeling, but many authors agree that the function is subjective in nature and carries the intention of design or use [11], [16]. In FBS, function is defined as "*a description of behavior abstracted by human through recognition of the behavior in order to utilize the behavior.*" Since the function is abstracted from the behavior, the function alone is not meaningful for representing the system. Therefore, in FBS a function is represented by a tuple of function symbol and behavior that can realize the function. Function-Behavior (F-B) relations are established when a function is connected to a physical feature.

The function symbol is a text that describes the function in the form of "to do something." No further restrictions or guidelines are necessary to describe the function at this level because the function symbol itself is just intended for human recognition.

Functions form a hierarchical structure that results from the decomposition of general functions into more specific subfunctions, forming a function tree [17]. Decomposition of functions is classified as either causal decomposition (i.e., into subfunctions whose execution is causally related) or task decomposition (i.e., the subfunctions can be executed independently from each other).

When several functions and F-B relations have been placed in the model, the designer can proceed to connect the entities of different physical features that represent the same object. This is referred as unification of entities.

4 EXISTING DEVELOPMENTS RELATED TO FBS

The FBS modeling scheme proposes a framework to model functions. Even though these models are useful by themselves, other methods and tools appear along the development of FBS implementations. These tools are complementary to the FBS modeler and aim to make use of the advantages of the functional model. This section introduces some of the tools that relate more strongly to the model integration goal.

4.1 Qualitative Process Abduction System and Qualitative Process Reasoner

The Qualitative Process Abduction System (QPAS) [15] has as a goal to suggest to the designer physical features that can achieve a behavior, taking as input a description of the behavior (by means of a state transition table) the designer desires. The system finds for the designer suitable ways of attaining a certain behavior (i.e., physical features) from a set stored in a database. QPAS also offers a more "stepped" solution by suggesting and instantiating physical phenomena to build a new physical feature "on the fly".

After a defining the FBS model, the Qualitative Process Reasoner (QPR) [14] can simulate it qualitatively to verify that all the phenomena in the behavior network can be executed. With this simulation the system can detect possible "side effects". These side effects are physical phenomena which are not considered in the modeled behavior network, but that are activated by virtue of their enabling conditions (see section 3.2). The qualitative reasoning system is based on QPT. A simulation consists of generating all the possible state transition sequences (behaviors) from the model and comparing them to the desired (modeled) state transitions.

4.2 The Pluggable Metamodel Mechanism

The pluggable metamodel mechanism [18] aims to attain multiple model integration in design. Its implementation is the Knowledge Intensive Engineering Framework (KIEF) [6]. KIEF is supported over a knowledge base that stores concepts which include those used for the FBS model [13]. Objects from different modelers, like FBS or geometric CAD systems, are mapped to the objects of the knowledge base. This mapping is part of the knowledge base, and constitutes part of the knowledge about the modeler data. A metamodel of the system is built according to the ontology of the knowledge base. KIEF manages data transfer and consistency between modelers. Other possibilities of KIEF include suggesting modelers for a specific part of the model and creating models in a specific modeler by using information from other models. An application example of this process can be found in [18].

Next we briefly present the concepts of the physical concept ontology [13] that specifies how to build the knowledge base.

- Entity: Represents an atomic physical object.
- Relation: Represents a relationship among entities to denote static structure.
- Attribute: It is a concept attached to an entity. It takes a value to indicate the state of the entity.
- Physical phenomenon: Designates physical laws or rules that govern behaviors.
- Physical law: Represents a simple relationship between attributes.

All the concepts have a name that can be used to identify them. With the exception of the physical laws, all objects can have supers. Supers are objects from which the object inherits properties.

5 DISCUSSION AND PROPOSED IMPROVEMENTS

The past sections show the main features of FBS and other tools related to it. The way in which all these tools and concepts can be applied in order to obtain a concise integration of models for a design process is rather apparent, and is well documented in the references. This section discusses some details on which the authors consider worthy working more. It is not the purpose of this paper to evaluate the performance of the referenced implementations of FBS and its related tools, but to discuss the potential of such developments with respect to model integration and to propose improvements if available.

The next subsections analyze aspects that, according to the authors, have room for improvement and will increase the value of the FBS methodology. Section 5.1 presents a metamodel integration paradigm supported on FBS. Sections 5.2 and 5.3 relate to the topic of behavior simulation, and sections 5.4 and 5.5 are related to requirements 2 and 4 in section 2. Section 5.6 relates closely to requirements 5 and 7 in section 2.

5.1 Model integration over a metamodel

The ideas from section 4.2 revolve around mapping objects from different models to a metamodel. In KIEF the metamodel is built mainly by extracting information (e.g., connections between objects) from an FBS model. The metamodel also contains other information related to objects, such as physical phenomena, physical laws, and knowledge about modelers which are not modeled in FBS but make part of the knowledge base that FBS uses. The

proposal is to use the FBS model directly as the metamodel on which objects of other models can be mapped, by using the entire ontology of KIEF.

The "building blocks" of the FBS model must be detailed enough to allow representing the objects used in other models, but at the same time these blocks must act like components which result practical for the user and allow him to build a model quickly. As appears in section 7.2, at the moment the authors are taking first steps to build an FBS-based metamodel so the previous challenge can be cleared in the near future.

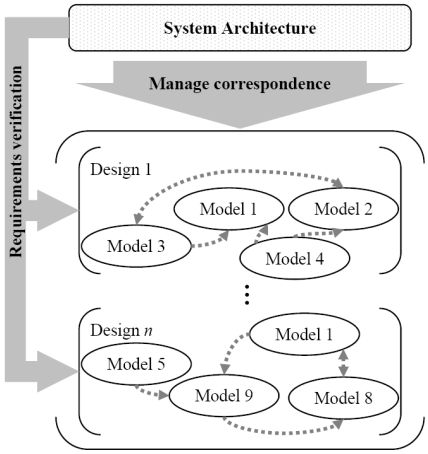


Figure 4: Diagram of the proposed model integration approach [4]

A rough diagram of the idea proposed here can be seen in Figure 4 (a more extensive explanation appears in an earlier work [4]). The FBS model is represented by the system architecture block. The lower part of the figure shows different stages of the design process (labeled "Design *x*"), represented by a set of models. Each of these models can correspond to single or multiple domains and to different levels of detail. The data correspondence between models is mapped in the metamodel and managed by the pluggable metamodel mechanism. The metamodel contains the necessary information to link a coherent, high-level, model of the system and the models mentioned before. Systems architecture is not clearly visible in the objects that compose the system. Some design rationale must be modeled and communicated to the users. Functional information is used to that end also.

5.2 Qualitative reasoning

A characteristic of FBS is that it is focused towards behavior simulation through computational means. Parting from an initial condition, the qualitative reasoning system generates all the state transitions reachable through the influence of the active physical phenomena.

Because qualitative reasoning works with rather incomplete information about the system, all qualitative reasoning algorithms face the problem of combinatorial explosion [1]. These algorithms implement mechanisms to reduce the number of combinations or to filter the results. The current implementation of QPR presents all the results, and thus some decision or filtering mechanism is desirable. As it is, the reasoning algorithm might, for example, instantiate the effects of gravity over every entity in a system. Though this is correct, in some cases the effects of gravity are negligible compared to other phenomena and the model grows unnecessarily. Next section proposes a partial solution for this.

5.3 Function specification and ontology

The use of FBS for the function symbol (i.e., the name of the function) is restricted to have an object that can be identified by the user and to which we can link the behavior information. The function symbol itself carries no meaning in the knowledge base, and the design intention is transmitted to the computer as the framework of connected concepts from the knowledge base.

The function symbol itself carries the most abstract part of the design intention in a function. The proposal is to describe the function symbol in terms of a predefined vocabulary that carries a meaning for the reasoning algorithms in the computer. Then, the algorithm can use such information to guide QPR towards the phenomena of interest. This does not solve the entire problem of combinatory explosion, but it may contribute to eliminate a good amount of spurious behaviors. In this way, we approach a natural-language-like functional representation [16] with a more formal background for "functional primitives" [19] to facilitate its application in an algorithm.

The basic idea is to identify the phenomena of interest as those that manipulate the main kind of energy specified by the function symbol, which in principle should be the biggest portion of energy flowing through the system, and therefore, the most representative in behaviors.

As an example of a restricted vocabulary to support function modeling in software, here we consider the work of [20], also mentioned by Chandrasekaran [19]. Although this particular ontology was developed from the device perspective of functional modeling [11], the vocabulary can describe a very broad variety of functions. It can be used in the verb-object format and also supports other constructions for functions symbols.

5.4 Function decomposition

Function decomposition is an essential part of the FBS model [14]. Nonetheless, the scope of function decomposition in FBS is more closely related to the ability of the algorithm to suggest physical features which are causally dependent than to give general guidelines about how a function must be decomposed.

Functional decomposition is in itself one of the core activities of the design process. A designer decomposes the required functions arriving to more concrete descriptions in every step. Having guidelines to perform such a crucial activity looks to formalize it so that it can be represented in a model in a reproducible way and to justify the decomposition choice to certain extent. The current problem can be pictured easily when performing a functional description of an existing system such as a permanent magnet DC motor. In this exercise, discrepancies appear even when the same person performs the functional decomposition of the same system several times (see left part of Figure 7, after the references). A reason for this is that a particular point of view can be used to decompose a function. On top of that, decomposing functions while remaining in the functional domain is very hard in practice, because at each level of decomposition some concreteness must be added [21], and this points towards a particular solution.

In the case of some of the device centered function models [22]-[24], functional decomposition is achieved by following internally the "flows" that a function processes, in a similar way to how functional block diagrams are created [17]. This decomposition approach is not applicable to FBS because its supporting ontology does not take functions, but processes, as the objects responsible for changes.

The authors propose to use a functional decomposition approach similar to the "zigzagging" presented in the axiomatic design theory [21], where the design parameters help to guide the decomposition process. For each function a corresponding behavior is assigned. As explained in section 3.2, behavior is carried out by physical features. Physical features carry information

about the involved processes (i.e., physical phenomena) and structure (i.e., entities and relations). Like this, it can be seen that the model contains functional and design parameter domains similar to those used in the zigzagging decomposition process of axiomatic design. The idea is to use this method to guide the functional decomposition process, and not to consider the details related to the independence axiom of the axiomatic design theory.

Looking back at the example in Figure 7, we see that both decompositions can be realized with a different choice of physical features (Figure 7 right), but in the case of the second decomposition the function "ConvertElectric EnergyToRotationalMechanicalEnergy" would be realized by the physical feature "IdealDCMotor" that contains less detail of what happens inside the motor at the second level of the decomposition (it uses the proportional relation between current and torque). The choice of a particular decomposition depends on the models used to represent its features.

5.5 Multiple level modeling and model consistency

Simultaneous modeling at several levels of detail is one of the potential uses that the authors see in functional models. By analyzing a function tree it is easy to identify how functions (or the interpretation we make of them) have the property of describing a consistent model of a system while at the same time more detail can be presented for some parts.

Here, a consistent model is understood as one that represents the modeled system without leaving any "holes" or unexplained parts in it. For example, a consistent model for a stepper motor might include a detailed dynamic model of the motor, geometric representations, and a "black box" controller model, while other consistent model can detail the controller structure and treat the physical part of the motor as a transfer function (which can be considered almost as a black box).

Though the tendency of some users of the FBS modeler is to associate functions to physical features only for functions which are not further decomposed into subfunctions [25], FBS does not impose this restriction. The proposal is to use the F-B relations and the mapping suggested by the pluggable metamodel mechanism at different levels of detail (i.e., different levels of function hierarchy) so that a user can build and view a consistent model of the whole system while looking in detail some parts of the model.

5.6 Model and data standardization

Model and data standardization are factors that strongly influence the use and acceptance of a system modeling implementation. This happens because standards are made accessible to more people by the organizations, and also because good standards tend to fill in the needs of industry better than other solutions. This is partially explained by the fact that most standardizing organisms are born from industry. The project in which the present work is carried out is closely related to industry, and thus, the advantages of standardization must be exploited as much as possible, though this is almost always desirable.

FBS defines a semantic structure for the knowledge base, but it does not define any data structure for it and it is not restrictive in that sense. The KIEF implementation is programmed in Smalltalk language, and the data of the knowledge bases is specific for that implementation. These choices were driven in part by the origins of KIEF in the research community, where basically the developers are the main users of the implementation.

Some mention to standards for data representation appears in literature about the pluggable metamodel mechanism [18]. There, STEP (ISO 10303) is mentioned as an example of standard data representation that can simplify retrieving data from complex products. The STEP standard is widely used by CAD systems mainly to exchange information about geometry, though the standard allows representation of other information relevant for product design such as dimensioning, configuration management data, and assembly data.

In recent years the extensible markup language (XML) has gained tremendous popularity. XML formatted data can be found in a broad range of applications such as web pages, modeling languages (e.g., UML), and mathematical notation (e.g., MathML). The STEP standard does not fall behind, and it is currently implementing an XML based representation for its application protocols (i.e., Part 28 XML). XML forfeits characteristics such as terseness in favor of qualities like extendibility, broad applicability, and human readability.

Apart from the data representation format, model standardization is also desirable. As an example, most 3D geometry modelers in CAD tools through the years have arrived to an implicit agreement in the available operations. This agreement is also related to the data in the representation models. That allows a user to quickly switch tools and still be able to produce the desired geometry.

One standard that is gaining strength in the modeling field is the Unified Modeling Language (UML). Though initially and most broadly used to describe software products, these days some of its "profiles" (which contain restrictions as well as extensions) are used to represent business models and real-time systems. The relatively new profile of Systems Modeling Language (SysML) [26] seems suitable to represent most of the information used in systems' design. It is also worth mentioning that part of the developing group of SysML also belongs to the group that develops STEP [26]. SysML has been successfully applied as part of an integrated design platform in works like [27] and [28].

At this point, the proposal is to implement FBS in SysML. This will get FBS in the path of standardization for both, data representation and modeling language.

6 INTEGRATION OF MODELING TOOLS

Gathering the ideas from section 5, we present the general approach to implement the integration of modeling tools using an FBS model. The function and behavior layers of the FBS model (cf. section 3) form a metamodel that plays the main role in integration. Till now, the proposal mainly addresses definitional integration. The metamodel is based on knowledge about physical concepts. The models, being abstractions of reality, are compatible with such concepts. Like this, a model-independent metamodel can be established. On the other hand, since current tool data and format are not standard, additional knowledge about this is necessary to integrate the tools. Using an XML compatible model aims towards data compatibility in the future, though this format is already supported by many tools.

The objects represented in the models are associated with the objects in the behavior layer. For example, a solid geometry represented in a CAD model can be associated with a "SolidBody" entity. Attributes in the model, like the volume of the solid, can be mapped directly to attributes of the entity. At the attribute level, a network of constraints is built using the laws attached to the phenomena. This network and the state layer may be stepping stones for procedural integration, providing information to coordinate the manipulation of the models.

The models can represent different domains and degrees of detail. The functional layer is related to the models

through the behavior layer. In this way, models are linked to a layer where their differences become less relevant. From another perspective, the functional layer also communicates to the user the role of a model in the design, supporting decision making. Diversity in the models' detail level is addressed by the hierarchical representation of the architecture, both in the function and behavior layers.

7 CURRENT PROGRESS AND FUTURE WORK

The first step was choosing FBS as a base for the functional models to be used in the project, after studying the basics of several developments related to functional modeling that can be found in literature. The readers should refer to [11] and [19] for a review of functional modeling approaches, and to [29] for an overview about functional reasoning.

Currently the authors are working to implement the physical concept ontology of KIEF (which contains the ontology of FBS) in SysML. This is done keeping in mind the ideas of section 5 while paying special attention to the model integration aspects.

To illustrate the implementation, we show part of the model corresponding to a permanent magnet DC motor. For the models here the authors used the commercial tool MagicDraw UML and its SysML plug-in.

7.1 FBS in SysML

Obtaining a formal description of how to develop an FBS model in SysML is an important first step for the implementation of functional modeling in the framework of the project. To properly understand the SysML objects of the mapping the reader should refer to the SysML specification [26]. A first proposal for such "mapping" is presented in this section. Italicized terms in the next paragraphs correspond to SysML terminology. *Block* is the term used for classes in SysML, and thus is used extensively. Most definitions are done at the class level, and thus can be reused to define instances of the objects that will be part of the actual model.

Entity: An entity can be mapped to the SysML *block* class (Figure 8.a). In the model, a *block* for the entity is created by specifying the *blocks* that correspond to its supers, and then the details are added to the new class. Instances of the class with specific values will be used in the model. The example contains entities such as "rotor" "coil", and "shaft", which in turn are children of the entity "Solid Body".

Attribute: Attributes described by numeric values (like moment of inertia, torque, and angle) can be represented as *ValueTypes*. Units and dimensions can be defined for a *ValueType* (see Figure 9). For the case of attributes that correspond to the derivative of other attributes (e.g., acceleration, velocity, and position) a *directed association* named "derivative" can be placed from the attribute to the derivative of the attribute (e.g., from position to velocity). For other attributes that describe special conditions of an entity (e.g., matter state), *enumerations* that contain the set of values/string descriptions (e.g., solid, liquid) can be defined. Attributes are placed directly in the entity classes (the *blocks*) as *properties* (SysML uses *value properties* and other kinds of *properties*). Values for the attributes can be assigned to the instances.

Relation: Relations can be represented (Figure 8.b) as *association blocks* that connect the involved entities (*blocks*). The relations do not appear from the side of the connected entities, but as the *type* of the *memberEnds* in the *association block*. The relation holds a *reference participantProperty* for each connected entity.

Physical law: Physical laws are represented by mathematic expressions. The idea is to store here the qualitative relations, so that the QPR can access this information from the metamodel. To represent this information SysML includes a very specific object called *constraint block* (Figure 8.c). With *constraint blocks*, systems of equations can be built by connecting the *ports* of several *blocks* in the *parametric diagram*, linking variables between mathematic expressions. The expressions are placed as *constraints* in the *constraint blocks*. Also other *constraint blocks* can be nested inside the *constraint block* as *constraintProperties*. The variables that appear in the expression are defined as *constraintParameters* of the *constraint block*. Figure 8.c depicts a constraint relating three parameters.

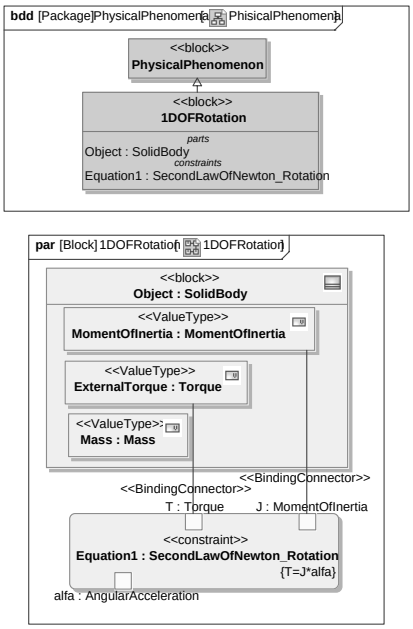


Figure 5: Physical phenomenon representation. Block representation (above) and statements definition (bellow)

Physical phenomenon: This is one of the most complex knowledge units in FBS. Therefore, special attention is required to map this structure in SysML. A physical phenomenon is also represented as a SysML *block* (Figure 5). A description of the mapping for specific parts of the physical phenomenon is next:

- Name: As all UML objects, the block has a name.
- Supers: Supers of a block are represented by a *generalization relation*.
- Entities: Entities are *part properties* of the *block*, *typed* by the *blocks* that define the entities.
- Attributes: They can be extracted directly from the related physical laws and entities.
- Physical laws: Defined as *constraint parameters*.
- Statements: *Constraint parameters* already tell us which attributes are involved in the physical phenomenon, and they are connected through *binding connectors* to the attributes (*value properties*) in the entities (*blocks*).

The physical phenomenon "1DOFRotation" in Figure 5 binds attributes of "Solid Body" to their corresponding *constraint parameters* in the "SecondLawOfNewton_ Rotation" physical law.

Physical feature: Physical features can be represented as *packages* that contain instances of the necessary physical phenomena, entities and relations. This limits the use of the physical feature as an object because it already uses instances, but allows a direct contact with

the entities inside a physical phenomenon. The feature "ShaftCoupling" (Figure 6) contains a phenomenon "Unification_Rotation" that associates the torque attribute of three assembled entities (motor rotor, coupling, and output shaft) to a constraint to computes the total torque transmitted through the assembly.

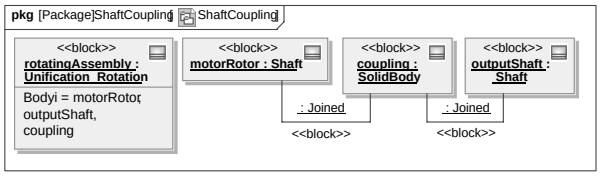


Figure 6: Physical feature representation in SysML

Function: Functions are represented by SysML *activities* (Figure 7, left). Modeling of F-B relations is done by *allocating* functions to the respective features. When modeling, the physical features are placed in the model, and unifying relations are created between entities from different features that represent the same real object. Like that it is possible to create a consistent model from a group of features.

7.2 Future work

The next step is to use the implementation scheme proposed in section 7.1. The goals of that step are to test if real systems can be modeled with the proposed implementation, to gradually build a knowledge base, and to refine the required modeling steps. For the modeling steps, special attention must be put in the way in which the user must input information to the model.

Another aspect to investigate is the choice of appropriate visualization methods for the model. Visualization of models is important to facilitate understanding and appeal of the model, which strongly influence the decision of using a model or not using it.

8 CONCLUSIONS

FBS has good potential to work as a metamodel over which other models can be mapped. However, the corresponding information about the modelers must be added. Also, more work has to be done to model software-related aspects, as the work here has focused so far on representation of physical objects.

Though definitive choices about the correspondence for some elements are still to be made, the current work proves that SysML is powerful and flexible enough for building in it meta-models that support model integration.

About the modeling process in SysML it is possible to conclude that, after mapping some components of the physical concept ontology, the authors could verify the flexibility of SysML to represent a wide variety of concepts. Nonetheless, such flexibility can cause difficulties in the choice of mapping for a component or term. Some diagrams, like the parametric diagrams, become easily cluttered when using more than ten blocks or so, and this cannot always be avoided with packaging.

9 ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR)' of the Dutch Ministry of Economic Affairs.

10 REFERENCES

[1] Barr, A., Cohen, P. R., 1989, *The Handbook of Artificial Intelligence*, Vol. 4. Chapter 21. Los Altos, CA: William Kaufmann, Inc.

[2] Rodenacker, W., 1971, *Methodisches Konstruieren*, Springer-Verlag, Berlin.

[3] Pahl, G., Beitz, W., 1988, *Engineering design: A systematic approach*, Springer-Verlag, Berlin.

[4] Alvarez Cabrera, A. A., Erden, M. S., Foeken, M. J., Tomiyama, T., 2008, "High Level Model Integration for Design of Mechatronic Systems," *proceedings of IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*. Beijing, China. pp. 387-392.

[5] Tomiyama, T., Umeda, Y., 1993, "A CAD for functional design," in *Annals of the CIRP'93*, 42(1), pp. 143-146.

[6] Tomiyama, T., Umeda, Y., Ishii, M., Yoshioka, M., Kirayama, T., 1996, "Knowledge systematization for a knowledge intensive engineering framework," *WG 5.2 Workshop on Knowledge intensive CAD-1*, pp. 33-52.

[7] Derelöv, M., 2008, "Qualitative modeling of potential failures: On evaluation of conceptual design," *Journal of Engineering Design*, 19(3), pp. 201-225.

[8] Dolk, D. R., Kottemann, J. E., 1993, "Model integration and a theory of models," *Decision Support Systems*, 9(1), pp. 51-63.

[9] Cutkosky, M. R., *et al*, 1993, "PACT: An experiment in integrating concurrent engineering systems," *Computer*, 26(1), pp. 28-37.

[10] Geoffrion, A. M., 1989, "Reusing structured models via model integration," *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, 1989*, Vol.III: Decision Support and Knowledge Based Systems Track, pp. 601-611.

[11] Erden, M. S., Komoto, H., van Beek, T. J., D'amelio, V., Echavarria, E., Tomiyama, T., 2008, "A review of function modeling: Approaches and applications," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(2), pp. 147-169.

[12] Forbus, K., 1984, "Qualitative process theory," *Artificial Intelligence*, 24(3), pp. 85-168.

[13] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T., 2004, "Physical concept ontology for the knowledge intensive engineering framework," *Adv. Eng. Inf.*, 18(2), pp. 69–127.

[14] Umeda, Y., Ishii, M., Yoshioka, M., Tomiyama, T., 1996, "Supporting conceptual design based on the function-behavior-state modeler," *AIEDAM,* 10(4), Sept. 1996, pp. 275-288.

[15] Ishii, M., Tomiyama, T., Yoshikawa, H., 1993, "A synthetic reasoning method for conceptual design," *IFIP World Class Manufacturing '93*, Amsterdam, pp. 3-16.

[16] Chakrabarti, A., Bligh, T. "An approach to functional systhesis in mechanical conceptual design. Part I: Introduction and knowledge representation," *Research in engineering design*, 6(3), pp. 127-141.

[17] European Cooperation for Space Standardization, 1999, *Space engineering – Functional analysis (E-10-05A*), (http://esapub.esrin.esa.it/pss/ecss-ct05.htm)

[18] Yoshioka, M., Sekiya, T., Tomiyama, T., 2001, "An integrated design object modeling environment - pluggable metamodel mechanism -," *Turk J Elec Engin*, 9(1), pp. 43-62.

[19] Chandrasekaran, B. "Representing function: Relating functional representation and functional modeling research streams," AIEDAM, 19(2), pp. 65-74.

[20] Hirtz, J., Stone, R., McAdams, D., Szykman, S., Wood, K., 2002, “A functional basis for engineering design: Reconciling and evolving previous efforts,” *Res. Eng. Des.*, 13(2), pp. 65–82.

[21] Suh, N. P., 1990, *The Principles of Design,* Oxford University Press, Oxford.

[22] Stone, R., Wood, K., 2000, “Development of a functional basis for design,” *ASME J. Mech. Des.*, 122(4), pp. 359–370.

[23] National Institute of Standards and Technology, 1993, *Integration definition for function modeling (IDEF0)*, [online] (http://www.idef.com/pdf/idef0.pdf).

[24] Wood, W., Dong, H., Dym, C., 2004, “Integrating functional synthesis,” *AIEDAM*, 19(3), pp. 183-200.

[25] van Eck, D., McAdams, D., Vermaas, P., 2007, “Functional decomposition in engineering: A survey,” *Proceedings of the ASME 2007 IDETC/CIE*, Las Vegas, Nevada, USA.

[26] Object Management Group, 1999, *OMG Systems Modeling Language (OMG SysML™), V1.0* , [online] (http://www.omg.org/cgi-bin/apps/doc?formal/07-09-01.pdf)

[27] Peak, R., Burkhart, R., Friedenthal, S., Wilson, M., Bajaj, M., Kim, I., 2007, “Simulation-based design using SysML: Celebrating diversity by example,” *INCOSE Intl. Symposium, San Diego*, [online] (http://eislab.gatech.edu/pubs/conferences/2007-incose-is-2-peak-diversity/2007-incose-is-2-peak-diversity.pdf)

[28] Tactical Science Solutions Inc., 2007, *Quicklook final report*, [online] (http://www.tacticalsciencesolutions.com/files/05-30-07%20Quicklook%20Final%20Report%20v1.19.pdf)

[29] Far, B. H., Elamy, A. H., 2005, “Functional reasoning theories: Problems and perspectives,” *AIEDAM*, 19(2), pp. 75-88.

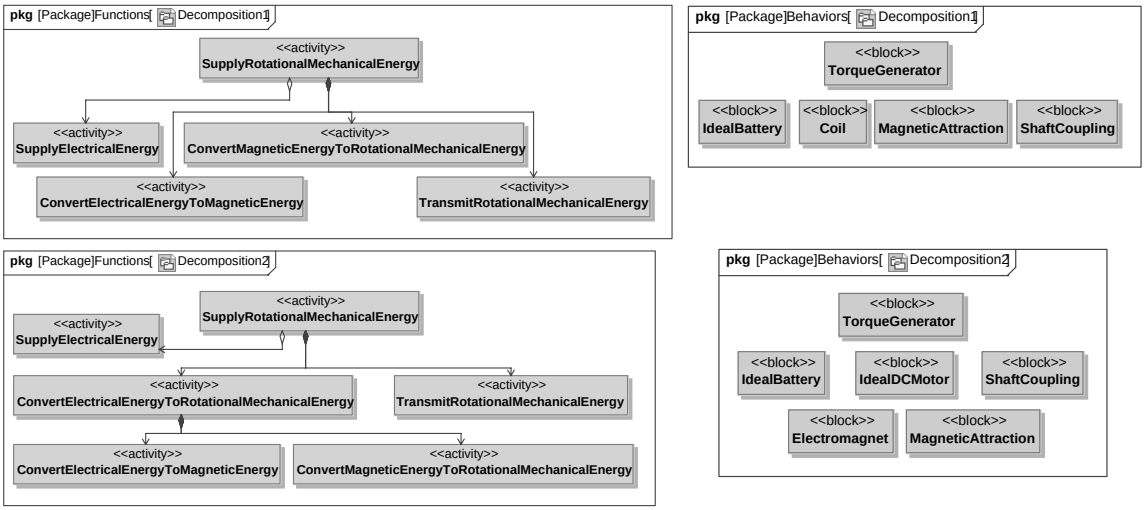
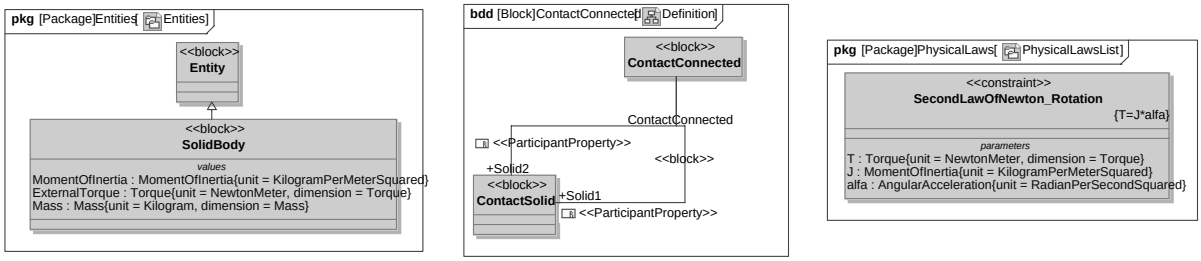


Figure 7: Example of two different functional decompositions for a permanent magnet DC motor. Function trees (left) and corresponding physical features (right)



(a) (b) (c)

Figure 8: SysML representations for (a) entities, (b) relations, and (c) physical laws

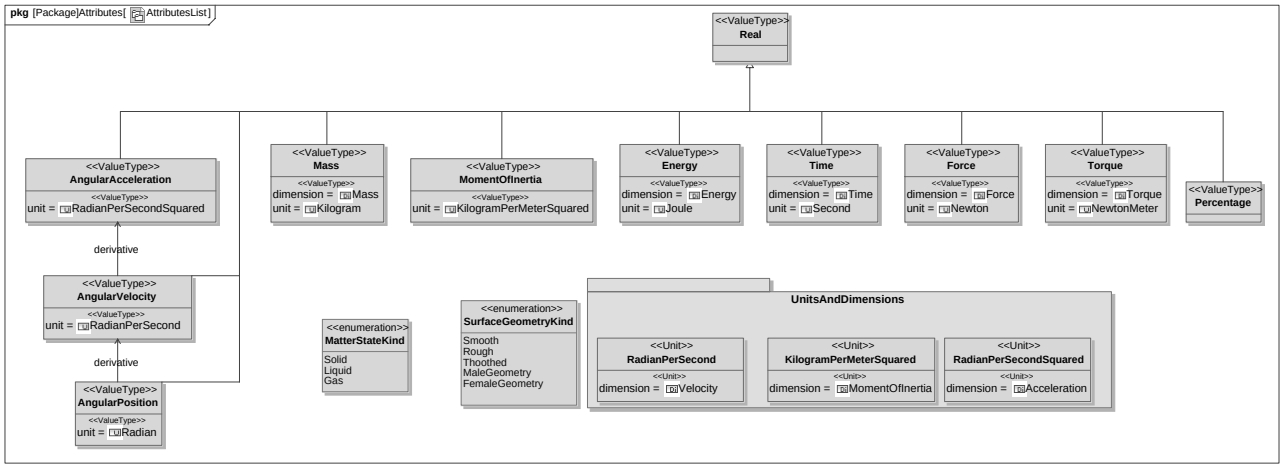


Figure 9: Representation of attributes and definition of units and dimensions in SysML