MODEL TRANSFORMATIONS FOR KNOWLEDGE BASE INTEGRATION WITHIN THE FRAMEWORK OF STRUCTURAL MODELLING

Janis Grundspenkis
professor of systems theory
Janis Jekabsons
research assistant

System Theory professor's group, Riga Technical University, Riga, Latvia
E-mail: jgrun@itl.rtu.lv, jajek@parks.lv

Abstract

Many different methods and models have been developed and successfully used for diagnosis problem solving. At the same time reuse and integration of different approaches and models are open questions till now. Practically it is impossible to use a model developed earlier in a new application without building it from scratch and using a description language specific for a corresponding method. In this paper we suppose that a set of knowledge bases constitutes a repository of syntactically different models developed for diagnosis problem solving. In the broad sense the problem of reuse of a model captured in a knowledge base is reduced to the problem of a model transformations from one specific modelling method to another. Several approaches to model transformations are outlined. This paper reflects results obtained in a pilot project where transformation methods and algorithms have been worked out for structural models of complex technical systems developed in the structural modelling and in the multilevel flow modelling approaches.

Keywords: structural modelling, multilevel flow modelling, model transformation, knowledge base integration

1. Introduction

Nowadays technical systems often are hybrid and consist of different types of components: software, hardware, electronic, mechanical, hydraulic, pneumatic and even chemical [1]. Many different modelling methods have been developed and used for a model-based diagnosis of complex technical systems. Some researchers (for example [2]) argue that for complex hybrid systems better diagnosis results may be achieved by integration of different modelling approaches. There are many obstacles along the way towards integral models for diagnosis. First, no universal models are known and it is doubtful that such models may be developed for a large-scale classes of technical systems. Second, frequently models are so specific that it is hard to imagine how they may be integrated. Thus, integration of different methods and models is a problem itself that must be solved to achieve more efficient diagnosis.

This paper deals only with one of possible model transformations, namely, structural model transformation. The reason why this model transformation is chosen is that most of the modelling methods (structural, functional, causal, behavioural and others) are based on or have to deal with a structure of a system. A structural model of the technical system is a description of system's physical structure in terms of specified components and connections between them. The components represent a physical objects of the real system and the connections represent a physical flows. It is well known that systems structure determines possible functions, behaviour and causal relationships in the system. That is why modelling of structure, in particular, in form of various modifications of tree structure or networks, are widely used in technical systems' diagnosis. Considerations of the role and the importance of a structural model is the corner-stone of the

proposed approach for integration of different modelling methods which are based on concepts of structural models.

Current results obtained in the pilot project of structural model transformations allow to carry out model transformations only if they belong to quite similar modelling methods based on different kinds of structures. In particular, the paper deals with structural model transformations that are developed for two modelling methods: the Multilevel Flow Modelling [3] and the Structural Modelling [4].

These two modelling methods have been chosen taking into account several aspects. First, we look at the problem of method and model integration from the point of view of knowledge base integration. Let suppose that there is a collection of different methods and models for model-based diagnosis problem solving. All knowledge is stored in corresponding knowledge bases, i.e., a collection of knowledge bases are available (we call this collection "a repository of models"). There may be several cases where knowledge base integration is a pressing problem. The most important case is that where powerful knowledge processing is necessary but from different modelling methods only one supports all kinds of reasoning. In other words, all already existing models (built using different modelling methods and languages) must be transformed and integrated in one particular model. The integration of knowledge bases can not be a straightforward task due to the different schemes used for knowledge base construction and difference of languages used for model description. Consequently, the reuse of some early built models depends on success of model integration or, in other words, it depends on possibilities of knowledge base integration. From this aspect it must be pointed out that structural modelling supports various kinds of reasoning [5] while multilevel flow modelling (MFM) has much weaker possibilities. At the same time the MFM is rather widely used method for diagnosis problem solving [6, 7] and its applications to real life problems allowed to create a number of models of complex technical systems which can be successfully reused in other applications.

The contents of the paper are organised in five sections. First section is devoted to the general principles of model transformations. In the second section a short discussion of structural modelling concepts and knowledge bases used to support various kinds of reasoning are presented. In the third section we give a short overview of the MFM. Similarities of model elements are discussed in the fourth section. The structural model transformation algorithms are presented in the fifth section. We conclude with the short discussion of obtained results and with the outline of further work.

2. Principles of Structural Model Transformations

Consider the situation of model development of a complex system. Practically there are two alternatives: to build a model from scratch or to reuse some already developed, tested and approbated models. A wide variety of modern technical systems that often are hybrid ones and consist of components of different types have caused the development of different methods and models for diagnosis problem solving. Within each framework many models have been developed which, in fact, are unique in sense that they are used only in one particular application. Even if it is known that suitable model for some part of the complex system already exists one can not use it because diagnosis requires a model of the system as a whole (fragmentary models without their integration are helpless). So, the question is how to reuse already developed models if they are appropriate in the specific diagnosis problem solving. Suppose we have a repository of technical systems' elements and subsystems together with their models which are taken from sets of different models of different modelling methods. In this case models and methods in a repository may be considered as a set of knowledge bases where the user can find the most appropriate models for his/her specific task. A repository may be utilised for reuse and model reconstruction but additional obstacles appear, namely, different schemes used for knowledge representation. A simple

transformation of knowledge bases by rewriting their contents using one common knowledge representation schema, for example, rules, frames or semantic nets followed by integration of knowledge bases will not allow to achieve the goal - to develop a model of the complete system as a whole. The solution must be found at the model level. The idea is that first of all the model integration must be provided. Due to the wide variety and complexity of the technical systems, especially, the hybrid ones, it is hard to expect the appearance of an universal modelling method at least in near future. That is why the model reengineering approach seems to be more perspective. In this paper the notion “model reengineering” is used to denote the procedures of multiple model integration.

In general there are three options for integration:

1) model integration using one common and universal interface (see Figure 1);
2) model integration using many pair-wise different interfaces (see Figure 2);
3) model integration using different types of transformations (see Figure 3).

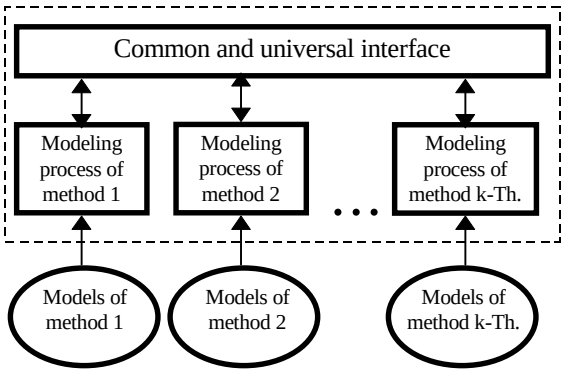


Figure 1. Model integration using common and universal interface

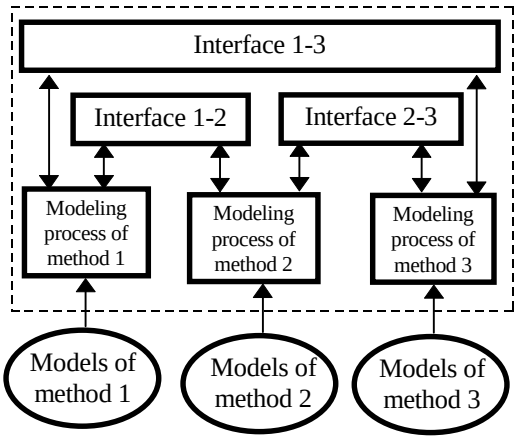


Figure 2. Model integration using many pair-wise different interfaces

In all figures the term “modelling process” represents model-based reasoning that is a dynamic process utilising knowledge represented by models. In this context models are passive components and modelling is an active component.

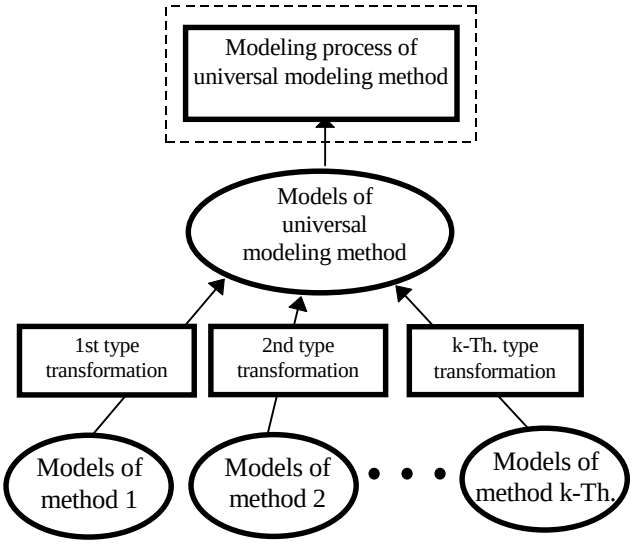


Figure 3. Model integration using universal modelling method

In the first case it is needed to design universal enough interface that provides information exchange among models. The complexity of an interface is proportional to the number of models. The implementation of the second case also is limited by the increasing number of needed interfaces. The third option suggests to develop as many transformation methods as there are modelling methods. This approach is effective in cases when considered modelling methods are rather similar.

Common features of all three model integration options are that, in fact, model transformations change a model of one modelling method into a model of another modelling method. Transformation always causes some losses of information (knowledge) captured in models. The determination of permissible level of knowledge losses is a hard issue. It is an area for future research and is not discussed in this paper. In ideal case all knowledge that is captured in an original model (an input of transformation) and is necessary for use in diagnosis of technical systems is transformed into a destination model (an output of transformation). The transformation must transform not only the elements of the models but also should remain the semantical aspects for to be able to use this destination model in the reasoning of the destination modelling method. Figure 4 illustrates basic principles of transformation.

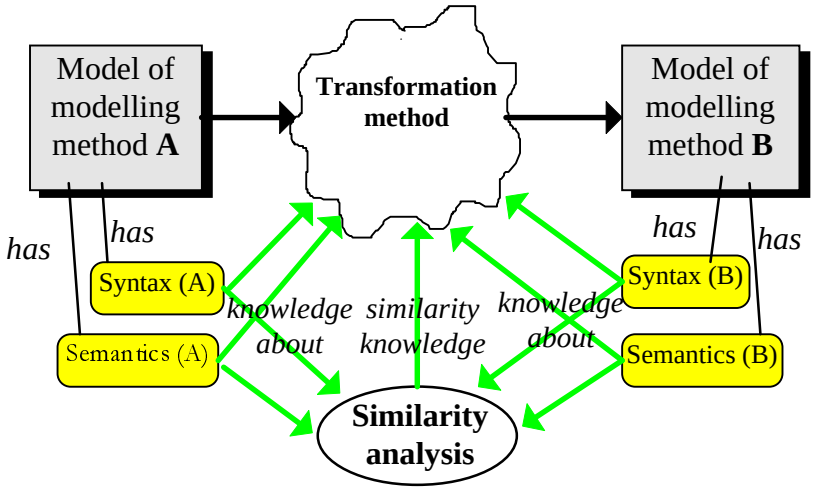


Figure 4. Basic principles of transformation

Each of modelling methods have its own syntax and semantics of models. Note the significant difference: syntax of a model can be represented in a formal way but usually it is much harder to give a formal description of semantics of modelling elements and models as a whole.

In the development of transformation method it is necessary to take into account syntax, semantics and also similarity of elements of both modelling methods. Similarity analysis of elements is based on the principle of finding for each element of transformation input an appropriate element of transformation output. It is worth to stress that, in general, all kinds of relations may exist between transformation input and output (one-to-one, one-to-many, many to one and many-to-many). In model transformation similarity means knowledge about replacement of elements of one model by elements of another model. Process of similarity analysis also needs knowledge about syntax and semantics of models. The examples of similarities will be shown further in this paper.

3. Structural Modelling Concepts and Construction of Knowledge Bases to Support Reasoning

Structural modelling is a systematic model-based knowledge acquisition framework [8]. An abstract causal domain model built within the framework of structural modelling consists from three models, namely, a model of morphological structure (MSM, in brief) and two kinds of models of functional structures (FSM). Building of these models is essentially a method for encapsulating domain knowledge into small, independent, composable and decomposable units of knowledge (see [5, 8]). The basic units of abstract domain model are called *objects*. These primitives have *input* and *output contacts* as it is shown in Figure 5. When interpreted in an application domain, abstract objects correspond to physical components of a given system, and contacts represent their inputs and outputs, respectively. Thus, each object captures knowledge about names of its causal inputs and outputs which is the mechanism for linking objects together. The connection of one object's output to another object's input is the only path by which the components may interact. The interactions between objects are called *flows*.

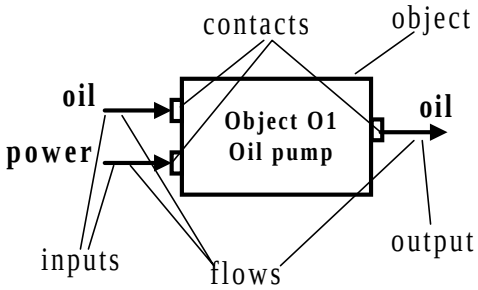


Figure 5. An example of an object

In general, an expert (user) does not need to be able to specify the connections of objects because within structural modelling framework this is done by the Automated Structural MOdelling System (ASMOS). The execution of corresponding ASMOS tool [4, 8] allows to get a *model of morphological structure* (MSM) which is visualised as a diagram or a digraph. The MSM represents a physical structure of a given system, i.e., it represents structural relationships which can be reasoned in logic. The essence of this kind of reasoning, called structural reasoning, is the exploration of paths and cycles between objects.

In fact, the MSM is insufficient for diagnostic and predictive reasoning because it doesn't contain the knowledge about causal relationships in a direct form. To support these kinds of reasoning the MSM is transformed into a functional model. Two models of functional structure are defined, namely, a *functional*. *model in a space of functions* (FSM FS) and a *functional model in a*

space of parameters (FSM PS). To support reasoning about processes proceeding in physical elements, the FSM FS encodes knowledge about functions of a system that operates in normal conditions, i.e., when a given system has no faults. If visualised, the FSM FS is displayed by a corresponding diagram.. Expert's understanding of how a given system works is organized as a representation that describes purpose why particular function is accomplished. Arcs of the FSM FS represent cause-consequence relations between functions. In this sense the FSM FS is a representation language for a causal dimension. To represent functions we use two notions - *behaviour* of a component and a *behaviour state* of a contact of a given component. Behaviour specifies an action that component performs upon its "substance", i.e., it specifies how a reaction to a given stimulus is achieved. In other words, behaviour is a characteristic of an input/output relation which may be considered as a causal explanation how structure insures the fulfilment of functional specifications. Behaviour states, in turn, specify how flows "act" in corresponding inlets and outlets of components. So, each object described in MSM have its own behaviour and object's contacts are mapped on corresponding behaviour states. For example, the behaviour of a fuel pump, if it operates normally, is "pumping" (in this case fuel may be considered as default "substance"). The behaviour state of input contact of a fuel pump is "fuel flows through the input of fuel pump" and the behaviour state of output contact of a fuel pump is "fuel flows through the output of fuel pump". Having notions of behaviour and behaviour state the question is "what is the semantic of a function?" We suppose that function closely relates with a purpose (a goal) why the designer has included certain component in a system. More precisely, f*unction* captures the intended purpose of the object (component), that is, function specifies what the response is to a given stimulus. In other words, function specifies what is the result of proper behaviour (functioning) of a given component. So, our notion of function is similar with that used in multilevel flow modelling [9]. Thus, "to provide fuel flow under pressure" is the purpose why a fuel pump is included in the system. To achieve this goal, the pump must behave properly, i.e., its behaviour must be "pumping". It is worth to stress that in the structural modelling framework a topology of the FSM FS is derived from the MSM automatically [5, 8].

When some changes occur in the system's components or some links between components change, the result usually is that the functional effectiveness decreases or we may notice that the behaviour of the system deviates from that of normal operation. In terms of structural modelling that means changes of behaviour states and, as consequences, changes of behaviour of the system as a whole. In diagnosis we need to detect changes in normal operation, and if they are detected, we need to make a decision "why system's behaviour have changed". We also can consider the inverse task, namely, to find the answer to the question "what will happen if definite change will occur?" In both cases we need a model to support these two kinds of behaviour reasoning, called diagnostic and predictive reasoning, respectively. In the framework of structural modelling behaviour states as well as functions are qualitative characteristics (because they are not characterised by numerical parameters) of processes taking place in physical components. In engineering practice the notion "variables" (or parameter values) is used to get quantitative characteristics. It is rather obvious that each pair of serial behaviour states, i.e., each function must be characterised by a certain variable to allow to detect changes in system's behaviour. Thereby, we can construct the second model of functional structure called a *model of functional structure in a space of parameters* (FSM PS) using a transformation from the FSM FS to the FSM PS. In ideal case (not always held in real world) only one parameter corresponds to each function. Thus, the transformation is straightforward, i.e., the topology of the FSM PS will be isomorphic with the topology of the FSM FS. This makes the systematic problem domain knowledge acquisition supported by structural modelling much easier because all topologies of models are generated automatically using ASMOS and the user (problem domain expert) adds semantics in accordance with interpretation of structural modelling primitives in a particular problem domain.

The acquired knowledge about morphological and functional structures represented by a hierarchy of frames is captured in *a topological knowledge base*, or the TKB in brief [10]. In order to use well known reasoning mechanisms of AI, namely, forward and backward chaining. the TKB is used to obtain a deep knowledge rule base (DKRB). To achieve the transformation from the TKB to the DKRB the model of functional structure in a parameter space is decomposed into substructures. This transformation is described in [8]. These substructures are called *event trees*. Each event tree reflects cause-consequence relations between a subset of possible faults and changes of parameter values, i.e., failures. There is the straightforward transformation of event tree into a subset of IF - THEN rules or, so called, cause-consequence rules (C-C rules). The set of cause-consequence rules that compiles the DKRB is a deep causal model of the system under diagnosis. The main distinction between the DKRB and the commonly used rule base in diagnosis expert systems is that the DKRB can support reasoning with the goal to establish what really is going on in the system under faults, i.e., we can get a "full picture" of failure propagation. In commonly used rule bases the set of rules, in fact, represents only so called shallow knowledge acquired from experience. Each rule may be considered as a mapping of a subset of faults to a subset of observed symptoms without any information about the real chain of failure propagation from causes (faults) to consequences (symptoms). Paper [10] has shown an easy way how the set of rules stored in the DKRB may be transformed into the set of production rules.

It is important that structural modelling proposes an automated way of deep knowledge rule base building [8] because it is based on the formal algorithms of decomposition of the FSM PS and the change of knowledge representation schema from the event tree form to the C-C rule form.

So, the architecture of the knowledge base used in structural modelling includes two parts:
1) a topological knowledge base (TKB) that captures all acquired knowledge and consists from three models: the MSM, the FSM FS and the FSM PS described above;
2) a deep knowledge rule base (DKRB) that supports various kinds of reasoning about the investigated system.

For already explored systems all models that have been built should be stored in a repository. A repository makes it possible to reuse models when necessary. The effectiveness of a process of new model building depends not only on the ability to identify a suitable model when specifications of current situation are available but also from the number and variety of already modelled systems. That is a reason why an addition of models is so important. In this paper we discuss one of the options, i.e., how within the structural modelling framework use models that have been built for different applications and have been implemented using different modelling techniques. These models are added to the topological knowledge base. This process is interpreted as a knowledge base integration based on model transformations. Due to the limited scope of this paper we describe knowledge base integration only for the case if we have models which are developed using multilevel flow modelling techniques.

4. A Short Overview of the MFM

Multilevel Flow Modelling (MFM, in brief) is developed by professor Morten Lind at the Technical University of Denmark (see [3, 6, 9]). Its purpose is to model a system as a man-made system with certain intentions. An important feature of the MFM is the representation of system at multiple levels of abstraction where system description is made in terms of goals, functions and physical components (devices). This aspect is equal to the description of system along two dimensions, namely, a means-ends and a whole-part dimensions. Means-ends dimension describes how the certain goals of a system are achieved by functions and what components carries out these functions. The complexity of a system or its parts are reduced by using abstraction principles along a whole-part dimension. The abstraction and decomposition principles are used to represent system's

parts at the proper level of granularity. The MFM uses a set of previously defined primitive function concepts related to the processing of material, energy and information flows. Using these concepts the knowledge of process plant is represented. The MFM deals with five elements of modelled system which are displayed in Figure 6:

1) goals - represent the intentions of designed system;
2) functions - show how goals can be achieved. There are six functions: Source, Sink, Transport, Barrier, Balance and Storage;
3) functional structures - connected functions of one certain level of abstraction, e.g., water circulation structure, thermal energy supply structure; etc.
4) components - show how the functions of the system can be implemented;
5) relationships - connect goals to functional structures, goals to goals, components to functions and functions to functions.

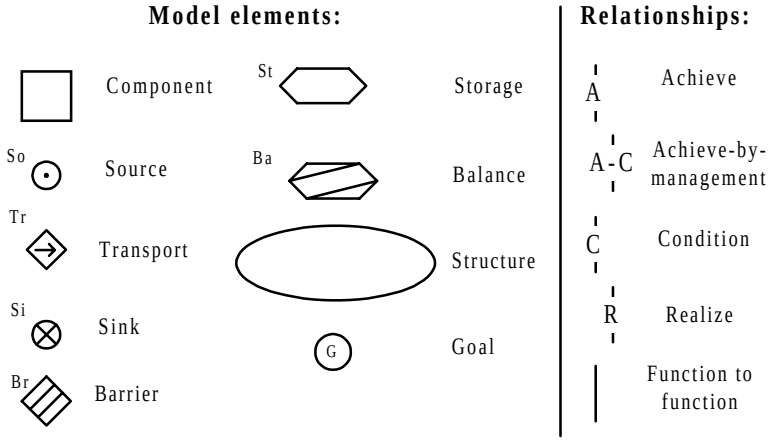


Figure 6. Graphical representations of model elements and their relationships

5. Similarities of Model Elements

In accordance with the proposed transformation method for each model element of one modelling method it is necessary to find analogous or similar element of another modelling method. If such element does not exist then there is a need to find a possible construction of number of elements to replace this one. If this effort also fails then the transformation of the particular element is impossible (knowledge losses take place). The alternative way is to change syntax or/and semantics of the destination modelling method with the goal to adapt it for element transformation. In practice it may be consuming work or even impossible task. Lets look at some examples of element transformations from the Multilevel Flow Modelling to the Structural Modelling. The transformation of a balance function of the MFM to the object of the MSM is depicted in Figure 7. It is easy to see one-to-one correspondence between these two primitives. Transformation of a storage function is analogous.

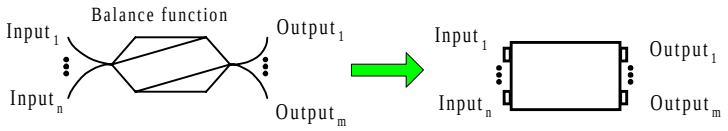


Figure 7. The transformation of a balance function

It is very simple also to transform a source function to the object with only one contact - the output contact. Similarly a sink function is transformed to the object with one input contact and a transport function - to the object with one input and one output contact.

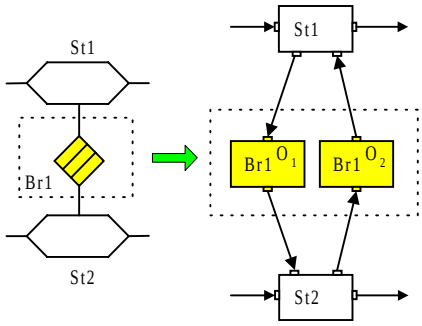


Figure 8. The transformation of a barrier function

The transformation of a barrier function (Br1) concerns determination of one element similarity with the construction of two elements - objects O_1 and O_2. This case is showed in Figure 8. Dotted area bounds the elements of transformation. Barrier function represents a hindrance for flow that is inadmissible (undesirable) between elements St1 and St2 of a storage function. In the case of system's damage this undesirable flow may exist in one or another direction (from St1 to St2 or from St2 to St1).

The concept of the goal in the Structural Modelling do not exist. From the viewpoint of model's syntax we can easily depict a goal in the MSM as a specific object with at least one input contact. We name it a goal-object. In general the goal-object can have from 1 to many inputs and from 0 to many outputs (see example in Figure 9).

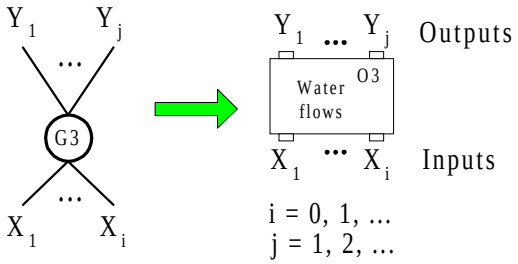
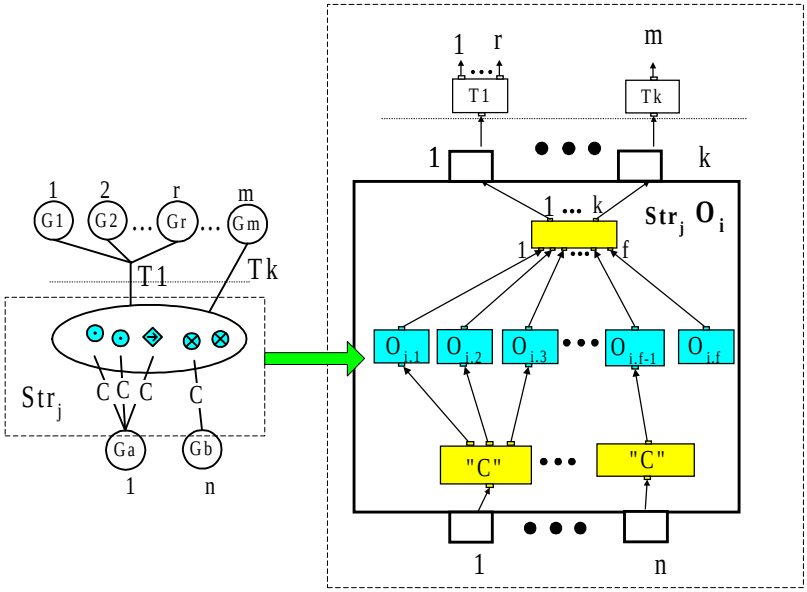


Figure 9. Example of a goal representation as a goal-object O3

During modelling process each goal-object has discrete and binary behaviour. Behaviour is binary because all inputs and outputs deal with logical type of information: "true" or "false" only. Behaviour is discrete because there are only two states: goal is achieved or not. The behaviour is defined by the following expression:

$$Y_1 = Y_2 = ... = Y_j = ... = Y = \bigcap_i X_i$$

In each output (Y_1, Y_2, ...) is the same one value that is equal to the logical conjunction of all inputs X_i. Function objects that are connected to goal-object need special additional inputs and outputs for connections to the goals.



Symbols have the following meaning: m - the number of output goals of functional structure; k - the number of output goal groups of the functional structure; r - the number of goals in the 1st group of goals (r=1,2,3,...); T1 -the label (“A” or “A-C”) of goal relationship for the 1st group; Tk - the label (“A” or “A-C”) of goal relationship for the k-th group; f - the number of functions in the functional structure; n - the number of input goals of functional structure; i - the number of objects; j - the number of functional structures in the model; G1,…,Gm - the output goals of the functional structure; Ga, Gb - the input goals of the functional structure.

Figure 10. General example of functional structure transformation

More difficult cases occur when so called nested transformation is needed. It means that one construction is transformed into another construction and afterwards there is necessity to apply element to element transformation inside the construction. The nested transformation of the functional structure of MFM to the set of connected objects of the MSM is depicted in Figure 10.

6. Transformation Algorithms

The first step of the transformation methods' development is the linguistic description of the transformation algorithm. It represents the strategy and steps of transformation of model constructions and elements at certain level of details. A linguistic descriptions of two different transformation algorithms are given in Table 1 and Table 2. More detailed description is used for practical needs. The input of the transformation process is a model of Multilevel Flow Modelling. Functional structures and functions are changed into similar elements of Structural Modelling in accordance with defined similarity of model elements. The output of transformation process is the MSM of Structural Modelling.

Table 1. The linguistic description of 1 ST transformation algorithm.

Step	Action
1.	***MFM model changing:***
1.1.	Create joining knot points with label "A" or "A-C" for grouped (by similar type) functional structure output goals replacement.
1.2.	Create joining knot points with label "A" or "A-C" for grouped (by similar type) functional structure input goals replacement.
1.3.	Create knot points with label "A" or "A-C" for each independent (not grouped) goal
1.4.	Place the knot points of functional structure output goals outside the structures.
1.5.	Place the knot points of functional structure input goals inside the structures.
1.6.	Assign dependency-direction for all flow connections in the model.
1.7.	Determine the number of functional structures - j.
1.8.	***For each functional structure S_j do the following:***
1.8.1.	Determine the number of output goal groups of structure - k.
1.8.2.	Determine the number of output goals of the structure - m.
1.8.3.	Determine the number of input goals of the structure - n.
1.8.4.	Determine the number of functions within the structure - f.
1.8.5.	Replace ellipse of functional structure with the functional structure object with k output and n input contacts.
1.8.6.	Reconnect the flow nodes of knot points of input goals to their corresponding input goals (not directly but via free input contacts of functional structure object).
1.8.7.	Connect all knot points of functional structure output goals to the output contacts of functional structure object
1.8.8.	Create one special object of output goals of functional structure inside the structure. It will have f input and k output contacts.
1.8.9.	Connect each output contact of output goal object to the output contact of functional structure object.
1.8.10.	***Within the functional structure object S_j for each function F_f do the following:***
1.8.10.1.	Determine the number of input goals that are connected with conditional relationship to this function - c.
1.8.10.2.	Replace function with a similar object keeping (or restoring) connections with other functions or objects of functional structure object. (Reminder: a barrier function should be replaced by two objects!)
1.8.10.3.	If replaced (in previous step) function had a "R" type relationship with component then assign the name of the component as the name of the object. If the connected component has left only this one "R" type relationship function-component then eliminate the component. Eliminate the "R" type relationship of the replaced function.
1.8.10.4.	Supplement the object of function with c control inputs. Each control input connect to the corresponding knot point of functional structure input goal.
1.8.10.5.	Supplement a function with one output contact of functional state of the object.
1.8.10.6.	Connect the functional state output contact to any still free input contact of the output goal object of functional structure object.
1.9.	Replace each knot point of a goal with additional goal object assigning the name of the knot point ("C", "A" or "A-C") as the name of the goal object.
1.10.	Replace each goal of the model with a goal object assigning the label of the goal as the name of the goal object.
2.	***Simplification of MSM model (it is not obligatory):***
2.1.	If it is necessary exclude all abstract transport objects replacing them by connecting flows.

Table 1. The linguistic description of 1ST transformation algorithm (continuing).

Step	Action
2.2.	If it is necessary exclude all additional goal objects outside the functional structure objects that has only one input and only one output.
2.3.	Number the objects.

Table 2. The linguistic description of 2ND transformation algorithm.

Step	**Actions**
1.	***Multilevel Flow Modelling model changing:***
1.1.	Delete all relationships of goals
1.2.	Delete all goals
1.3.	Attach labels of the functional structure to each flow inside the functional structure
1.4.	Delete functional structures
1.5.	Assign directions to flows
1.6.	Transform each function in accordance with similarity
1.7.	If replaced (in previous step) function had a "R" type relationship with component then the name of component assign as the name of the object. If the connected component has left only this one "R" type relationship function-component then eliminate the component. Eliminate the "R" type relationship of the replaced function.
2.	***Morphological Structure model simplification:***
2.1.	Combine repeated objects
2.2.	If it is necessary exclude all abstract transport objects replacing them by connecting flows.
2.3.	Number the objects

The first algorithm includes goal transformation and multilevel transformation features while the second - transforms multilevel structure of the MFM model to the one level structure of the MSM. The first algorithm requires the extension of the MSM that is achieved by adding the concept "goal". When using the second algorithm no the changes of structural modelling are needed but the loss of knowledge about goals and multilevel constructions take place.

Due to the scope of this paper the examples of model transformations are not included.

7. Conclusion

In this paper model transformations supporting knowledge base integration within the structural modelling framework have been proposed. Model transformations allow to integrate knowledge bases from different applications and modelling techniques. As an example illustrating the whole approach, we have described one case, namely, transformations of models of the multilevel flow modelling (MFM) to the model of morphological structure (MSM) of structural modelling. Model transformations from the structural modelling techniques to the MFM are under the development. It is worth to point out that if we have a model of the morphological structure then using ASMOS tools we can generate both kinds of functional models and to built the deep knowledge rule base semi-automatically. The result is a knowledge base that supports structural, behavioural and causal (diagnostic and predictive) reasoning [5].

Future work is connected with investigations of models from different classes and with looking for their similarities that will allow to expand the approach to other classes of models and to work out corresponding model transformation methods. The similarity aspects of models' semantics, i.e., the investigation of permissible level of knowledge losses during the model transformations will be a topic of future research, too.

8. References

[1] Kurki, M. Model-Based Fault Diagnosis for Mechatronic Systems. *PhD. Thesis,* Technical Research Centre of Finland, VTT Publications 223, Espoo, 1995.

[2] Abu-Hanna, A. Multiple Domain Models in Diagnostic Reasoning. *Ph.D. Thesis.* University of Amsterdam, Amsterdam, 1994.

[3] Lind, M. Representing Goals and Functions of Complex Systems - An Introduction to Multilevel Flow Modelling. Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, 1990.

[4] Grundspenkis, J. The Synthesis and Analysis of Structure in Computer Aided Design. *Computer Applications in Production and Engineering: Proceedings of the First International Conference.* Amsterdam, 1983, pp.301-316.

[5] Grundspenkis, J. Reasoning Supported by Structural Modelling. *Lecture Notes of the Nordic-Baltic Summer School'98: Intelligent Design, Intelligent Manufacturing and Intelligent Management.* Kaunas University of Technology Press "Technologija", Kaunas, 1999, pp.57-100.

[6] Jaako, J. The Extension of Multilevel Flow Modelling. *PhD Thesis.* Department of Process Engineering, University of Oulu, Oulu, 1996.

[7] Fang, M. MFM Model Based Diagnosis and Implementation. *Report 94-D-712,* Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, 1994.

[8] Grundspenkis, J. Structural Modelling of Complex Technical Systems in Conditions of Incomplete Information: a Review. *Modern Aspects of Management Science,* No 1, Riga Technical University, Riga, 1997, pp.111-135.

[9] Lind, M. Status and Challenges of Intelligent Plant Control. *Proceedings of 4th IFAC Symposium on Dynamics and Control of Chemical Reactors,* Helsingor, 1995.

[10] Grundspenkis, J. The Extension of Structural Modelling Approach for Procedural Knowledge Representation. *Data Bases and Information Systems:* P*roceedings of the Third International Baltic Workshop,* Vol. 1, Latvian Academy Library, Riga, 1998, pp.152-166.