Int. J. Human–Computer Studies (1994) **41**, 585–617

Tasks and ontologies in engineering modelling

JAN TOP AND HANS AKKERMANS

Netherlands Energy Research Foundation ECN, P.O. Box 1, NL-1755 ZG Petten (NH), The Netherlands and University of Twente, Information Systems Department, P.O. Box 217, NL-7500 AE Enschede, The Netherlands

(Received 15 March 1994 and accepted in revised form 31 August 1994)

Constructing models of physical systems is a recurring activity in engineering problem solving. This paper presents a generic knowledge-level analysis of the task of engineering modelling. Starting from the premise that modelling is a design-like activity, it proposes the Specify-Construct-Assess (SCA) problem-solving method for decomposition of the modelling task. A second structuring principle is found in the distinction between and separation of different ontological viewpoints. Here, we introduce three engineering ontologies that have their own specific roles and methods in the modelling task: functional components, physical processes, mathematical constraints. The combination of the proposed task and ontology decompositions leads to a particular approach to modelling that we call *evolutionary modelling*. This approach is supported by a knowledge-based system called QuBA. The implications of evolutionary modelling for structuring the modelling process, the content of produced models, as well as for the organization of reusable model fragment libraries are discussed.

1. Introduction

Constructing models of physical systems is an important aspect of problem-solving behaviour in engineering disciplines. A model-based approach to tasks like design, diagnosis and prediction is called for in order to justify and enhance *heuristic* knowledge that links observed structure and required, faulty or expected behaviour. A model is an artificial structure that is employed to investigate that behaviour. This artificial structure is based on purposeful abstractions of the observed system, taking into account only those features that are relevant with respect to the task at hand. For example, in design tasks a model can be constructed for testing several design alternatives before actually constructing a prototype; in diagnosis, faulty behaviour can be explained by setting up a model that explains given observations; in general, predictions can be made by instantiating models for specific situations.

Thus, modelling is a performance task that occurs in or even underlies many different engineering tasks. Currently, however, we are still far from a stable theory in AI of the generic knowledge structures that give us an understanding of what modelling actually "is". This applies in particular to the synthetic aspects of model generation and of making modelling decisions, rather than to considering model selection from an already given model fragments library, as is the focus in most current AI work. Yet, the question is: what generic types and structures of knowledge underlie the construction of such model fragments and libraries?

1071-5819/94/010585 + 33$08.00/0

Developing such a theory is highly important from a practical point of view, since it provides the foundation for knowledge-based support systems, facilitating, improving and speeding up Computer-Aided Engineering.

This paper takes the position that a model-based approach to modelling is needed, rather than a heuristic one. To this end, it is illuminating to view modelling from a knowledge acquisition perspective with respect to the problem-solving behaviour of expert physicists and engineers. We investigate the generic knowledge structures in engineering modelling, whereby we focus on the modelling of dynamic physical systems. Both the process and the product aspect of physical modelling are considered. With respect to the former, we will discuss a generic task decomposition and associated problem-solving methods (PSMs) that help structure the modelling process. With respect to the latter, we will explain the importance of introducing and separating different ontological viewpoints. This provides a structure for the specification of model fragments to be used in libraries as well as imposing further structure on the modelling process itself.

In Section 2 we argue that modelling is to be viewed as a form of design task. This has immediate consequences for the generic task decomposition of modelling. Specifically, we propose the specify-construct-assess method as a problem-solving method for the task of modelling. Next, in Section 3 we show that different ontological viewpoints are to be taken into account and, more strongly, are to be separated in modelling and engineering of physical systems. The task and ontology decompositions proposed by us lead to a modelling approach that we call *evolutionary modelling*, and this is discussed in Section 4. We also briefly consider the implications of this view for reusable model fragment libraries. In Section 5 we discuss some pragmatic consequences of *ontological commitment*. Finally, in Section 6 we present an overview of the knowledge-based modelling support system QuBA, that implements the evolutionary modelling approach.

2. Modelling as a design-like task: the SCA method

Our basic hypothesis is that modelling is a form of design. We are led to this position by the analysis of existing modelling applications, and by reflecting on our own experience with modelling and that of others. In this section we will characterize our view of modelling and design in general and present an associated PSM. Our description is in line with the common view that design is an iterative process (Chandrasekaran, 1988; Coyne, 1990; Gero, 1990; Maher, 1990), gradually refining the description of the artifact to be developed.

However, we assert that two important elements of the task are often neglected due to the predisposition for a *computational approach*. First, we note that modelling and design are *learning processes*. The incremental specification of requirements is an essential ingredient of these tasks. Simply assuming an *a priori* fixed set of requirements is too much of a simplification. Second, we argue that modelling and design employ multiple points of view. Moving between different *ontological* viewpoints should not be confounded with performing basic design actions. Whereas the viewpoints reflect the *product* side of design, the design actions refer to the design *process*.

FIGURE 1. The generic design task.

2.1. DESIGN OF MODELS

The general design task is presented in Figure 1, in an almost trivial way. The input to the task is the *intention* (or *goal*) of the designer. We consider this as an external source of information about the actual purpose of the design process. The intention is *not* simply a set of requirements. The point is that intentions are implicit, whereas the requirements are explicit, that is, expressed in some (more or less formal) way. The step from implicit intuitions to explicit requirements is an essential part of the design task [cf. Pahl & Beitz (1988), who make it into a separate phase of design called "clarification task"]. The output of the task is the *artifact description*, rather than the artifact itself, e.g. an engineering drawing.

When the intention of the designer is made explicit, two aspects are relevant:

- The ideal functionality of the artifact, usually measured as *system performance.*
- The pragmatic context in which the artifact is going to be built and used.

The design task is to find a structure that provides an optimal solution for the set of explicit, abstract requirements emanating from these objectives.

As in design, the output of the modelling task is an artifact description (conceptualization). However, modelling differs from general design with respect to the input of the task. In the modelling task, the intention is to design an artifact that allows one to answer certain questions about an observed or observable (imaginary) system through observation of this artifact. Therefore, the artifact must have some properties (i.e. attributes with values or behaviours) in common with the observed system. In modelling, one type of ideal functionality or "performance" is pursued: *similarity* with respect to the observed system. However, unless the model is the system itself it will carry many independent properties of its own, inhibiting perfect similarity. The modelling problem is to construct the most cost-effective artifact that allows an adequate answer to a given query. Limited similarity is pragmatically adequate precisely because not all aspects are relevant for the problem at hand.

Given our view of modelling as a form of design (Figure 1), we can now specify the inputs to this task as indicated in Figure 2. First, the *observations* are a source of

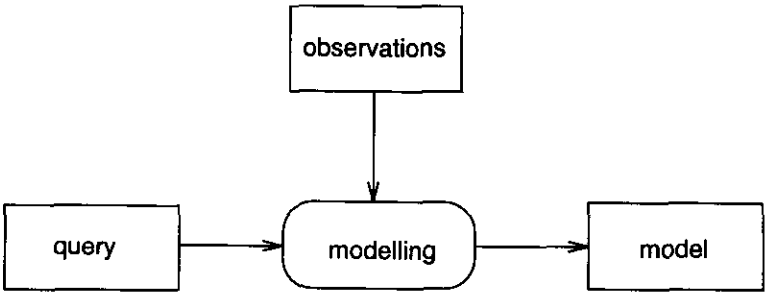

FIGURE 2. The generic modelling task.

information about the behaviour and structure of the observed system. Second, the *query* species which information is to be obtained from the model and the conditions under which this will be done.

2.2. TASK DECOMPOSITION PSM

If we consider modelling to be a form of design, we can apply task decompositions that are commonly used for design tasks in order to decompose the modelling task. This will be the subject of this subsection.

A typical decomposition of the design task is given in Maher (1990). Here, Maher makes a distinction between *formulation, synthesis* and *evaluation* as subtasks of design. The first subtask involves the specification of requirements, the second, generation of design alternatives and identification of additional constraints, while the third subtask assesses the generated alternatives. Many similar decompositions have been suggested, often in conjunction with specific design domains. Here we will derive precisely this decomposition on very general grounds. We only assume that the artifact (here, a model) to be designed is viewed as a *system*, consisting of a limited set of components that collectively determine the properties of the artifact. Such a systems approach is quite standard in engineering.

Many engineering design tasks are based on composition. This implies that an artifact is assembled or thought to be assembled from individual, preferably *generic*, components. Generic components provide a way to introduce practical domain experience into the design process because they have a number of properties that have proved to be useful in certain well-defined contexts. They can be reused in different systems and can be supplied "off-the-shelf". Generic components can also be classified in terms of their abstract properties, and ordered in part-of-hierarchies. Proper decomposition and classification are important aspects of the design task, since they allow the designer to express his or her intention in terms of the generic components, thus reducing the open-endedness of the design problem. If the components are not supposed to be modified internally, the design task boils down to *configuration.*

Now, taking the systems perspective, we know that some characteristics of the system are determined by the interactions between the components (the structure of the artifact) and so they can only be observed on the (partially) completed artifact. This means that the design task is essentially separated into the subtasks *construct* and *assess.* Moreover, the artifact can only be assessed in terms of a set of explicit requirements, which are specified by the designer. Hence, the design process is to be decomposed into:

- **Specification.** The designer will have a number of *a priori* requirements, and additional requirements are generated during the design process. Moreover, some requirements may be switched off at a later stage because they appear to be in conflict with more important ones. The specification task—for which the designer is responsible—gets input from the (partial) artifact and the violated requirements, in addition to the intentions of the designer.
- **Construction.** The construction task entails the actual selection, assembly and

modification of components. In some cases the *violated requirements* can immediately affect the construction task. This is important in the case of automated design, since it implies automatic feedback of violated requirements.

- **Assessment.** Assessment of the artifact under construction is only possible relative to the explicit requirements, possibly resulting in a set of violated requirements.

There is one important difference between the approach presented here and other approaches aiming at automation of modelling or design tasks. Rather than avoiding dealing with *specification*, we consider it to be an essential element of the task. Usually, the specifications are assumed to be supplied *a priori*, followed by an iteration of a generate-and-test (Chandrasekaran, 1988; Coyne, 1990; Falkenhainer & Forbus, 1991) or propose-and-revise cycle (Marcus & McDermott, 1989). However, this ignores the fact that a central aspect of modelling and design is "asking the right question". This simplification is of course called for by the elusive character of problem formulation in general. We believe, however, that a decomposition along the above lines effectively helps to clarify the role of this subtask in automated modelling systems.

Figure 3 shows the inference structure [along the lines of the KADS methodology (Wielinga, Schreiber & Breuker, 1992; Akkermans, van Harmelen, Schreiber &

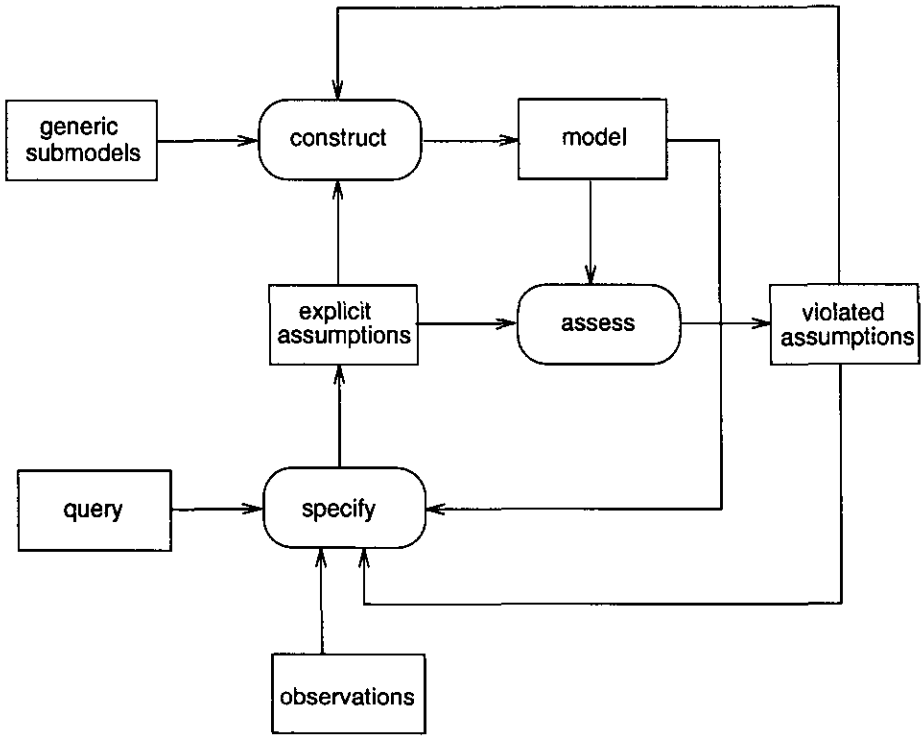

FIGURE 3. Decomposition of component-based modelling.

Wielinga, 1993; Schreiber, Wielinga & Breuker, 1993; Wielinga, Van de Velde, Schreiber & Akkermans, 1993)] that expresses the subtasks of the modelling task and their input–output interrelationships according to our proposed Specify-Construct-Assess (SCA) problem-solving method. Further decomposition is possible: for example, component-based construction may be split up into *select/reject, instantiate* and *assemble.* We will not further discuss this here, and refer the reader to Top and Akkermans (1993, 1994). We also note that control is not indicated, as it will depend on the specific circumstances in which this generic inference structure is applied. Modelling will in general be an iteration over the three subtasks *including* the specification subtask. Task control is discussed in Section 4.

2.3. DESIGN REQUIREMENTS AND MODELLING ASSUMPTIONS

As we stated before, the ultimate requirement in the design task called *modelling* is that the artifact is to display properties of the "real" system, in such a way that a given query can be answered. The fact that not the system itself but the modelling artifact is queried, implies that modelling *assumptions* have to be made. These assumptions serve as the actual requirements for the construction of the model. Note that there is a subtle difference here between the notions "assumption" and "requirement", in the sense that they refer to different perspectives. As a requirement, a statement expresses the need for pragmatic constructibility of the model, whereas the same statement as an assumption refers to an observational constraint. For example, saying that *linearity* is assumed means that we require the mathematical model to have a certain structural characteristic. However, at the same time it means that we restrict our view of the "real" process in such a way that a linear approach is acceptable. Modellers often do not explicitly substantiate this limitation of scope. A control engineer will often assume linearity without explicitly stating in which way this limits the operating range of the model. Or, in another case, a modeller can decide to neglect thermal effects in a device for computational reasons, without saying that as a consequence certain temperatures predicted by the model are not reliable.

The assumptions in principle express the range over which the model can be considered to be *valid,* that is, known to correspond to observations. However, if a model is only reliable within the range of its validation it is useless, since it only repeats what was already known. The use of models lies in their extrapolation from the validation range—if validation is possible at all. The most rudimentary way of generalizing a model is to extend the ranges of the observable attributes beyond a single quantitative value. Further generalization implies underspecifying the relations between these attributes. This is what basic qualitative simulation (Weld & de Kleer, 1990) is all about. However, other extrapolations include considering other variables than those involved in the actual validation, or other mechanisms or even complete devices. The need for complete validation reduces if the model structure is assessed in terms of general, explicit knowledge for the domain and type of problem considered. For example, we know that most mechanical springs will react according to Hooke's law for small excursions. This knowledge can be used to support the use of a linear model, that was necessitated for computational reasons.

In the following section we will show that a further analysis of the set of modelling assumptions must be set in the context of *ontological differentiation.* Different perspectives imply different modelling assumptions, but at the same time these different assumptions must be mutually consistent.

3. Multiple engineering ontologies†

The previous section has discussed the modelling process. In this section we will focus on the *content* of engineering models. In order to find general handles for structuring models we observe that during the modelling process engineering systems are considered from various different ontological angles. This observation is reflected in the Qualitative Physics area where we find the distinction between the device, process and constraint approaches (Bobrow, 1984; Weld & de Kleer, 1990). However, rather than taking one of these as a dominant view, we consider them equally important. Also, we do not believe it appropriate to simply mix them up as is done in automated modelling approaches. The point is that they should be separated in order to be able to maintain the underlying assumptions, while at the same time their mutual consistency should be guaranteed for describing full-fledged physical system models. We employ the separation of ontological viewpoints as a major *structuring principle* for storing engineering models and model fragment libraries.

Below we describe in some detail ontologies of importance to the modelling of dynamic physical systems. For each ontological viewpoint we will describe the type of information contained by a model at that level, but also provide potential assertions *about* this content. Although in principle both types of information are elements of the complete set of modelling assumptions, usually one begins by making meta-assumptions. They can range from very general (e.g. "*exclude quantum-mechanical effects*") to rather specific (e.g. "*neglect friction in this part of the system*"). These meta-assumptions serve as initial requirements for which an adequate artifact (the model) has to be found. Note, however, that experienced modellers tend to express their initial assumptions as much as possible in terms of the model structure, in particular if a dedicated modelling language is available.

3.1. FUNCTIONAL COMPONENTS

The first observation to be made when attempting to draw conceptual borderlines between different types of engineering knowledge is the fact that *device components* play a central role. A major activity in engineering is to design and develop generic physical components with more or less ideal functionality, such that the overall design task given a specific problem can be reduced to a configuration task, if

† A better term is "ontography", since we want to refer to different aspects of description without implying the connotation of existence claims that usually go with the notion of ontology. We owe the term *ontography* to Paul van der Vet (University of Twente).

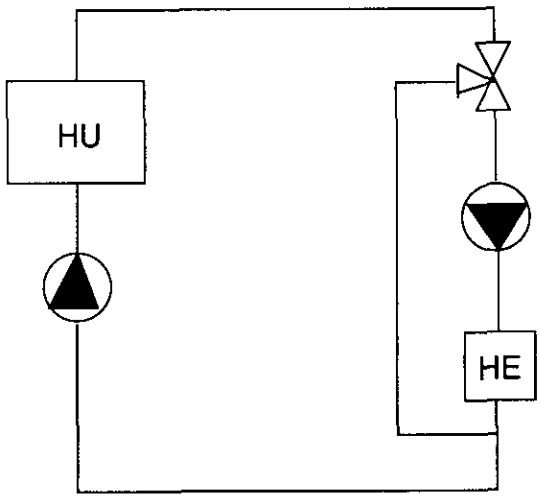

FIGURE 4. Engineering drawing of simple fluid flow system, with two pumps, heating unit (HU), heat exchanging element (HE), three-way valve and connecting pipes.

necessary supplemented with some local tuning. It is clear that these generic components will play an important role in automated modelling.

An *engineering drawing* (see e.g. Figure 4) is typically used for configuration of a system out of device components. On the other hand, the drawing is also interpreted as a representation of more or less ideal functions or processes: an initial abstraction step is implied. Do the connecting lines in the figure stand for pipes with a certain length, mass, unique number, etc., do they represent a class of pipes with a certain shape, or do they denote ideal, frictionless connections? It depends on the modeller how the engineering drawing is going to be interpreted. Hence, the engineering drawing as such is not an unambiguous representation.

Nevertheless, we assert that, in practice, components are basic starting points for modelling, precisely because components are designed to perform well-defined functions. However, we will have to use a representation in which no tacit assumptions are introduced. Therefore, we define a first abstraction level in terms of *functional components,* which are subsystems expressing two distinct aspects of the observed system:

1. The interface between a subsystem and its environment. As far as its dynamic behaviour is concerned, a component is completely defined by the set of *energy ports* through which it exchanges energy with the environment. Energy ports can occur in degenerate form, only passing through simple signals. The complete interface could be referred to as the *energetic extension* of a subsystem. Functional components define the system decomposition and connectivity.
2. A label to indicate a class of engineering functions. Since devices with common functions are usually based on similar physical processes, they provide additional organization for the underlying physical process level. In automated

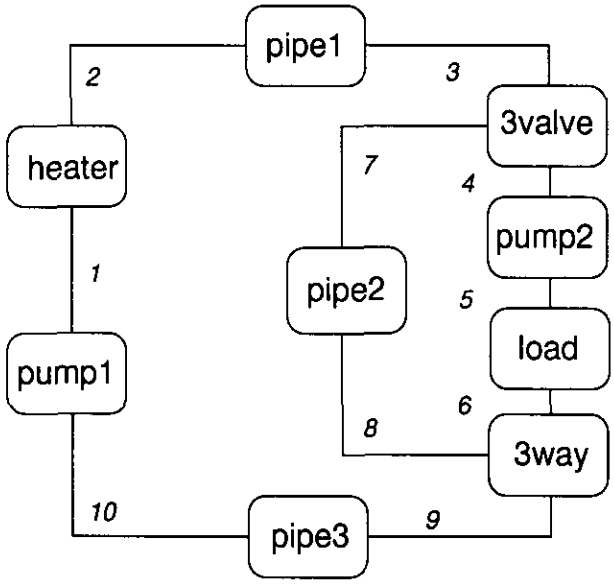

FIGURE 5. Functional component model of the fluid flow system.

modelling the labels provide an initial selection mechanism for process submodels.

One way to represent this initial abstraction of the observed system is shown in Figure 5 for a fluid flow system. The nodes of this functional component diagram map to functional components, whereas the edges represent power or signal flow between ports. Thus, the interconnections in the component model shown in Figure 5 have a well-defined meaning: they refer to specific measurement points for pressure and fluid flow. In some domains (e.g. electrical, hydraulic) the functional component graph maps quite nicely to the engineering drawing, but in other domains (e.g. mechanics) this is not necessarily so. Therefore, we view the engineering drawing as an initial *observation* of the system, being still open for multiple interpretations, whereas the functional component graph explicitly indicates the "reticulation", i.e. network structure, assumptions made.

Typical meta-assumptions at this level are:

- The system can be closed or open, i.e. ports can be unconnected.
- The model may include a given set of physical subdomains.
- The components can be modelled as independent entities.

3.2. PHYSICAL PROCESSES

The second level of abstraction describes the actual physical processes that underlie model behaviour—or rather classes of model behaviour. A formal way used in engineering to describe these processes is called the *bond graph* (Paynter, 1961; Karnopp, Margolis & Rosenberg, 1990; Top & Akkermans, 1991). For the fluid

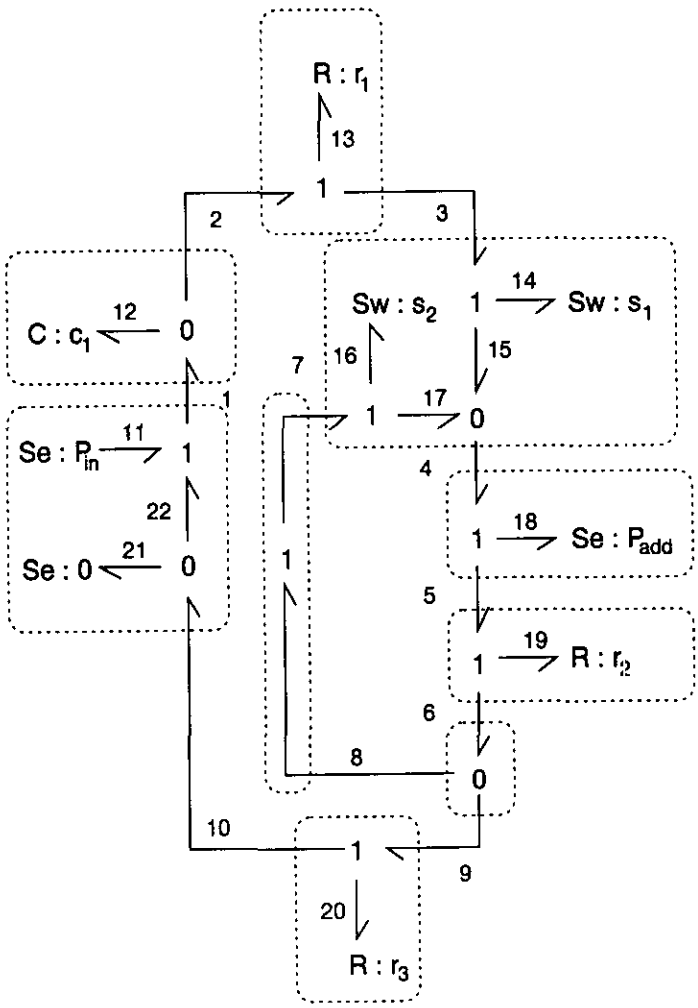

FIGURE 6. Process level description of the fluid flow system. All bonds are in the hydraulic domain, with *total pressure* as effort and *fluid flow rate* as flow. The dashed boxes refer to the functional components.

flow system an example is given in Figure 6. The basic difference between the component view and the process view is the fact that the latter represents abstract *processes* or *mechanisms* rather than devices. Bond graph nodes stand for generic physical mechanisms, such as power sources (Se, Sf, Sw), dissipation (R), energy storage mechanisms (C, I), energy conversion (TF, GY), and energy distribution mechanisms based on conservation principles (0, 1). Edges (called bonds, hence the name) stand for energy exchange paths (implying a complementary pair of shared physical quantities between mechanisms, called effort and flow). As a whole, a bond graph can be viewed as representing the topology of the physical mechanisms active in the system. Within a functional component each bond graph is a flat structure; the hierarchical structure of a bond graph model is organized at the component level.

At this level, a number of model properties can serve as meta-assumptions. By employing the bond graph as a descriptive language one already assumes a macro-physical approach. Moreover, it is assumed that the system can be viewed as a network of internally homogeneous lumps, although these lumps can be very small. The bond graph itself can be characterized by the following properties:

- Causal characteristics such as: consistency, locations and types of feedback paths, presence of static coupling (in particular rigid links in mechanical systems), dynamic system order.
- Being closed or open with respect to energy.
- Existence of a steady state.
- Complexity.

3.3. MATHEMATICAL CONSTRAINTS

The third ontology in describing engineering models is that of *information* rather than energetic links: it describes the mathematical structure of the model. This is typically in the form of a set of (ordinary) differential and algebraic equations. The physical and mathematical representation are frequently fused, but differentiation between these levels is certainly sensible, both from a theoretical and from a practical point of view. They deal with essentially different types of knowledge and they carry with them different methods (e.g. causal analysis vs. symbolic algebra). A bond graph represents *physical* mechanisms, which only define the *global form* of the mathematical relations that describe the actual phenomena.

As in the case of the other two ontologies, the mathematical level can be viewed as a network, where the edges are variables and the nodes are relations between variables. This is reflected by the *block diagram*, which is commonly used in control engineering. In Figure 7 we present an example of how the mathematical level looks in textual form.

The block diagram and the associated equations make up the actual contents of the model. These contents can be characterized by the following meta-assumptions:

- (Non-)linearity.
- Order of the set of differential equations.
- Form of the set of differential equations (explicit or implicit state-space, n-th-order differential equation, etc.).
- State-space characteristics (periodic behaviour, chaos, stable points).
- Solvability.

In Section 2.3 we mentioned that apart from computationally oriented properties, also the validity range of the model should be made explicit. This is often omitted, thus assuming that for a given model any query is allowed. However, since the model can only display observed behaviour for a number of quantities over a certain range and with a certain accuracy, the mathematical model must be qualified by additional meta-assumptions:

	$P_{11} = P_{in}$
	$\phi_{22} = \phi_{11} = \phi_1$
	$P_1 = P_{22} + P_{11}$
	$P_{12} = c_1^{-1} V_1$
	$V_1 = V_1(0) + \int \phi_{12} dt$
	$P_1 = P_{12} = P_2$
	$\phi_1 = \phi_{12} + \phi_2$
	$P_{13} = r_1 \phi_{13}$
	$\phi_2 = \phi_{13} = \phi_3$
	$P_2 = P_{13} + P_3$
	$P_3 = P_{15} + P_{14}$
	$\phi_3 = \phi_{14} = \phi_{15}$
*	s1: $(e_{14} = 0$ if $\alpha > 90\%) \vee (\phi_{14} = 0$ if $\alpha < 10\%)$
*	s2: $(e_{16} = 0$ if $\alpha < 10\%) \vee (\phi_{16} = 0$ if $\alpha > 90\%)$
	$P_7 = P_{16} + P_{17}$
	$\phi_7 = \phi_{16} = \phi_{17}$
	$P_4 = P_{15} = P_{17}$
	$\phi_4 = \phi_{15} + \phi_{17}$
	⋮

FIGURE 7. Partial signal or mathematical description of the fluid flow system. The relations indicated by * have been specified explicitly; the others are default as they are linear.

- **Quantities of interest.** These are the quantities (measurable qualities) through which the model is assumed to mimic the observed system. As a matter of fact, they are the principal entities in the model.
  Quantities referred to in the query are quantities of interest, but may appear to be irrelevant later and other quantities may have to be added. For example, in modelling a gear train the initial idea could be that internal friction is an important factor in determining the driving force. However, during the modelling session it appears that the gears are operated at low velocities and not the internal friction but the mechanical load is relevant.
- **Scopes of quantities of interest.** The model is not required to display all possible values for all quantities of interest. The scope of a quantity restricts the values for which the model is assumed to be valid. In a precisely defined context, scopes will be narrow, possibly a single value.
  On a map, small villages are indicated by dots instead of drawing the actual outline, and one cannot obtain the actual size of these villages from the map. Trying to do so is asking the wrong question for this model, because below a certain minimum the size is not a quantity of interest anymore.
- **Accuracies of quantities.** Each quantity of interest must behave like the observed quantity within a certain margin. This also expresses the experimental accuracy that was obtained when validating the model.

The *scope* of a quantity (parameter or variable) generalizes the notion of "value". Any instantiation of the quantities of interest that complies with the ranges specified here and the associated set of mathematical equations is assumed to be valid. Hence, we do not consider the *initial conditions* of state variables as such to be a part of the model. A state variable may be initialized at any value within its scope. In conventional approaches the information represented at this level is usually restricted to simple parameter values, thus allowing the derivation of the behaviour of the system by simulation. However, by taking this broader view it is possible to increase model reusability, and at the same time to detect improper use. Of course, for a specific device the parameter values can be restricted to a single value by narrowing down the experimental conditions. However, rather than viewing the actual instantiation of the parameters and variables as a basic modelling step, we see it as a form of analysis (validation or prediction). In this way simulation is given its proper position in the overall modelling process.

An example of this part of the information level can be seen in Table 1, containing a number of particular details:

- The flow in the pipe between the pump and the three-way valve is assumed to be laminar. Thus, the associated fluid flow has to be a property of interest which is restricted to a certain maximum value.
- The switching relations define a controlling variable α which must be either greater than 90% or less than 10%. For intermediate values the model is not valid.

In sum, we have distinguished three separate engineering ontologies—functional components, physical processes, mathematical constraints—that together provide the vocabulary for and structure of a dynamic physical model. In addition, each ontological viewpoint carries with it its own characteristic modelling and analysis methods. At the component level one is dealing with structure mapping and system-theoretic analysis; at the process level the central method is that of causal analysis; at the signal level we have the well-known symbolic and numerical calculus methods of mathematics, including simulation.

TABLE 1
Meta-assumptions in the fluid flow system at the information level

Quantity of interest	Scope	Accuracy
P_{in}	$P_{in} = 2 \vee P_{in} = 5$ kPa	±2%
c_1	$c_1 = 10^{-4}$	±2%
q_1	$0.0 < q_1 < 0.5$ m^3	
r_1		
ϕ_{13}	$-0.01 < \phi_{13} < 0.01$ m^3/sec	
α	$\alpha < 10 \vee \alpha > 90\%$	
ϕ_{15}		
⋮		

This is not to say that there are no other ontologies that may be relevant to engineering modelling. This limitation is rather due to our restriction to the energetic-dynamic aspects of physical models. For other aspects other ontologies may also be needed (e.g. in structural mechanics or finite element analyses, or computer-aided drafting, a geometric ontology is central). In actual fact, the mathematical level may be seen as a common basis for multiple ontologies (here in particular related to physical, geometric and material properties of the system). This does not diminish, however, our central point that the distinction and separation of engineering ontologies is crucial to the modelling task.

4. Evolutionary modelling

4.1. EVOLUTIONARY MODELLING: DECOMPOSITION AND CONTROL

The task and ontology decompositions that we have set out in the previous sections *together* lead to a specific style of modelling that we call *evolutionary modelling*. The separation of ontological viewpoints as proposed in Section 3 not only provides structure for the content of models, but also for the modelling *process*. This is depicted in Figure 8. The modelling task can initially be divided into three design subtasks, one for each ontological viewpoint. As described in Section 2 on the SCA problem-solving method, each of these subtasks can subsequently be decomposed into *specification*, *construction* and *assessment*. The modelling task iterates over these activities, moving between the three viewpoints. An important conclusion therefore is that not only task decomposition PSMs but also ontology distinctions are important in structuring the problem-solving task.

Let us now consider the *task control structure* in evolutionary modelling. First we recall the inference structure that was developed in Figure 3 for the present modelling task. Ideally, the modelling process consists of a single specification step, followed by an iteration process that performs a number of generate-and-test cycles

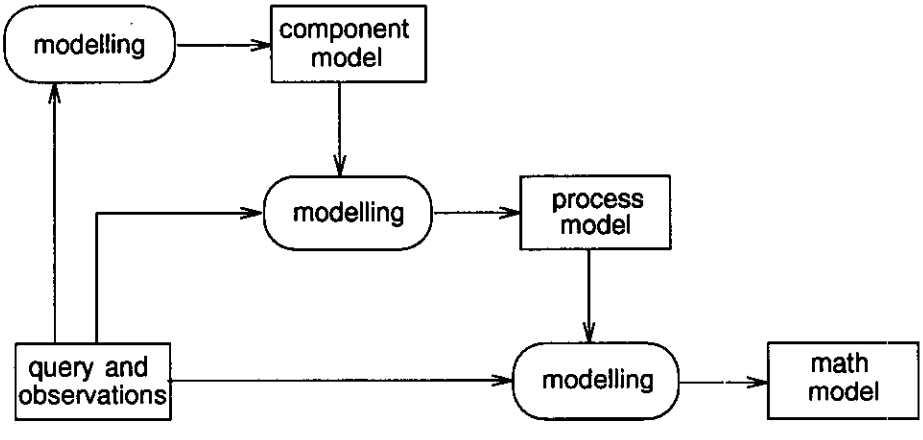

FIGURE 8. The evolutionary modelling approach.

specify (· · · → assumptions)
repeat
    construct (· · · → model)
    assess (· · · → violated assumptions)
until no more violated assumptions

FIGURE 9. Basic control structure for automated modelling without user interaction. The inputs of the inferences are left out for clarity, but can be obtained from the inference diagram (Figure 3).

until the *a priori* assumptions are met and the question can be answered (Figure 9). This is the usual point of view in AI (Chandrasekaran, 1988; Goel & Chandrasekaran, 1989; Falkenhainer & Forbus, 1991) and ultimately leads to automated modelling as a form of *constraint satisfaction.*

However, there are two reasons why this ideal picture is not valid anymore in *evolutionary modelling.* First, since specification is an essential element of the modelling task, there must be a second iteration in which implicit assumptions are made explicit so that they can be revised. This aspect of modelling has a clear parallel in design, where the specification of requirements is a laborious process of an exploratory nature (a point abundantly clear in information and knowledge systems!). This is depicted in Figure 10. The modeller starts by specifying his or her initial assumptions, for example by defining the global system boundary and other meta-assumptions. Given these initial assumptions, a first proposal for the model structure is made, for example by editing a graphical representation. Next, an iteration over construction and assessment is performed. Here the construction task itself performs *local* checking by constraining possible structures, whereas the assessment tasks entails *global* checking of the model. Depending on the outcome (including both the model *and* the violated assumptions) of this autonomous process, the modeller will supply new specifications, with the objective to construct a more specific question.

There is a second issue that complicates things further. We have seen that any non-trivial modelling task involves multiple viewpoints. The above iteration occurs for each of these viewpoints. In an ideal situation, the modelling process would successively pass through the different perspectives in a top-down manner, completing one level at a time. However, modellers are not that well-mannered. They repeatedly change perspective and shift up and down between levels. Therefore, the

repeat
    specify (· · · → assumptions)
    repeat
        construct (· · · → model)
        assess (· · · → violated assumptions)
    until no more violated assumptions
until answer is obtained

FIGURE 10. Control structure for evolutionary modelling, restricted to a single ontological viewpoint.

above control structure will be interrupted from time to time when moving to another level.

Nevertheless, it is still possible to say something about task control in that case. To do so, we have to distinguish between different reasons for shifting perspective. First, in a moment of unforseeable creativity, the modeller may decide to introduce an assumption "out of the blue" at an arbitrary level. This kind of spontaneity cannot and should not be frustrated. However, when the modeller happens to be less assertive, the support system will provide a guideline by proposing new assumptions and assessing old ones. This is precisely the role it should play as a knowledge-based system. What will happen in general is the following. The modeller specifies a model at one ontological level which will be constructed and assessed by the support system. The construction task may introduce new meta-assumptions at another level. The modeller is made aware of this, and can decide to move to the other level. By accepting the new assumptions, the construction-assessment process is invoked at this level. After this, the modeller can decide to move back to the original level or to continue at the second. Hence, task control is partly determined by the consequences that construction at one level has for specification at another level.

If the different levels are properly chosen, the amount of interference between them will be minimized. This will allow the modeller to concentrate on one level at the time. In practice this is usually not strictly possible. However, we believe that the levels we have identified in *evolutionary modelling* provide a suitable differentiation for the modelling task considered, since interference is well defined and restricted to neighbouring levels only. It is beyond the confines of the present paper to demonstrate the evolutionary modelling process outlined here by some specific physical modelling exercises, but the reader can find an extensive application example in Top and Akkermans (1993, 1994).

4.2. MODEL FRAGMENT LIBRARIES

The *evolutionary modelling* approach depends on the availability of a collection of submodels, and it provides the organizational framework required for structuring and maintaining such a collection. Model libraries can be used to share and reuse knowledge that is not part of an abstract theory (such as physics) but still generic across different applications. Explicit storage of models is necessary in order to supply domain-specific but task-independent knowledge, for which no theoretical support is available.

If engineering models are organized around the three mentioned engineering ontologies, the modeller can be supplied with *reasonable alternatives*. The availability of sensible alternatives is important in any design activity because it provides potential answers for the questions summarized as "What if...?" A library restricts the infinite number of possible alternatives to a limited set that has proved to be useful in practice. The library structure should be such that among the alternatives some are indicated to be close neighbours for a given model, that is, they differ only slightly along a single dimension. Two general structuring principles can be used to organize a library. First, there is a *part-of-hierarchy* that is defined by the decomposition of functional components. Every functional component knows about

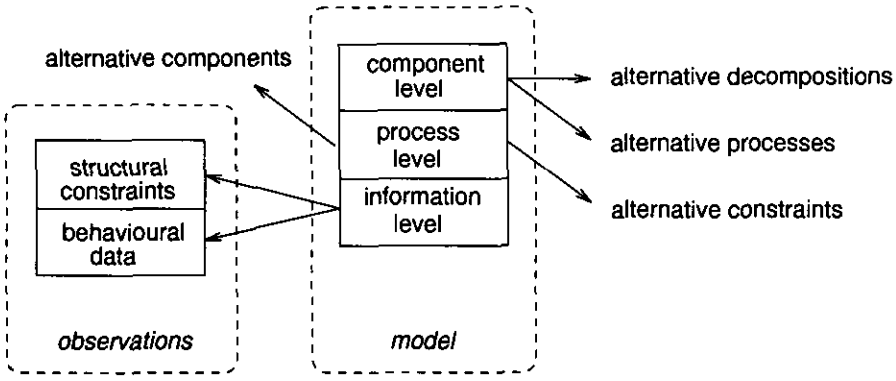

FIGURE 11. Composed model, retrieved from a library based on evolutionary modelling. The arrows denote links between different entries in the library. The information level can refer to observational information in the form of structural (geometry or material) and behavioural constraints (experiments). Note that in practice it is useful to separate between mathematical relations and data. The latter are defined as the quantities of interest with their scopes and accuracies, including both variables and parameters.

one or more decompositions to its immediate subcomponents and about the supercomponents it can be a part of. Second, models are ordered in a *taxonomy* based on invariant properties. For example, in the domain of electrical motors one can have AC- and DC-motors, where AC-motors can be of the one-, two- or three-phase type. More detailed models (that is, models based on more detailed assumptions) can inherit general properties from less detailed ones.

In our library, each of the ontological levels is stored as a separate library entry, as depicted in Figure 11. This figure shows how the three ontologies provide a framework for structuring reusable physical models. This framework is used in practice as a basis for organizing the library developed in the European Esprit-III OLMECO project, which is concerned with reusable models for mechatronic component design.

The *model* part in this figure shows the contents of a composed model for a given situation. A complete model—i.e. a model for a particular device within a certain task environment—is composed by selecting a library component from each level, while keeping the references to possible alternatives. When retrieving a model from the library, indexes of alternative models are obtained simultaneously, suggesting alternative decompositions. Further, alternative implementations can be obtained for the component, process or signal level. Moreover, identical bond graphs may occur in different components, in particular if the domains are not required to be equal. The latter references suggest an alternative occurrence of a given physical process, which can be useful in the context of design. For example, instead of a mechanical damped-spring system, its equivalent in the electrical domain can be considered, provided that it can be interfaced properly.

In addition to the entries for the actual energetic-dynamic model, important information is stored in the *observations* section of the library. The entries in this section contain behavioural and structural descriptions together with the conditions under which they were obtained. Structural constraints are geometrical and material

TABLE 2
Observational data examples for the fluid flow system

Component	Specs	Conditions
⋮		
pump Type AXP301	$P_{in} = 0, 2, 5, 10\,\text{kPa} \pm 2\%$	$\phi_{in} < 0.1\,\text{m}^3/\text{s}$
pressure vessel Type PC87	$c = 10^{-4}\,\text{m}^5/\text{N} \pm 5\%$	
	$0.0\,\text{m}^3 < q_1 < 0.5\,\text{m}^3$	
⋮		
cylindrical pipe	$r = 128\eta l/(\pi d^4)$	$Re = \rho v d/\eta < 2300$
⋮		

specifications, behavioural data are observations of dynamic quantities. For the fluid flow example shown before, we give an example in Table 2.

A model library structured in this way can support the modelling process in a number of ways, depending on the information available *a priori*. The most straightforward use would be to construct application-specific models that simply make a single selection at each level. In that case, a detailed model is obtained, referring to a specific device under specific experimental conditions. Another, more flexible strategy would be to select a number of *potential* implementations, which collectively satisfy a set of global requirements. By explicitly specifying the conditions for transitions between models, the most suitable model can automatically be selected. For example, if the frequency of the signal through an electrical resistor reaches a predefined limit, the model switches from the pure resistor model to the one having parasitic capacity. If the frequency comes below that limit, the simpler model is used. A similar approach was proposed in Addanki, Cremonini and Penberthy (1991), although it was not explicitly restricted to a subset of the complete library, which would probably lead to untractable complexity for reasonably large libraries. Such a *cluster* of models should be viewed as an application-dependent intermediate stage: it is more specific than the generic library models, but more flexible than the simply instantiated models. This approach allows model flexibility to be extended to include changing structures, rather than just changing parameters.

Yet another use of the library occurs when *new* models are being developed, for which no validated instances exist in the library. In that case, the models contained in the library can serve as templates, that indicate example solutions. The described conceptual differentiations allow for a more *qualitative* approach than just comparing simulation results. A model can for example be constructed by retrieving only the component and process levels from the library and filling in the lower levels manually.

4.3. RELATED WORK

Automated modelling of physical systems has become an important research issue in recent years. Conventionally, engineering has viewed modelling as an *art*, and the main focus has been on what could be called *extended simulation support* rather than on automation of the complete trajectory of model development. Important steps with respect to the overall modelling process have been made in AI.

In Falkenhainer and Forbus (1991) the *compositional modelling* approach is described. The strategy followed has a close resemblance to the one followed in evolutionary modelling. Models are composed from model fragments, which collectively form a domain theory. The basic inference steps are as follows. In *scenario expansion* all applicable model fragments are identified on the basis of a scenario description. The latter is comparable to our functional component model, although the semantics of a scenario is less well defined. Then, *query analysis* is performed, leading to the identification of a relevant set of quantities. Using these, the minimal set of *model fragments* is selected. In our terminology, this is equivalent to specifying the quantities of interest plus a number of structural constraints. Next, *candidate completion* is to be carried out since each model fragment can give rise to additional fragments. This is done by means of dynamic constraint satisfaction. In evolutionary modelling this step is equivalent to selecting primary and secondary physical processes and their mathematical implementation. *Candidate evaluation and selection* finally selects the simplest one from all models that apply.

There are a number of differences between this approach and *evolutionary modelling*. First, Falkenhainer and Forbus automatically generate the best model for a given set of requirements, whereas we consider the specification of these requirements as a part of the modelling process. In evolutionary modelling, the objective is to maintain a balance between interactiveness and automation. Rather than simply taking the assumptions for granted and adapting the model until it complies with these assumptions, we allow for revision of the modelling assumptions. Compositional modelling provides a higher degree of automation at the cost of leaving the requirements development and specification task out of sight. Second, in *compositional modelling* no distinction is made between different conceptual levels. As a result, a proper organizing framework is missing. Third, the representation of the underlying physics is on an *ad hoc* basis because model fragments and assumptions are domain and application specific. In evolutionary modelling the bond graph method is employed to supply the required generic formalism.

Addanki *et al.* (1991) have developed the *Graphs of Models* (GoM) approach. The nodes of the graphs are models and the edges are the assumptions that have to be changed in going from one model to another. Models are checked against internal requirements and empirical data. If a conflict is detected, an alternative model can be selected on the basis of the model graph through a goal-directed search. The method is based on the assumption that models and transitions between models are defined *a priori*. The GoM aproach differs from evolutionary modelling in the same aspects as mentioned above with respect to compositional modelling. However, graphs of models could be introduced in evolutionary modelling to represent coherent *clusters* of pre-compiled models, in the context of model libraries. This would extend the applicability ranges of models.

The work of Nayak (1992) and of Gruber (1993) derives from the compositional modelling approach. The focus of Nayak's work is on improving the algorithms for finding adequate models, and it gives a formalization of various notions within compositional modelling. Its ontological outlook is quite strongly mathematical, even where it concerns causal explanations of physical behaviour. Gruber discusses engineering modelling from a knowledge acquisition standpoint. He also puts, as we do, quite some emphasis on the interactivity needed in supporting the modelling

task, although his proposed set of reasoning subtasks does not include what we call the Specification subtask. The remaining part of the task decomposition is, in different terminology, similar to the Construct-and-Assess part of the PSM proposed by us. He furthermore stresses the need for explicit representation of applicability conditions in knowledge-based modelling support. Evolutionary modelling, due to the ontology-based organization of model fragment libraries, leads to a different solution for this problem. For example, structural and behavioural preconditions are not represented as a variety of predicates within a model fragment but, respectively, as the link between the functional component part and the physical process part, and as the scope of the quantities of interest. Several other contributions to the automation of the modelling task are currently being made (see the compilations in Stein, 1991; Falkenhainer & Stein, 1992).

5. Experience

Our approach has a clear potential for advancing computer-aided modelling and design, precisely because it is rooted in conventional engineering. In this section we provide support for this claim by referring to the bond graph paradigm as a basis for representing general engineering knowledge. In the next section an overview of our modelling environment QuBA will be given as a practical implementation of the ideas underlying *evolutionary modelling.*

5.1. BOND GRAPH MODELLING

When inspecting the engineering literature one may get the impression that modelling means transformation from a scenario description to mathematical equations. However, it appears that in fact only a limited set of abstract mechanisms is used when making this transition. For example, *storage* of some quantity occurs in many different circumstances, but maps to a single type of physical process. The generic set of physical mechanisms defines an abstract level between the component description and the mathematical model. However, this type of causal knowledge is usually left implicit and as a consequence many attempts to automate the modelling process are based on *ad hoc* mathematical descriptions for each device for every type of problem.

One exception to this rule of implicitness is found in the bond-graph literature (Breedveld, Rosenberg & Zhou, 1991). Here, the concise graphical form of the graph with its well defined semantics (Paynter, 1961; Breedveld, 1984*b*; Karnopp *et al.*, 1990) has made it possible to explicitly represent the mechanisms that were tacitly used in electrical, mechanical, thermodynamical and other domains. This has significantly contributed to the communication of these models and the explicitness of underlying modelling assumptions, as is shown by the rapidly growing number of applications. Also, the bond graph paradigm apparently fits well into the computation approach to engineering, since it has provoked a suite of computer-based modelling systems. However, most of them are merely viewed as front ends to numerical simulation. In addition to this the use of bond graphs has been an

important step forward in engineering education (Stein & Rosenberg, 1991). Bond graphs enable students to generalize concepts from different subdomains and to understand the physical rationale behind mathematical models.

5.2. NUMERICAL SIMULATION

In particular, the explicit representation of physical causality shows the importance of bond graphs as an intermediate level of abstraction in the physical domain. Although the engineering community tends to confuse computational order of calculation with physical causality (Top & Akkermans, 1991), the bond graph enables one to analyse physical causality independently from the underlying mathematical model. Moreover, the availability of an adequate set of causal mechanisms also contributes to computational efficiency. It appears that mathematical models derived from proper bond graphs will have less computational problems associated with them than those derived in a more implicit way (van Dijk, 1994). This becomes particularly clear in the case of the switching element, one mechanism that was recently introduced (Strömberg, Top & Söderman, 1993) as a natural extension of the bond graph definition. This shows that a proper ontology reduces complexity for the modeller and shifts it to the underlying computational engine as a routine task.

5.3. MODEL LIBRARIES

In this context we should also mention the OLMECO-project that is taking place in the ESPRIT-programme of the European Commission. The idea is to develop reusable and sharable models based on the bond graph language, in this case dedicated to mechatronic systems. From the industrial side there is a growing awareness that such a common knowledge base will significantly improve both quality and efficiency of the engineering process. Recently an experiment was run to simulate the expected use of the library. It appeared that people tend to follow the proposals made by the library, in particular if the problem is new to them. Initially, a set of components is searched for in order to comply with the given scenario. Next, these components are either further decomposed or implemented by bond graphs. From then on, the modellers start switching between different levels. One of the most important conclusions of this experiment was that the feedback information supplied by the computer support system is crucial at every step of the modelling process. *Ontological commitment* will enable the system to provide all relevant information.

6. QuBA: Qualitative Bond-graph Analysis

In this chapter we describe the functionality of a system called QuBA, short for Qualitative Bond-graph Analysis. In contrast to the majority of modelling support systems, it does not claim to provide a full-fledged stand-alone environment for simulation and analysis of mathematical models, as does, for example, CAMAS (Broenink, 1990; Broenink, Bekkink & Breedveld, 1992). Instead, the idea behind QuBA is to create an environment for exploring the complete modelling trajectory,

emphasizing the initial *conceptualization* steps. This is not unlike the objectives behind MAX (van Dijk, de Vries, Breunese & Breedveld, 1992; de Vries, 1994), which supports modelling in terms of ideal physical elements. However, QuBA is based on the evolutionary modelling approach described in the previous sections, and accordingly allows the description of models at the *component, process* and *signal* levels.

In AI a number of systems have been developed to perform automated modelling (Falkenhainer & Forbus, 1991; Addanki *et al.*, 1991; Nayak, 1992). As we have pointed out, these implementations assume that the model specification phase has been completed. The task of the programme is to generate an adequate model given a fixed set of *a priori* assumptions. However, we also showed that this ignores the fact that incremental specification of assumptions is an intrinsic part of the modelling task, and should therefore be supported. In QuBA, each level has its own capabilities for all three subtasks: *specification, construction* and *assessment.* Moreover, since QuBA builds on the domain knowledge embodied by bond graphs, it has a much stronger theoretical basis for modelling physical systems.

The differentiation between the three conceptual levels is the most prominent feature of the system (see Figure 12). The three levels represent different ontological perspectives on a single model, each of them represented in a graphical way. This facilitates a strict separation between different types of knowledge and methods of analysis. In addition to the three graphical views, a separate view is reserved for messages resulting from the assessment process: warnings are given when the user attempts to perform illegal actions or when global assumptions are violated.

6.1. COMPONENT LEVEL

At the level of *functional components* the initial reticulation takes place, defining the energetic extension of devices. In QuBA, this is represented in the form of a *hierarchical* decomposition rather than an engineering drawing, in order not to introduce tacit assumptions and interpretations.

At this level, new component models can be defined or existing ones can be entered from a library. Presently the library is of limited size, but considerable effort

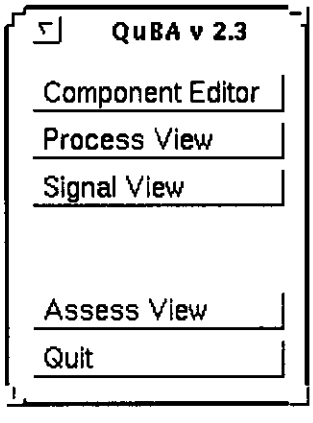

FIGURE 12. The main menu of QuBA allows the user to open and close each of the three views and a separate *assessment view*, and to close the system.

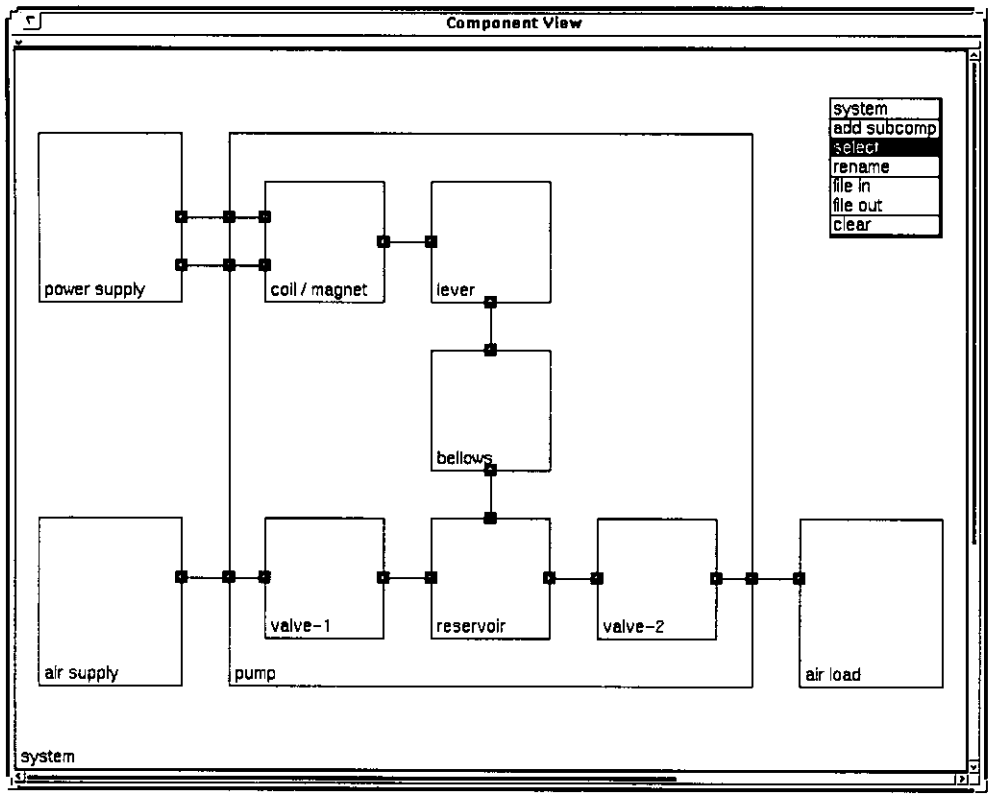

FIGURE 13. The component view, showing a two-level decomposition for an airpump. A subcomponent is drawn inside the area defined by its supercomponent.

is being made in the ESPRIT project 6521 OLMECO to construct a set of reusable and sharable models. Component models are organized in a part-of-hierarchy, which is displayed at all levels of decomposition simultaneously. Components are mutually connected via component-level connections that define links between ports. When the modeller decides that further decomposition is not required, an implementation at the bond graph level can be selected for each leaf of the decomposition tree. Figure 13 shows an example of the functional component window.

6.2. PROCESS LEVEL

At this level, the primitive bond graph elements are the basic building blocks. A bond graph can be entered as a complete submodel that fits into the interface of the associated component, but it can also be entered manually. When in edit mode, the user can select an element type by activating one of the buttons below the graphical field, as shown in Figure 14. In QuBA, the constitutive one-port elements C, I and R are viewed as merely instantiations of the general multiports. In this way, the number of primitive elements is kept to a minimum. For compatibility with the literature a different symbol (double stroke capital) is used when more than one bond is connected.

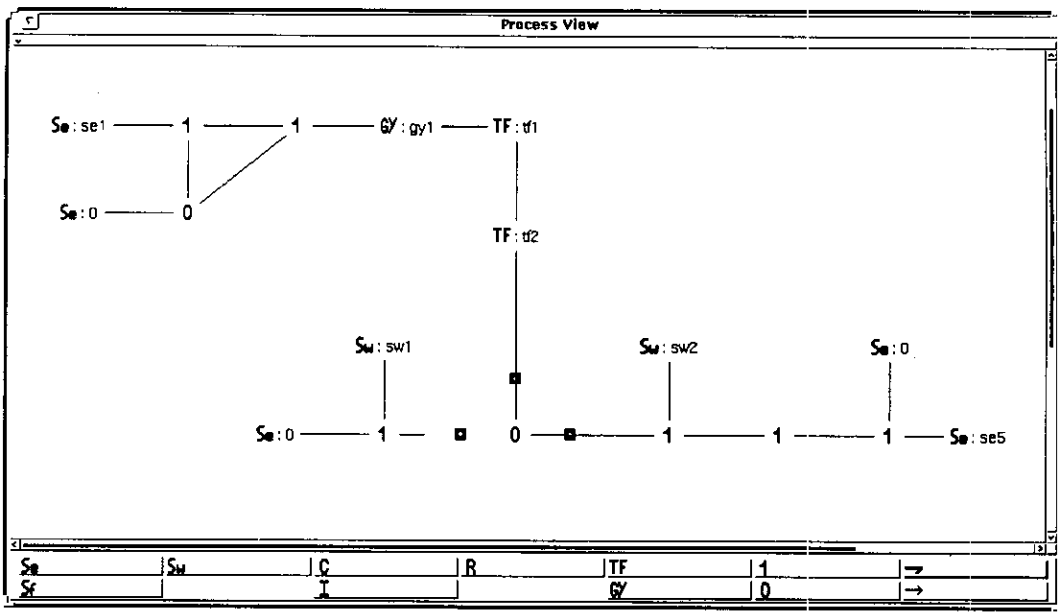

FIGURE 14. The process view for the pump system at an initial stage of the modelling process. At the point shown here the process-level specification for the *reservoir* is being given. Simply, an 0-junction is assumed to be adequate. It is connected to the ports of the selected component, the ports being visible both at this level and that of the components. Since the latter had already been interconnected, the new energy bonds are automatically extended to the associated 1-junctions and the *TF*-element. After filling all elementary components, the bond orientations have to be assigned and zero-output sources have to be removed.

Besides the standard set of bond graph elements, the switching element *Sw* (Strömberg *et al.*, 1993) can be used (Figure 15). It either has state *zero-effort* or *zero-flow*, initially set by the user. The state of the switch changes during simulation if the threshold of the active condition is exceeded. The conditions may refer to any variable in the system, including time. Note that this facilitates the construction of arbitrary step- and pulse-signals if the switch is combined with a source element.

Both energy bonds and signals can be used, although the latter are idealizations of *physical* interactions. Although internally (and for external storage) the bonds are identified by the elements they connect, they are also assigned a sequential number when introduced into the model, in order to simplify the names of bonds and variables. A physical subdomain can be assigned to each bond. Domain assignments are propagated automatically through 0- and 1-junctions.

In agreement with the bond graph theory, active bonds are only allowed to run from a 0- and 1-junction, which determines the signal that the bond represents, to *TF, GY, R, Se, Sf* or *Sw*. At the process level no further details are given about the mathematical structure in order to stay as close to the physical perspective as possible. Active bonds at the process level only indicate the existence of an influencing signal within the model, which is necessary to detect the existence of causal paths. Any modulation from outside the *complete* model must be specified in the *modulation relation* of the respective element, as described below. This modulation depends only on time.

In QuBA, causal analysis is viewed as one of the most important tools to assess a model, e.g., as shown in Figures 15 and 16. Causality can be assigned manually to each bond, but more interestingly, it can be applied automatically using the SCAP-G algorithm (Top, 1993), an extension of SCAP (Breedveld, 1986; Rosenberg, 1987; Karnopp *et al.*, 1990). Moreover, it will first check that there are no empty ports left. Causal peculiarities such as conflicts at source elements, derivative causalities and algebraic loops (between R-elements or along junction structures) are reported and the assigned causality is shown in the graph. The order in which causality was generated by SCAP-G can be visualized by selecting the option *display causal number*. After assigning causality, the set of causal feedback loops can be detected and listed in the assessment window. If a loop is selected in this window, the participating bond and signals change colour in the graphical field, both at this level and at the signal level, enabling direct localization of loops in the graph (Figure 17).

The possibility that a model does not have exactly one unique solution for its steady state can be detected already at the bond graph level by *equilibration* (Breedveld, 1984*a*; Iwasaki, 1988). In QuBA this test can be performed automatically. In addition, a graphical procedure for *exaggeration* (cf. Weld, 1990) has been implemented in the system (Top, 1993).

6.3. SIGNAL LEVEL

At the signal level the variables, parameters and mathematical relations can be accessed. The block diagram is used to visualize the model structure at this level (Figure 18). In order to retain the link with the physical level, the layout of this block diagram is determined by the bond graph topology. In fact, every bond graph element provides its own relations for the signal level. For example, a C-element contributes an algebraic relation (or a set of relations) and one or more integrating or differentiating relations; cf. Figure 19. Moreover, in addition to these "intrinsic" relations, for each bond graph element a so-called *modulating relation* is defined.

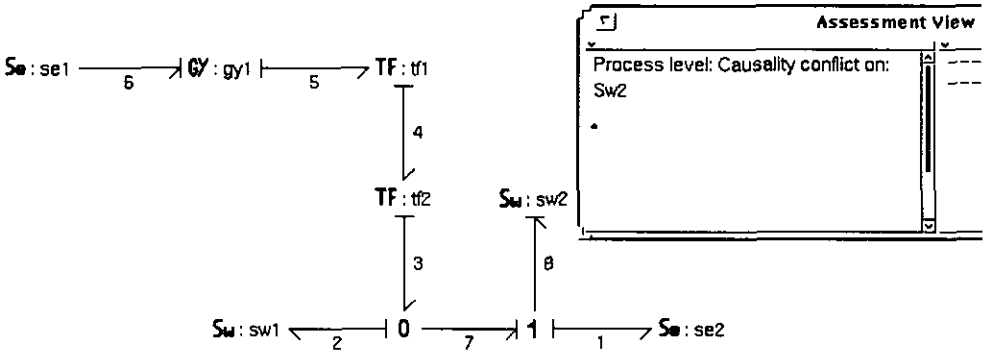

FIGURE 15. Process view for the pump system. The reference sources shown in Figure 14 have been eliminated and the bond graph has been simplified. When the switches are both in effort-mode, a causality conflict is detected at one of them. The assessment window reports this conflict. Note that the view is now set to display the sequential number reflecting causal order.

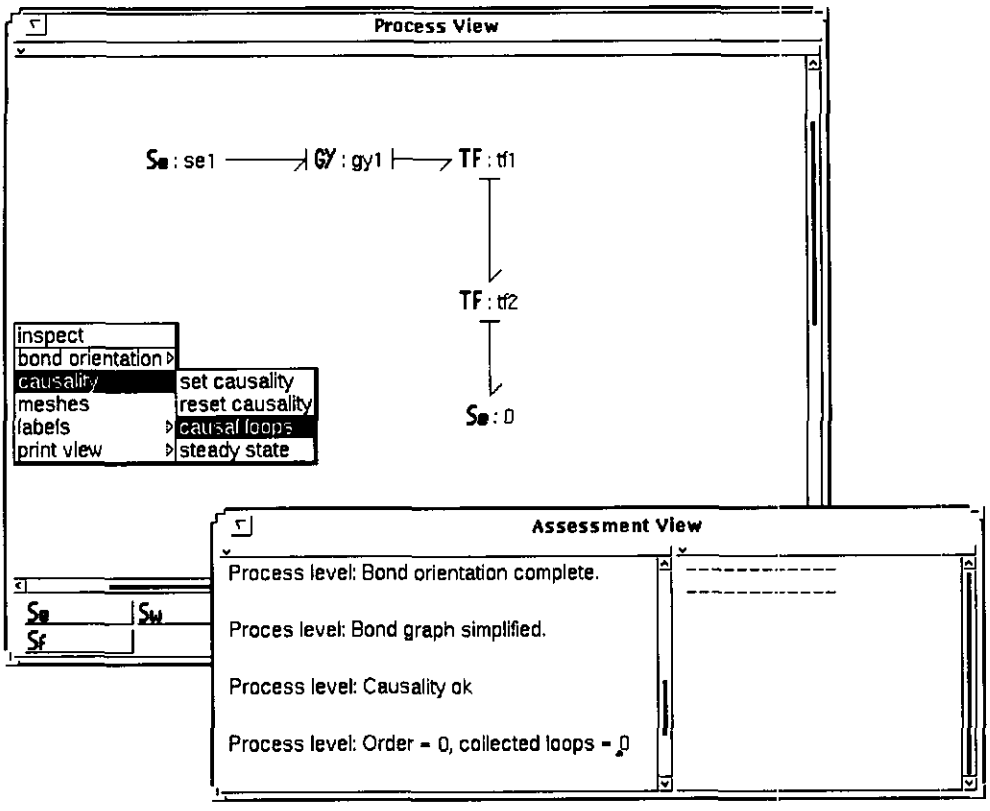

FIGURE 16. The process view after elimination of the switches. Now causality appears to be in order. At this stage the model has no causal loops and is of zero-order, as follows from the causal analysis.

This allows the user to specify any non-linear relation, even if the element cannot actually be modulated by an external variable. Every parameter is an output of a modulation relation. In this way, the possibility of assigning variable parameter values is introduced in a natural way and the introduction of non-linearity in general is facilitated.

At the signal level the set of causal paths between any two variables or from any variable to all inputs can be found and listed. The second possibility immediately displays the dependency of a quantity on external influences. By selecting one of the causal paths in the list, it is highlighted in the graph. Finally, there is an option to write the list of causal relations for inspection or further processing (Figure 19).

The usual way to assess the behaviour of a system is to inspect the values of the variables of interest over time. For that purpose, a simulation run can be performed and the results can be displayed graphically. In *QuBA* the emphasis is one the qualitative modelling process more than on numerical analysis. We rather leave this to dedicated packages for simulation and analysis, such as CAMAS (Broenink, 1990; Broenink *et al.*, 1992). However, to enable simple simulations an ordinary 4th-order Runge-Kutta integration algorithm has been implemented (Figure 20).

A limited simulation facility is particularly useful for demonstrating models with

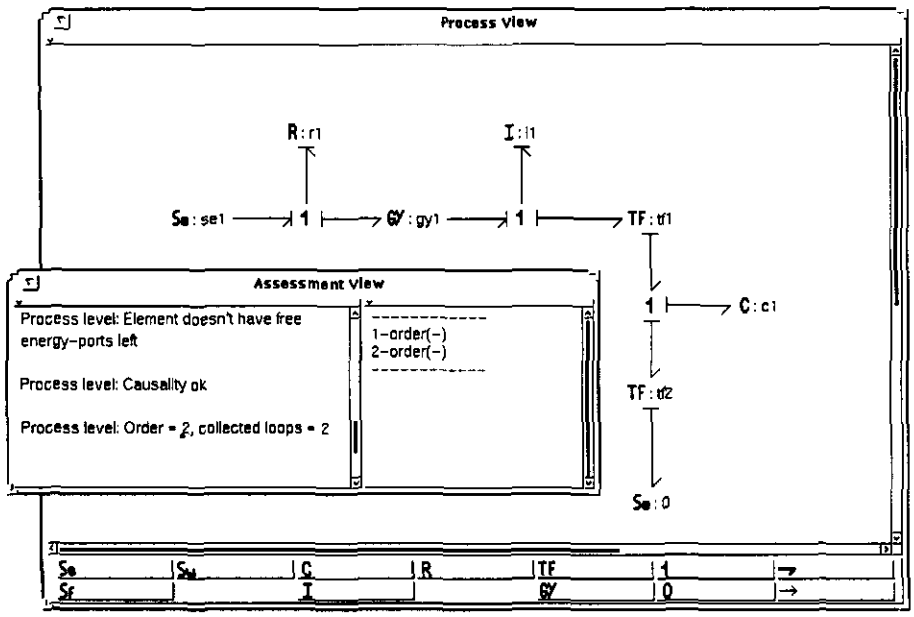

FIGURE 17. The process view for the extended pump model. Two causal loops have been detected. When one of the loops is selected in the assessment view it becomes visible in the bond graph by changing colour. Note that the assessment window contains a message that resulted from a previous editing action: an attempt was made to make an illegal energetic connection.

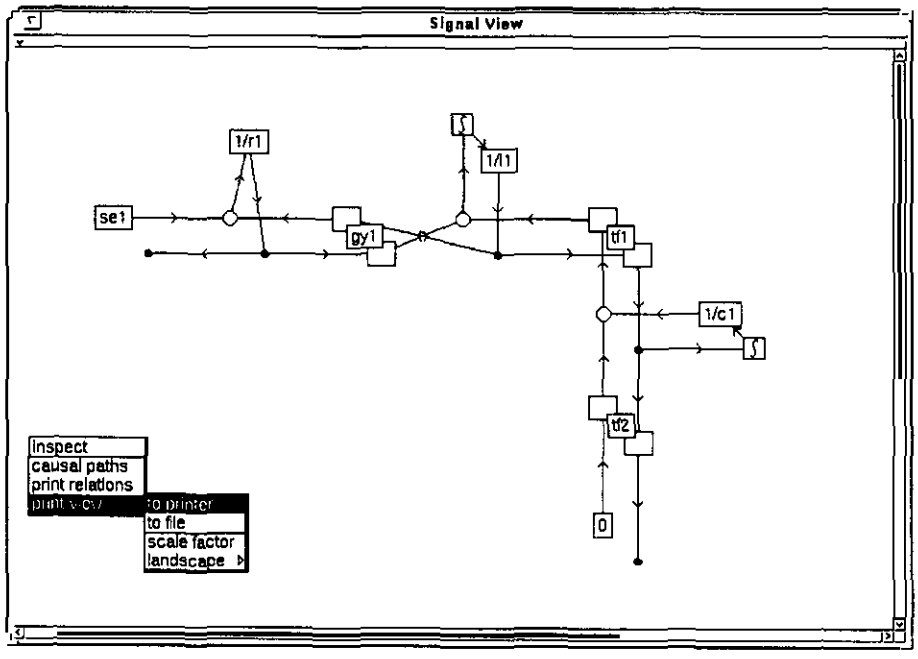

FIGURE 18. The signal view, showing a structured view of the mathematical model for the pump system. Blocks refer to relations, circles to summations and connections represent variables. By selecting them in the graphical field they can be edited.

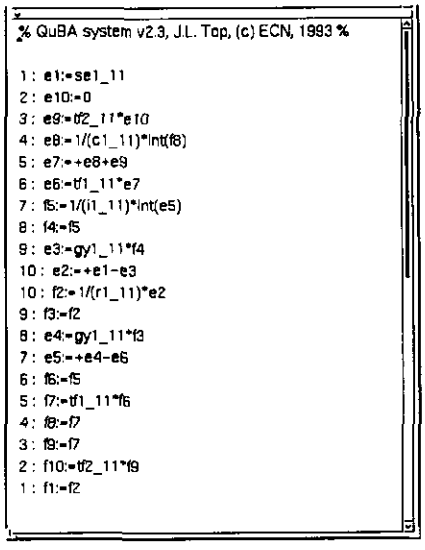

FIGURE 19. The relations generated at the signal level in causal order.

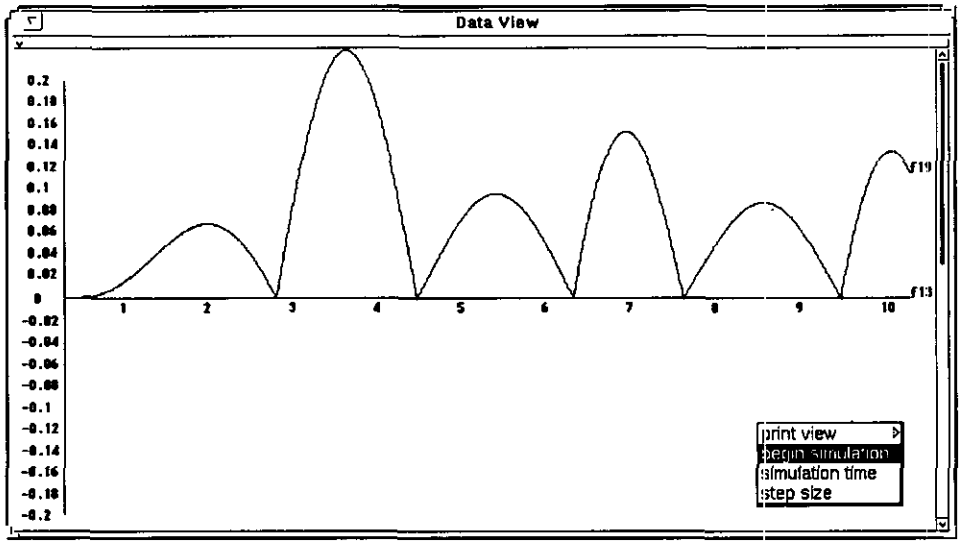

FIGURE 20. Signal level analysis by simulation. In this diagram the behaviour of the input and output flows are given. Since the input resistance of the pump is relatively small, the amplitude for the flow (f_{19}) is initially large. The output flow (f_{13}), on the other hand, increases before it reaches a steady amplitude.

dynamic causality, i.e. models that contain the switching element *Sw*. In Strömberg *et al.* (1993) and Top (1993) it has been shown that the structure of the mathematical model is dynamic if the switching element *Sw* is used. This is due to the fact that different causal directions arise for the different states of the switch, representing a changing system boundary. This means that the causal set of relations that is actually being used for simulation will have to be rearranged and recompiled during run-time. In QuBA the causal set of relations is generated in a dynamic way. Moreover, if a bond graph element has one or more operating conditions associated with it, these are collected and compiled as well. Presently this is only applied for defining the switching conditions of the *Sw*-element, but in future any *scope of a quantity of interest* can be included as a part of the monitoring of operating conditions. Now, if one of the conditions is violated it is checked which switch has to change state. Next, SCAP-G is applied, and the new set of relations is compiled.

6.4. SUMMARY AND FUTURE DEVELOPMENTS

QuBA is an experimental environment based on the *evolutionary modelling* approach for energetic-dynamic systems. It allows interactive, graphical development of bond graph-based models. In the design of the system most attention has been given to the pre-mathematical stages of the modelling process. The main features of QuBA are:

- Separation of conceptual levels in terms of functional components, bond graphs, mathematical relations.
- Plug-in facility of submodels at the functional component level for either subcomponents or process models (bond graphs).
- Support for the subtasks of modelling and design:
  - —*Specification.* Graphical editing facilities. Display of causal paths and loops.
  - —*Construction.* Assembling syntactically correct models. Generation of causal equations.
  - —*Assessment.* Model analysis facilities such as model simplification, causal analysis, equilibration and simulation.
- Dynamic causality introduced by the switching element.

In future versions of QuBA the following aspects should be further elaborated:

- *Assumption management.* In line with the evolutionary modelling approach, the quantities of interest, their ranges and accuracies should be handled explicitly. Moreover, the use of other meta-assumptions should be supported, in particular in relation with the selection of library models. The latter will necessitate the development of a useful taxonomy of functional components.
- *Model library.* The availability of an extensive model library is crucial. Presently, a major effort towards the development of such a library and its contents is being made in the ESPRIT 6521 project OLMECO. A future version of QuBA should be able to access this library. Within OLMECO the intention is to define a standard representation format, based on SIDOPS (Broenink, 1986). A future version of QuBA should at least be able to read and write this intended standard, or even use it as its internal format.

- *Mathematical analysis.* Of lower priority is the extension of the numerical machinery of QuBA, in particular if the internal format is to be compatible with that of packages like CAMAS. However, it would be interesting to investigate the possibility of employing external software libraries for advanced numerical integration and analysis techniques. Proper assumption management and adaptive modelling is only possible in an integrated environment.

7. Conclusion

Advancing automated modelling of physical systems is important to reduce the engineering bottleneck in the development of technical systems. Our work attempts to contribute to this by integrating and extending ideas and methods from both artificial intelligence and conventional engineering.

We have outlined a comprehensive and principled framework for the task of modelling in physical systems engineering. It is distinctive with regard to other approaches in several respects.

- We have emphasized the importance of the interactive and iterative aspects of modelling, elements which tend to be neglected in computer science views on modelling. The task of specification of modelling assumptions and decisions is crucial in model building. Current AI approaches are basically limited to model selection from a library given a predefined set of assumptions.
- The proposed set of ontologies and the generic way they are connected provides a structured organizing framework for libraries of reusable physical models. Current AI model fragment libraries provide a very flexible framework, but a rather too flexible and open one. Because of this feature of "*anything goes*", they tend to be *ad hoc* and very much application dependent.
- Practice shows that *ontological commitment* is crucial for communication about and exchange of models.
- The representations we use derive from standard physics and engineering. This has two advantages: (i) it makes AI work more useful and acceptable for practicing physicists and engineers; and (ii) a lot of general physics knowledge is automatically built in. This is still lacking in AI approaches such as compositional modelling.

This is why we find these approaches in their current state to be insufficiently structured and accidental in their relationship to conventional physics and engineering. Although in recent years the appreciation in AI of the role of mathematics has grown, there is much more to physical systems engineering than this alone. Any attempt concerned with automated modelling should in our view build on existing achievements in the physical and engineering sciences.

Moreover, viewing modelling as a matter of repeated selection and assembly of model fragments for a predefined set of requirements, as is often done in AI, is too much of a simplification. This simplification is based on the assumption that requirements and model fragments are more or less independent entities, which is a tempting viewpoint within the context of automation. This, however, suggests that a modelling scheme can be developed independently from the view of the world that is being taken by the modeller, as if some general modelling algorithm can be

developed in which any ontology can fit. Our claim is that, on the contrary, model-based reasoning is a matter of switching between different views of the world. Each view defines a coherent and more or less self-contained set of concepts and defines both the set of potential requirements and the set of possible models.

There are several ways to make ontological distinctions. A natural borderline exists between *fields of expertise,* such as psychology, chemistry and physics, or within the latter: fluid dynamics, solid state physics or quantum mechanics. Moreover, we suggest that for each of these fields a comprehensive computational model contains a component level, a process level and an information level, as we proposed for the domain of macrophysical systems. The component level defines the "real" entities that make up this particular world and their interconnections. These components perform certain functions. The process level states which mechanisms can bring about changes. Finally, the information level provides an abstract view in terms of variables and relations between variables, thus enabling prediction and explanation through computation. It is essential for model-based reasoning to be aware of the current ontological point of view at any time in order to focus the problem-solving process.

This work has been partially supported by the CEC Esprit-III projects P6521 "OLMECO" and P5248 "KADS-II" and by the Netherlands Ministry of Economic Affairs/Senter in the framework of the "EBIB" Telematics Guidance Project. The partners in the EBIB project are Kropman BV, the Netherlands Energy Research Foundation ECN and University of Twente. The partners in the OLMECO project are PSA Peugeot Citroën (France), BIM (Belgium), Fagor (Spain), Ikerlan (Spain), Imagine (France), University of Twente (The Netherlands) and ECN (The Netherlands). The KADS-II consortium includes Cap Gemini Innovation (France), Cap Programator (Sweden), ERITEL SA (Spain), IBM France, Lloyd's Register (UK), SICS (Sweden), Siemens AG (Germany), Touche Ross Management Consultants (UK), University of Amsterdam (The Netherlands), Free University of Brussels (Belgium) and ECN (The Netherlands).

References

ADDANKI, S., CREMONINI, R. & PENBERTHY, J. S. (1991). Graphs of models. *Artificial Intelligence,* **51,** 145–177.

AKKERMANS, J. M., VAN HARMELEN, F., SCHREIBER, A. TH. & WIELINGA, B. J. (1993). A formalisation of knowledge-level models for knowledge acquisition. *International Journal of Intelligent Systems,* **8,** 169–208. Also in: K. M. FORD & J. M. BRADSHAW, Eds. *Knowledge Acquisition as Modeling,* pp. 169–208. New York: Wiley.

BOBROW, D. G., Ed. (1984). *Qualitative Reasoning about Physical Systems.* Amsterdam: North-Holland.

BREEDVELD, P. C. (1984*a*). A bond graph algorithm to determine the equilibrium state of a system. *Journal of the Franklin Institute,* **318**(2), 71–75.

BREEDVELD, P. C. (1984*b*). *Physical systems theory in terms of bond graphs.* Ph.D. thesis, University of Twente, Enschede, The Netherlands.

BREEDVELD, P. C. (1986). A systematic method to derive bond graph models. In G. C. VANSTEENKISTE, E. K. H. KERCKHOFFS, I. DEKKER & J. C. ZUIDERVAART, Eds. *Proceedings 2nd European Simulation Congress, Antwerp,* pp. 38–44.

BREEDVELD, P. C., ROSENBERG, R. C. & ZHOU, T. (1991). Bibliography of bond graph theory and application. *Journal of the Franklin Institute,* **328**(5/6), 1067–1109.

BROENINK, J. F., (1986). SIDOPS, a bond based modelling language. In S. G. TZAFESTAS & P. BORNE, Eds. *IMACS Transactions: Complex and Distributed Systems: Analysis, Simulation and Control,* pp. 81–86. Amsterdam: North-Holland.

BROENINK, J. F. (1990). *Computer-aided physical-systems modeling and simulation: a bond-graph approach.* Ph.D. thesis, University of Twente, Enschede, The Netherlands.

BROENINK, J. F., BEKKINK, J. & BREEDVELD, P. C. (1992). Multibond-graph version of the CAMAS modeling and simulation environment. In P. C. BREEDVELD & G. DAUPHIN-TANGUY, Eds. *Bond Graphs for Engineers,* pp. 253–262. Amsterdam: North-Holland.

CHANDRASEKARAN, B. (1988). Generic tasks as building blocks for knowledge-based systems: the diagnosis and routine design examples. *The Knowledge Engineering Review,* **3**(3), 183–210.

COYNE, R. (1990). Logic of design actions. *Knowledge-Based Systems,* **3,** 242–257.

DE VRIES, T. J. A. (1994). *Conceptual design of controlled electro-mechanical systems.* Ph.D. thesis, University of Twente, Enschede, The Netherlands.

FALKENHAINER, B. & FORBUS, K. D. (1991). Compositional modeling: finding the right model for the job. *Artificial Intelligence,* **51,** 95–143.

FALKENHAINER, B. & STEIN, J. L., Eds. (1992). *Automated modeling,* DSC-Vol. **41,** ASME Winter Annual Meeting, 1992.

GERO, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine,* **11**(4), 26–36.

GOEL, A. & CHANDRASEKARAN, B. (1989). Functional representation of designs and redesign problem solving. *Proceedings IJCAI'89,* pp. 1388–1394. San Mateo, CA: Morgan Kaufmann.

GRUBER, T. R. (1993). Model formulation as a problem-solving task: computer-assisted engineering modelling. *International Journal of Intelligent Systems,* **8,** 105–127. Also in: K. M. FORD & J. M. BRADSHAW, Eds. *Knowledge Acquisition as Modeling,* pp. 105–127. New York: Wiley.

IWASAKI, Y. (1988). Causal ordering in a mixed structure. *Proceedings AAAI-88,* pp. 313–318, Saint-Paul, MN.

KARNOPP, D. C., MARGOLIS, D. L. & ROSENBERG, R. C. (1990). *System Dynamics: A Unified Approach,* 2nd edition. New York: Wiley.

MAHER, M. L. (1990). Process models for design synthesis. *AI Magazine,* **11**(4), 49–58.

MARCUS, S. & MCDERMOTT, J. (1989). SALT: a knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence,* **39**(1), 1–38.

NAYAK, P. P. (1992). *Automated modeling of physical systems.* Ph.D. thesis, Stanford University, CA, USA.

PAHL, G. & BEITZ, W. (1988). *Engineering Design—A Systematic Approach.* The Design Council. London: Springer-Verlag.

PAYNTER, H. M. (1961). *Analysis and design of engineering systems.* Cambridge, MA: MIT Press.

ROSENBERG, R. C. (1987). Exploiting bond graph causality in physical system models. *Trans. ASME J. Dyn. Syst. Meas. Control,* **109,** 378–383.

SCHREIBER, A. TH., WIELINGA, B. J. & BREUKER, J. A., Eds. (1993). *KADS: A Principled Approach to Knowledge-Based System Development.* London: Academic Press.

STEIN, J. L., Ed. (1991). *Automated Modeling,* DSC-Vol. **34,** ASME Winter Annual Meeting, 1991.

STEIN, J. L. & ROSENBERG, R. C., Eds. (1991). *The Art of Physical System Modeling: Can It Be Taught?* DSC-Vol. **36,** ASME Winter Annual Meeting, 1989.

STRÖMBERG, J. E., TOP, J. & SÖDERMAN, U. (1993). Variable causality caused by discrete effects. *Proceedings of ICBGM'93, San Diego,* pp. 115–119. San Diego, CA: SCS.

TOP, J. L. (1993). *Conceptual modelling of physical systems.* Ph.D. thesis, University of Twente, Enschede, The Netherlands. ISBN 90-375-0291-1.

TOP, J. L. & AKKERMANS, J. M. (1991). Computational and physical causality. *Proceedings IJCAI-91.* pp. 1171–1176. San Mateo, CA: Morgan Kaufmann.

TOP, J. L. & AKKERMANS, J. M. (1993). Layered modelling of physical systems. *Proceedings of the 1993 Winter Annual Meeting,* DSC-Vol. **47,** pp. 95–107. ASME.

TOP, J. L. & AKKERMANS, J. M. (1994). Evolutionary modelling: a structured approach to model construction. *Proceedings of the 1994 IMACS MathMod Symposium,* pp. 124–127, Vienna, Austria: IMACS.

VAN DIJK, J., DE VRIES, T. J. A., BREUNESE, A. P. J. & BREEDVELD, P. C. (1992). Automated mechatronic systems modelling using MAX. In P. C. BREEDVELD & G. DAUPHIN-TANGUY, Eds. *Bond Graphs for Engineers*, pp. 279–290. Amsterdam: North-Holland.

VAN DIJK, J. (1994). *On the role of bond graph causality in modelling mechatronic systems.* Ph.D. thesis, University of Twente, Enschede, The Netherlands.

WELD, D. S. & DE KLEER, J., Eds. (1990). *Readings in Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann.

WELD, D. S. (1990). Exaggeration. *Artificial Intelligence*, **43**(3), 311–368.

WIELINGA, B. J., SCHREIBER, A. TH. & BREUKER, J. A. (1992). KADS: a modelling approach to knowledge engineering. *Knowledge Acquisition*, **4**(1), 5–53.

WIELINGA, B., VAN DE VELDE, W., SCHREIBER, A. TH. & AKKERMANS, J. M. (1993). Towards a unification of knowledge modelling approaches. In J. M. DAVID, J. P. KRIVINE & R. SIMMONS, Eds. *Second Generation Expert Systems*, pp. 299–335. Berlin: Springer-Verlag.

Paper accepted for publication by Editor Professor B. R. Gaines.