Implementation of Component-based Simulation Support Tool for Conceptual Design

T. Brandejský

Presented paper speaks about the problem of conceptual design stage simulation support. Simulation support is a very useful part of conceptual design system due to capability to verify ideas in early design stages. The problem excluding classical simulation tools is lack of information about designed device, is inconsistency and uncertainty. Thus specialised tool must be developed. The component-oriented editor of component descriptions and models is presented. This tool enables to describe components not only in terms of algebraic equations, but also by fuzzy rules. The problem of dynamic work with uncertainty representation during simulation and design processes is also solved. Presented tool also differentiates from standard tools like Mathematica or Matlab in its ability to work with component-based models, where each component is described from many aspects and only few of them are valid in concrete use. The tool must be able to select relations relevant in concrete simulation task and omit the rest.

Keywords: component-based simulation, conceptual design, mixed uncertainty description, heterogeneous model, dynamic uncertainty representation selection.

1 Introduction

Conceptual design is a term representing design activities before own modelling and drawing of the system by any CAD system. The conceptual design is a process of formulation and precisiation of vaguely formulated preconditions, function descriptions and structure of a designed system. Due to lack of information in these early design stages, we must use proper techniques, especially qualitative. We also must reason, that information about system increases during design process and so used formalism must enable movement from imprecise to more precise, from qualitative to quantitative description.

Design process is from many views optimisation process. Process not only decreasing with uncertainty, but also searching as better solution as possible within limits given by technical, physical and economical constraints. Optimisation needs to have on its background model and simulation tool. In field of conceptual design we also expect that such a tool is also capable to provide part-dimensioning work and to simulate under conditions of uncertainty. Sources of uncertainty are especially incompleteness of model, its vagueness and inner contradictions.

Human designers decrease design problem complexity by decomposition techniques. This approach enables them to use known parts and subsystems and focus their creativity on limited sub-problems. Thus, simulation support tool for conceptual design must be also component-based.

Because each component is represented by special file with structure enabling direct reading by Prolog consult predicate, inheritance is solved by special predicate in this description. Inheritance is allowed only single, not multiple like in C++, because used model contains object collection. Their use is in the field of modelling usually clearer than the use of multiple inheritances. E.g. the C++ programmers use often multiple inheritances on the place of static collections because multiple inheritance mechanism better fits information systems features. But in the field of technical systems ones the grouping of different functions into one indivisible system is not frequent.

2 Component based model for conceptual design

Each component-based simulation tool consists from two fundamental parts: component editor and own simulator. Presented editor supports all model features including inheritance, encapsulation, structural information (physical types of variables) and some consistency checks. The editor also enables work with component libraries, see figure 1. Editor distinguishes two types of components, simple and container ones. The structure of model is described by the work [1]. Model is described by Prolog-like structures. An example is sketched on the Table 1.

Simulation tool developed in Prolog language works in three ways. In the first presented simulation tool works as

Table 1: An Example of container component derived from parent component class "tank" with new variable *t*, physical meaning temperature with default value 1.1 and with inherited object heating of exchanger class

```
parent(P:\Tom\creativity\CMB\heating\tank.scf)

variable(t,real,temperature, 1.10000000000000E+0000)

inherited(heating,P:\Tom\creativity\CMB\heatying\exchanger.scf)
```


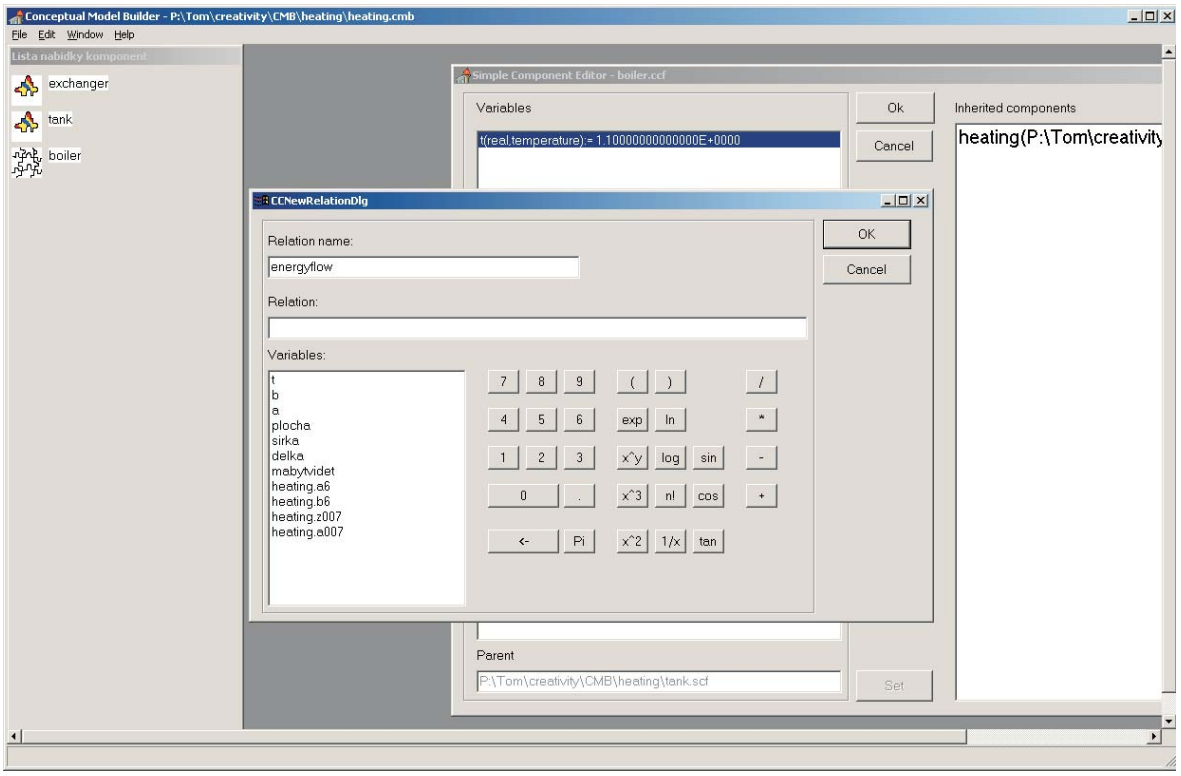

Fig. 1: Component Editor editing relation of container component

pre-processor generating on the base of component libraries models for commercial simulators like Matlab or Mathematica (selects relevant relations – equations). In the second simulation tool generates and simulates qualitative model (from algebraic relations or from functional ontology). In the last simulation tool uses all its' capabilities, selects proper relations like in last cases and simulates system with or without conditions of uncertainty. Model structure can be derived from UML description by special mapping scheme described in the work [2].

Used uncertainty model is based on interval-valued fuzzy sets. As it is known from many works of me, Atanassov, Mizumoto and Tanaka, interval fuzzy sets enables to describe intervals of possible values, fuzzy uncertainty and, because they are a special case of second order fuzzy sets, also rough approximation of probabilistic uncertainty.

Because the use of universal description is computationally intensive and brings problems in cases of state variable integration and some functions, an alternative way based on the use of dynamic representation choice from given set of representations (fuzzy numbers, fuzzy linguistic variables, intervals, interval-valued fuzzy sets) on the base of input data uncertainty type and operations used in computed relations will be also discussed in the next part.

In the case of conceptual design it is impossible to eliminate uncertainty e.g. by defuzzyfication, because uncertainty distribution brings significant information about sources of uncertainty – about wrongly determined system components, about contradictions in designed system etc.

2 Methods applicable in the area of heterogeneous models of systems

Within the conceptual design stage, the function of designed system can be described not only in the form of algebraic equations, but also in the form of fuzzy rules [3]. Similarly, we must reason parameters and initial conditions of simulation minimally in three forms – crisp data, fuzzy numbers and fuzzy linguistic variables.

Thus we must solve four basic cases: crisp data – algebraic equations, crisp data – fuzzy rules, fuzzy data – fuzzy rules, fuzzy data – algebraic operations and their combinations. We must also distinguish fuzzy data in the form of predefined

Table 2: Basic methods applicable for partial combinations of operation and data descriptions

Operations	Data		
	Crisp	Fuzzy numbers	Fuzzy linguistic variables
Algebraic operations	Numerical mathematics	Operations with fuzzy numbers	Method AOFULV
Rules	Fuzzyfication of fuzzy rule and Defuzzyfication	Similarity based reasoning	Fuzzy rules

fuzzy linguistic variables and fuzzy data in the form of fuzzy numbers (which usually has dynamic structure). Table 2 represents basic methods applicable for solving of each combination.

We can recognise six basic situations in Table 2. The following points describe each method more in detail.

- Numerical mathematics represents sophisticated mathematical discipline applicable in the area of crisp (numerical) data and algebraically described operations. There are many information sources, e.g. [4].
- Fuzzy numbers operations are used in situations, where input data contains fuzzy uncertainty and are described as fuzzy numbers. Operations are described algebraically. From the viewpoint of Pedrycz [5] granulation, fuzzy numbers can be in some application better than fuzzy linguistic variables due to better approximation of uncertainty. Analogously, algebraic operations with fuzzy numbers do not increase uncertainty of system description in contrast with fuzzy-rules. Increasing of uncertainty in the case of fuzzy--rules use is analyzed in the work [6]. The application of fuzzy numbers is possible only with respect to particular limitations, especially the need of use convex fuzzy numbers. The use of non-convex fuzzy numbers is possible (e.g. they are used in AOFULV method), but their use tends to computationally intensive calculations.
- AOFULV method (Algebraic Operations with Fuzzy Linguistic Variables) was developed for calculation of Algebraic Operations with data in the form of Fuzzy Linguistic Variables. The construction is following:
  The memberships of resulting variable R values (after execution of operation "*") are computed as maximums of variable R k-th linguistic values membership function f_m^P conjunction with product of all partial operations with elements $K(e_1, ..., e_m)$ of Cartesian product K unification:

$$\mu_k^R = \max\left\{ f_k^R \cap \left[\Theta\left(f\left(K\left(e_1,\ldots,e_m\right)\right)\right)\right]\right\} * \\ * \min\left\{\mu\left[K\left(e_1,\ldots,e_m\right)\right]\right\} \qquad (1)$$

  Symbol $f(K(e_1, \ldots, e_m))$ denotes m-dimensional vector of membership functions which form the element of Cartesian product $K = A_1 \times A_2 \times K \times A_m$. The membership function of e-th linguistic value of k-th linguistic variable there is understood as a fuzzy number. Symbol "*" (in (1)) represents an algebraic function with fuzzy numbers (not only elementary algebraic operation, like "+", "–", "*", "/"). The method is described e.g. in [6]. Most universal than fuzzy sets are interval valued fuzzy sets and membership interval fuzzy sets [7, 8].
- Fuzzyfication → Fuzzy-rules → Defuzzyfication: Such an operation sequence represents approach well-known from many implementations in areas of automatic control and fuzzy modeling. The approach applies rule-based description of a system on crisp data calculations. In the way it uses two transformations between crisp and fuzzy linguistic variable description – fuzzyfication and defuzzyfication. We often speak about fuzzy approximation of (unknown) function. The possibility of the way is proved by FAT theorem [9]. The problem is the selection of optimal combination of fuzzyfication and defuzzyfication operations and method of rule result membership calculation. We can choice from many viewpoints, e.g. from approximation linearity viewpoint.
- Analogical reasoning: The application of operations described by rules on fuzzy-numbers is difficult, because shape of fuzzy-number membership function changes during the time of calculation and so fuzzy-numbers changes its sense. On the other hand, rule-based reasoning is based on predefined terms of previously defined meaning.
- Thus application of rule-described operations on data in the form of fuzzy-numbers is possible only in the case of existence of transformation between fuzzy-numbers and fixed-meaning terms – fuzzy linguistic values.
  The possible way is represented by measuring of membership function similarity, for some methods see [10]. Similar approach is used for transformation of unified partial results to linguistic values memberships in AOFULV method.
- Fuzzy-rules: The application of operations described by fuzzy-rules on data in the form of fuzzy linguistic variables is simple and does not need any additional transformations.

3 Dynamic selection of uncertainty description

It is possible to solve uncertainty description selection problem and from it concluding implementation of operations from two viewpoints. Either our goal is to describe influence of initial parameters and operations uncertainty on precision of output parameters with maximal credibility (it usually is in conceptual design field) or our gal is to choice most effective method of work with uncertainty information saving the basic information about distribution of uncertainty (this situation become in the case of complex systems, when it is possible to eliminate information about uncertainty influence in the case of less significant components from the viewpoint of actually solved design operations).

In the following text transcription $X \Rightarrow Y$ denotes that X is transformed during evaluation to Y, large chars denotes variables, symbol '_' is used for undefined value, $op\left(X_1, K, X_n, R\right)$ denotes n-ary operation op, $op\left(type(X), type(R)\right)$ then means unary operation op with argument of $type$ type and with value X and result of $type$ type and value R. Analogously, the form $op\left(T_1(X), T_R(R)\right)$ then represents unary operation op with T_1 type argument and value X and with the result of T_R type and value R.

Particular uncertainty description types are denoted by Table 3:

Table 3: Uncertainty representations naming

Crisp	*crisp*(*X*)
Singleton	*sgltn*(*X*)
Fuzzy number	*fn*(*X*)
Fuzzy linguistic variable	*flv*(*X*)
MI fuzzy number	*mifs*(*X*)
MI linguistic variable	*milv*(*X*)

3.1 Algebraically described operations

Algebraically described operations represent basic type of system behaviour description from viewpoint of presented system of conceptual design simulation support. Thus this type will be discussed in this chapter. Second type is rule based description. This type will not be discussed on this place because the solution is analogous.

3.2 Uncertainty description type solution for the case where uncertainty description of operation result is not given

The two basic cases becomes from the viewpoint whether uncertainty description type of the partial-result one is not given, or it is; and from the method of operation evaluation. The situation, when result uncertainty description type is given becomes usually as consequent of user choice (e.g. the postulate for representation by pre-defined linguistic variables on the place of fuzzy numbers which must be interpreted and which are not suitable e.g. for co-operation of simulation system and of expert system).

3.3 Transformations of argument type

Unary operator

Unary operator is described by ordered twin of parameters – description of value and uncertainty of an argument and description of value and uncertainty of the result. If concrete type of uncertainty description of the result is not asked the type of uncertainty of the result is the some as the one of the argument.

$$op(N(X), \_(R)) \Rightarrow op(N(X), N(R)) \tag{2}$$

Binary operator

$$op(N(X_1), N(X_2), \_(R)) \Rightarrow op(N(X_1), N(X_2), N(R)) \tag{3}$$

$$op(fn(X_1), crisp(X_2), \_(R)) \Rightarrow (crisp(X_2) \Rightarrow sgltn(X_2)), op(fn(X_1), sgltn(X_2), fn(R)) \tag{4}$$

$$op(flv(X_1), crisp(X_2), \_(R)) \Rightarrow (crisp(X_2) \Rightarrow sgltn(X_2)), op(fn(X_1), sgltn(X_2), fn(R)) \tag{5}$$

$$op(flv(X_1), fn(X_2), \_(R)) \Rightarrow op(flv(X_1), fn(X_2), fn(R)) \tag{6}$$

$$op(mifs(X_1), crisp(X_2), \_(R)) \Rightarrow (crisp(X_2) \Rightarrow sgltn(X_2)), op(mifs(X_1), sgltn(X_2), mifs(R)) \tag{7}$$

$$op(mifs(X_1), fn(X_2), \_(R)) \Rightarrow op(mifs(X_1), fn(X_2), mifs(R)) \tag{8}$$

$$op(mifs(X_1), flv(X_2), \_(R)) \Rightarrow op(mifs(X_1), flv(X_2), mifs(R)) \tag{9}$$

$$op(milv(X_1), crisp(X_2), \_(R)) \Rightarrow (crisp(X_2) \Rightarrow sgltn(X_2)), op(milv(X_1), sgltn(X_2), mifs(R)) \tag{10}$$

$$op(milv(X_1), fn(X_2), \_(R)) \Rightarrow op(milv(X_1), fn(X_2), mifs(R)) \tag{11}$$

$$op(milv(X_1), flv(X_2), \_(R)) \Rightarrow op(milv(X_1), flv(X_2), mifs(R)) \tag{12}$$

$$op(milv(X_1), mifs(X_2), \_(R)) \Rightarrow op(milv(X_1), mifs(X_2), mifs(R)) \tag{13}$$

Transformations of argument description types and high order operation results are analogous and for short will not be described in detail on this place.

3.4 Transformation of pre-defined type of result

It is possible to transform this situation on previous case with subsequent transformation of uncertainty description type into asked; or it is possible to start from asked type of uncertainty and search the most simple sufficiency method of solving. The first case can be described by transformations (14) – for unary operator and (15) for binary:

$$op(P(X), G(R)) \Rightarrow op(P(X), \_(R)), (\_(R) \Rightarrow G(R)) \tag{14}$$

$$op(P_1(X_1), P_2(X_2), G(R)) \Rightarrow op(P_1(X_1), P_2(X_2), \_(R)), (\_(R) \Rightarrow G(R)) \tag{15}$$

The search of effective way of calculation in the case of asked type of result uncertainty description is expressed by transformations (16) and (17). It is easy to add the other cases by analogous way. In every case we search such operation which is adequate to the most precise type of argument uncertainty representation increased on the level of the result uncertainty representation type (if it is higher).

$$op(P_1(X_1), P_2(X_2), crisp(R)) \Rightarrow (P_1(X_1) \Rightarrow crisp(X_1)), (P_2(X_2) \Rightarrow crisp(X_2)), op(crisp(X_1), crisp(X_2), crisp(R)) \tag{16}$$

$$op(crisp(X_1), crisp(X_2), T(R)) \Rightarrow op(crisp(X_1), crisp(X_2), crisp(R)), (crisp(R) \Rightarrow T(R)) \tag{17}$$

3.5 Composed operation case

In the case of composed operation the type of partial result representation is not ever defined. Everything is given by selected approach (maximal credibility or the precise adequate to asked type of uncertainty representation of the result of the whole composed operation).

In the first case the type of partial result uncertainty representation is selected by method described by chapter 3.3.

In the case of the effective method search the following rule is valid: the better type of partial result is not selected than the representation of the final result uncertainty is asked. If the arguments are of a less precise type, this type is selected. The following order of representation types is reasoned from the viewpoint of precise:

Crisp – singleton – fuzzy number - MIFS

Table 3: The order of uncertainty representation types from the viewpoint of precise.

Also this choice can be described by proper grammar, e.g.:

$$op_1\big(crisp(X_1), mifs(X_2), op_2\big(milv(X_3), \_\big), fn(R)\big) \Rightarrow op_1\big(crisp(X_1), mifs(X_2), op_2\big(milv(X_3), fn(\_)\big), fn(R)\big) \qquad (18)$$

4 Simulation model construction from component description

Many aspects of component description and simulation model derivation are described in the paper [1]. So, in this chapter after small recapitulation only novel approaches to model construction and simulation will be described. The first chapter of this paper brings introduction to components description and proper editor tool. Because many relations in this model describe value of the same variable, simulation tool must select optimal one on the base of values with known value (value determined by user or previous calculation). Simulation and dimensioning of technical system usually conduce to situations, when more relations than one must be use. This fact increases combinatorial complexity of the task of simulation model derivation from component model of device.

Used component model and component description differs from standard models (e.g. in SIMULINK) because model of component applicable in the field of conceptual design must be applicable in any possible use of the component. So, it must describe all component behaviours, but in concrete use only few of them will be relevant. The first version of mechanism of simulation model (solving set of equations) collecting was described in [1]. This method works similarly like Prolog language. It starts its work with selecting of equation determining value of asked variable (on the base of known and unknown attributes ratio). Then it tries recursively to determine its unknown attributes. If the way is wrong, it returns about one step up and selects different relation etc. This method is successful in one-component case and in case of encapsulated component without relations between them.

It cases when input on one encapsulated component depends on output of other, the different solution method must be used. In that situations it is need to look what next relations can be calculated, which next variables can be on the base of previous result determined.

The part of communication with the tool using both this method is sketched on Fig. 1. The prototype of conceptual design simulation support tool does not implement multiple uncertainty support and is used only for solving relation set collecting algorithm verification.

```
C:\Documents and Settings\brandejsky\My Documents\Doc a GACR\Exe
command line empty - enter root component (model) file name
..\cmb\gacr\x1.ccf
if you want to generate model enter 'g', otherwise anything else
z
I simulate model
Entering known variables
enter single name, then you will be asked for magnitude insertion
empty line ends entering,
n_in
n_in:=12
total_transmission_ratio
total_transmission_ratio:=0.1

enter names of asked variables
n_out


["n_out"]
n_out = 1.2
```

Fig. 2: Communication with conceptual design simulation support tool prototype

The tool on the beginning of session ask for the name of the main – root component (whole model can be understood as a container component). Then it ask if set of equations for specialised simulation tools will be generated or if the simulation will be provided by this tool, then it asks for variables with known magnitudes and on the end for asked variables. Then the tool collects solving set of equations and presents results. The tool is also capable to use default values on variables and in the future will be collected by mechanism of physical unit management (by now supported in component editor – see Fig. 1) and by uncertainty dynamic description management presented in chapter 3.

Conclusion

The presented paper on the background on conceptual design simulation tool summarises possibilities of the use of universal uncertainty management within the conceptual tool without the need of implicit conversion on the most common possibilistic uncertainty representation. The method applicable on rule based description is not discussed because the solution is analogous with the method for algebraic one. The paper also describes universal component-based representation, equations collecting algorithm and proper component and model editor tool. The tool is developed as a part of intelligent conceptual design system.

Acknowledgement

Presented work was supported by Czech Research Grant Agency project, contract No. GACR 102/01/0763.

References

[1] Brandejský T.: "Architecture of Simulation Tool for Conceptual/Functional Design." Kerchkhoffs E. J. H. a Snorek M.(eds): *Modelling and Simulation 2001*. 15th European Simulation Multiconference 2001, June 6–9, Prague, SCS San Diego, CA, USA (2001), p. 390–3.

[2] Brandejský T.: *Transformation of UML into Bylander qualitative description of systems*. Research report 13/2002. CTU, Faculty of Transportation Sciences, Prague, 2002.

[3] Brandejský T.: "Integrated fuzzy-numerical models for conceptual design." Kerchkhoffs E. J. H. a Snorek M.(eds): *Modelling and Simulation 2001*. 15th European Simulation Multiconference 2001, June 6–9, Prague, SCS San Diego, CA, USA (2001), 387–9.

[4] Törnig W.: *Numerische Mathematik für Ingenieure und Physiker. Band I.: Numerischen Methoden der Algebra. Band II: Eigenwertprobleme und Numerische Methoden der Analysis*. Berlin-Heidelberg-New York, Springer-Verlang, 1979.

[5] Pedrycz W.: "Granular Computing for System Modelling." Szczerbinska H. (ed.): *Modelling and Simulation: A Tool for The Next Millenium*. SCS, Warszaw, Poland (1999), p. 391–394.

[6] Brandejský T., Bíla J., Brož K.: "Fuzzy Qualitative Modelling of Distributed Energy and Heat Supply Complex." In: Szczerbinska H. (ed.): *Modelling and Simulation: A Tool for The Next Millenium*. SCS, Warszaw, Poland (1999), p. 391–394.

[7] Brandejský T.: *Methods of Building Membership Interval Logic Models*. Mendel 2000 conf., Brno University of Technology, (2000), p. 238–242.

[8] Brandejsky T.: *Membership Interval Fuzzy Logic Reasoning*. Proceeding east-west fuzzy colloquium 2000. 8th Zittau Fuzzy Colloquium, September 6–8, (2000), p. 36–41.

[9] Kosko B.: *Fuzzy systems as universal approximators*. Proceedings IEEE International Conference on Fuzzy Systems, IEEE (1992), p. 1153–1162.

[10] Setnes M.: *Fuzzy rule-base simplification using similarity measures.* Delft university, M. Sc thesis (1995).

Doc. Dr. Ing. Tomáš Brandejský
phone: +420 224 359 530
e-mail: brandejsky@fd.cvut.cz

Department of Telecommunications and Informatics

Czech Technical University in Prague
Faculty of Transportation Sciences
Konviktská 20
110 00 Prague 1, Czech Republic

