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

FROM CREATIVE IDEAS TO OPTIMIZED CONCEPTS AND BACK A METHOD FOR COLLABORATIVE CREATION OF SOLUTION ALTERNATIVES IN DECISION SUPPORT SYSTEMS

Kees van Overveld, Maxim Ivashkov

Abstract

We consider possibilities to automate the process of generation and evaluation of conceptual solutions by Decision Support Systems (DSS) in the context of ill-defined design problems. In early phases of the solution process, each conceptual solution is not fixed yet but can be configured in multiple ways. Therefore, the application of DSS with respect to the generation of solutions requires the presence of the following knowledge: the set of alternative solutions and the set of design decisions, which are necessary for the solution configuration. However, in the context of ill-defined design problems it is difficult to obtain the above sets beforehand.

With respect to the evaluation of solutions, active usage of DSS requires the presence of the set of computational objectives, which have to rely on computational models. Such models are supposed to relate the set of design decisions and the set of objectives, for the sake of automated evaluation. However, it may be problematic to obtain ready to use models due to a lack of structure in ill-defined problems [1].

We suggest a method called the CCC method (Collaborative Concept Creation method) for systematic development and update of the sets mentioned above and gradual build up of computational models by means of collaboration between a designer and a computer.

Keywords: Generation and evaluation of solution alternatives, decision support systems, human-computer collaboration, attributes, user input, spreadsheet

1 Introduction

In order for a DSS to be used, we need the following sets, which we will refer to as DSS sets:

- a set ***S*** of solutions to a problem,
- a set ***D*** of parameterized design decisions,
- a set ***O*** of objectives.

Using a DSS in the context of ill-defined design problems is limited due to the difficulty to obtain DSS sets beforehand [2]. In this paper we propose a method to obtain and update the above sets systematically. This method aims to improve completeness of the above sets by means of a systematic procedure. Other issues addressed in the paper are automated evaluation of multiple computational and non-computational objectives.

In previous approaches the problem of making objectives computational has been addressed in two major ways: either in a computer oriented way or in a human oriented way. In a computer oriented way the objectives can be computed for some situations by using various computational techniques such as

- Algorithmic methods resulting from physics, economy, engineering,
- Artificial intelligence techniques [3],
- Hybrid methods, that combine several techniques using fuzzy logic, neural nets etc. [4].

In a human oriented way, the objectives are evaluated by means of analytical methods such as the Analytical Hierarchy Process [5], Utility models, Weighted Sum Method, etc. We claim that in the context of ill-defined design problems it would be good to make a merge between a computer oriented way and a human oriented way.

The computer-oriented approach requires the presence of substantial explicit knowledge about the discipline domain of a problem. As a consequence, it is problematic to apply the mentioned computational techniques for evaluation of the conceptual solutions, which can differ in underlying technical principles and thus belong to different disciplines. On the other hand, a human oriented way suffers from human biases, subjectivity and the lack of explicitation of expertise and intuition. It is characteristic in the later category that human experts are interrogated for their opinions, possibly founded on expertise and intuition, rather then explicit models. Due to implicit nature of experts' knowledge, the repeatability and reliability of the results of the evaluation can be questionable.

We suggest a generic method, which we refer to as the CCC method (Collaborative Concept Creation method). The word "collaboration" here refers to collaboration between human interpretation and evaluation and computer optimisation. The CCC method enables smooth human-computer collaboration, which should lead to gradual and systematic development of DSS sets on the one hand, and provide interaction at necessary moments between the designer and the computer during the computer driven evaluation and optimisation on the other hand. This should enable integration of a computer oriented way and a human oriented way of dealing with objectives.

2 The CCC METHOD

The primary focus of this article is to introduce a method for concept creation process using smooth collaboration. The term "method" indicates that there is a number of optimisation steps to be taken. These steps may follow in not necessary sequential order. Each step describes rather an update of DSS sets throughout the design process: from early conceptual solutions to detailed, quantitative and optimised solutions.

We adopt here an analytical model for design processes introduced earlier in [6]. This model aims to describe the knowledge build up during the design process. It uses three terms, namely "concept", "attribute" and "constraint". This model is used to relate updates of DSS sets with the design process. At each step of the design process a new concept, a new attribute or a new constraint is added into the current state of the process.

Using this approach, DSS sets and their updates will be formally defined. Such a formal approach has two major advantages: on the one hand it brings a discipline into designers' work and makes it more systematic, on the other hand a formally defined method allows

computational support by means of a software tool. In the section 3 we introduce the software tool ACCEL, which provides such support.

2.1 The set of conceptual solutions

The generation process starts with an initial set of ideas for the solution, typically obtained from a brainstorm (or other creativity techniques such as brain-writing, lateral thinking [7], TRIZ [8], etc.). The space $\boldsymbol{S}=\{s_i\}$ is the space of all solutions to a problem, which were produced so far. Solutions in $\boldsymbol{S}$ are samples from the larger conceptual set $\boldsymbol{C}$, which is the set of all concepts in the problem context. Notice that sets $\boldsymbol{S}$ and $\boldsymbol{C}$ are *unstructured* and that $\boldsymbol{S}\subseteq\boldsymbol{C}$. If two teams would work on the same problem, the set $\boldsymbol{C}$ is the same, but both teams would build different subsets $\boldsymbol{S_1}$ and $\boldsymbol{S_2}$. The word 'concept' is used in a generic way to refer to anything that is in $\boldsymbol{C}$: a solution, a solution feature, a requirement, a stakeholder etc. A concept may refer to any issue that has or needs a name and that has to be defined or distinguished.

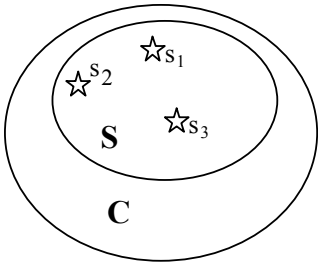

Figure 1. The set of solutions $\boldsymbol{S}$ is a subset of the conceptual set $\boldsymbol{C}$

Example 1: In order to develop a new transportation method we generate the following initial set of solutions: s_1='*freight train*', s_2='*bicycle*', s_3='*conveyor belt*'.

Each initial solution concept s_i appears first without any explicitly represented knowledge about it, but it has a clear interpretation due to the fact that people can imagine what such a concept means. In other words, there is a large amount of knowledge (world knowledge) that is implicitly packaged in each of the suggested concepts. However, any structure that may underlay this world knowledge has not been made explicit yet in the mere set $\boldsymbol{S}$. In a next step, we attempt, therefore, to make elements of this structure visible.

2.2 The set of attributes

For any concept c_i, we assume that all relevant information that is contained in c_i is accessible via a set of attributes $\{a_j\}$ that are meaningful for c_i. An attribute a_j is defined as a function, which returns a value v_{ij} in dependence of its argument, where this argument is a concept c_i; $v_{ij}=c_i.a_j$, which is equivalent to the functional notation used in math, namely $a_j(c_i)$. v_{ij} is such that it is an element of the range $\boldsymbol{R_j}$ of possible outcomes of an attribute a_j. The set of values in $\boldsymbol{R_j}$ is denoted as $\boldsymbol{v_{*j}}$. The set of values for a concept c_i is denoted as $\boldsymbol{v_{i*}}$ and is the tuple $(v_{i0},v_{i1},v_{i2},\ldots,v_{ij})$.

We distinguish between four, semantically distinct kinds of attributes: design attributes ($\boldsymbol{A_d}$), objective attributes ($\boldsymbol{A_o}$), contextual attributes ($\boldsymbol{A_c}$), and auxiliary attributes ($\boldsymbol{A_a}$). Design attributes and objective attributes are attributes of primary interest. They allow to formally define what the set of design decisions and the set of objectives are, which are considered in the sections 2.3 and 2.4 respectively. Contextual and auxiliary attributes are attributes of secondary interest, which need to be taken into account during modelling.

Design attributes

Definition: An attribute (a_j:$a_j\in\boldsymbol{A_d}$) is a design attribute if it is an independent attribute that constitutes a decision for which the designer has full authority. □

Attributes in A_d generate design alternatives. The range of a design attribute represents the set of decision options. So, ‘ *Vehicle. Fuel=ALT(Petrol;Gas)*’ has the interpretation “the vehicle shall be such that its fuel is either petrol or gas”: it allows the designer to constraint the value of the attribute *‘Fuel*()*’*.

Objective attributes

Definition: An attribute (a_j:$a_j \in A_o$) is an objective attribute if R_j allows optimisation (this mean that the type of R_j shall be ordinal). □

Optimisation can both mean: finding a minimum or a maximum, where attributes in A_d are to be varied. An objective attribute is a dependent attribute that constitutes the effectiveness (success) of the solution concept. In order to evaluate an objective attribute as a function of attributes of kind A_d and A_c we need (quantitative or at least ordinal) model. Notice that attributes of kind A_d, A_c and A_a may be either ordinals or other types.

Contextual attributes

Definition: An attribute (a_j:$a_j \in A_c$) is a contextual attribute if it is an independent attribute that constitutes a fact of the world for which the designer has no freedom to decide□.

Irrespective from the concept, contextual attributes will return a unique constraint, for instance ‘petrol.specific_heat=[2] kJ/kg K’. As opposed to A_d, attributes of kind A_c represent observations (or guesses) about existing facts in the world, which form the context for the solutions. The values v_{*i}, where $a_i \in A_c$ are the facts that are given to designers or are obtained by means of educated guesses.

Auxiliary attributes

Definition: An attribute (a_j:$a_j \in A_a$) is a auxiliary attribute if a_j depends on values of attributes of kind A_d and A_c. □

Usually auxiliary attributes play an intermediate role in order to evaluate attributes in A_o; in themselves they do not express any desired feature of the solutions.

For instance, vibrations in themselves do not add to the perceived success of a solution, but some damage can be caused by vibrations. Therefore vibration amplitude is an attribute $\in A_a$, and not $\in A_o$.

The four described types of attributes are classified in the table below. So attributes A_d and A_o are primary attributes, A_c and A_a are secondary attributes; A_d and A_o immediately relate to the solution (the set of design decisions and the set of objectives), A_c and A_a are related to the context and are only indirectly related to the solution.

Table 1. The four kinds of attributes

	Independent	Dependent
Primary	1. A_d : designers choices	3. A_o: objectives
Secondary	2. A_c: contextual world knowledge	4. A_a: auxiliary

Notice that for any two concepts c_1 and c_2, we have that

$$(\forall j{:}a_j \in A_d \cup A_c{:}\ c_1.a_j = c_2.a_j) \Rightarrow (\forall j{:}a_j \in A_o \cup A_a{:}\ c_1.a_j = c_2.a_j).$$

Using our definitions of these four kinds of attributes, we can formally define the set of design decisions and the set of objectives.

2.3 The set of design decisions

The set of design decisions $\boldsymbol{D}=\{v_{*j}:a_j \in \boldsymbol{A_d}\}$. Each design decision ($v_{ij}:v_{ij} \in \boldsymbol{D}$) is taken from the set of decision options, which are contained in the range $\boldsymbol{R_j}$ of the attribute a_j.

Design attributes allow to systematically generating new solutions by combining decision options from different design attributes. Each generated solution can be considered either as a modification of an existing solution or as a conceptually new solution. For design attributes such as material or geometry, various combinations would lead to better-optimised solutions, for instance variations of the length and the width of a box might lead to optimised volume and area of the box. For other design attributes, for instance attributes that express physical principles or technologies behind solutions, variations can lead to conceptually new solutions, which might need further interpretation and feasibility analysis.

Example 2. For the problem 'find a means for transportation' we can propose the following design attributes: a_1='*Energy source*' with the range $\boldsymbol{R_1}$={*Diesel, Kerosene, Human Power*} and a_2='*Media*' with the range $\boldsymbol{R_2}$={*air, ground, sea*}. We can see that (*Diesel; Ground*) can be easily interpreted as s_1='*Locomotive*', (*kerosene; air*) as s_2='*Airplane*'. For (*Human Power; Air*) there is no straightforward physical interpretation although it may inspire creative imagination. Indeed, it has long been thought that it would be impossible for a human being to develop enough power to fly, until a solution was actually built.

2.4 The set of objectives

The set of objectives $\boldsymbol{O}=\{v_{*j}:a_j \in \boldsymbol{A_o}\}$. The set of objective attributes is used to evaluate all solutions. For every solution s_i the evaluation results in a set of values $\boldsymbol{v_{i*}}$, such that

$$\boldsymbol{v_{i*}}=\{v_{ij} | v_{ij}=c_i.a_j,\ a_j \in \boldsymbol{A_o}\}.$$

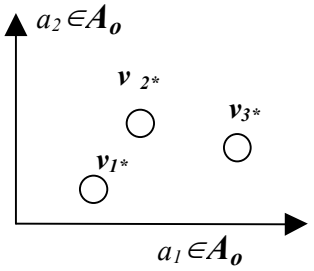

Figure 2. The set of objectives

From the definition of objective attributes it follows that for any $a_j \in \boldsymbol{A_o}$ and any solutions s_1 and s_2 there is a relation 'better then' or 'worse then' between v_{1j} and v_{2j}. A solution s_1 is better then s_2 if the relation 'better then' holds between v_{1j} and v_{2j} for all a_j: $a_j \in \mathbf{A_o}$.

The model(s) that relate values of attributes in $\boldsymbol{A_d}$ and $\boldsymbol{A_c}$ to values of attributes in $\boldsymbol{A_o}$ need to be provided by the designer. They constitute the interpretation, the physical causality or the intuition of the designer.

Models can be build or found for many design objectives but not for all. For many objectives related for instance to aesthetics, usability etc. it is problematic to obtain quantitative, computational models. Therefore we distinguish between computational and interpretational objective attributes.

Definition: an objective attribute ($a_j:a_j \in \boldsymbol{A_o}$) is a *computational objective attribute* if it is supported by a computational model.

Definition: an objective attribute ($a_j:a_j \in \boldsymbol{A_o}$) is an *interpretational objective attribute* if it is *not* supported by a computational model.

Consider a computational objective attribute '*Power*' and an interpretational objective attribute '*Safety*', which we will apply to some of the alternatives generated in example 2 (*Diesel; Ground*), (*Human Power; Ground*), (*Human Power; Air*). We assume that there are available models that relate '*Power*' to the media state and energy state. However, for '*safety*' we may not have found such computational models. Therefore, we have to rely on our own interpretation of safety for each of the solutions in particular. For instance, experts' interpretation may result in the following order: *Safety*((*Human Power; ground*)) >*Safety*((*Diesel; ground*)) > *Safety*((*Human Power; air*)). Notice that partial ordering, rather than full ordering, is sufficient for an objective attribute. Although such ordering may be subjective and not supported by a computational model, it still allows to express early intuitions, which otherwise would stay unexpressed.

2.5 The procedure of the CCC method

We can now express the procedure of the CCC method. It consists of seven phases. Although the phases are described in sequential order, in practice, the phases can be mixed. The phases are described in two ways: by a table, which describes each phase, and a transitions diagram, which explains information flows between DSS sets associated with each phase.

Table 2. The phases of the CCC method.

Phases	Description	Input	Output	Supporting techniques
1. Brainstorming	Generation of the initial set of alternative solutions and structuring it in form of a hierarchy	$\boldsymbol{C}$	$\{s_i\} \in \boldsymbol{C}$	Brainstorming, TRIZ
2. Observation	Constructing $\boldsymbol{D}$ by adding independent design attributes a_j. Obtaining $\boldsymbol{v_{ij}}$. Add at least so many attributes that all s_i can be fully distinguishable on behalf of their attribute values: no solutions s_1 and s_2 should exist anymore that have $\boldsymbol{v_{1j}=v_{2j}}$	$\boldsymbol{C}$	$\{a_j\} \in \boldsymbol{A_d}$ $\{v_{ij}\}$	TRIZ Attribute-seeking technique, creativity techniques [12]
3. Generation	New alternatives are generated by making new combinations of the elements of the various attribute ranges. This requires a computer or a lot of patience.	$(\boldsymbol{R_j}: a_j \in \boldsymbol{A_d})$	$\{v_{kj}: a_j \in \boldsymbol{A_d} \mid k>i\}$	Exhaustive or genetic generation
4. Eureka	Interpretation of newly generated solutions. This phase corresponds to the word "back" in the title of this paper.	$\{v_{kj}: a_j \in \boldsymbol{A_d}\}$	$\{s_k\} \in \boldsymbol{S}$	Expertise, imagination
5. Modelling	Listing attributes $\{a_j: a_j \in \boldsymbol{A_a} \cup \boldsymbol{A_o}\}$. Building up computational models underlying a_j	$\boldsymbol{C}$	$\{a_j\} \in \boldsymbol{A_o} \cup \boldsymbol{A_a}$	Expertise, customers wishes and requirements
6. Evaluation	Evaluate *computational* attributes $\{a_j: a_j \in \boldsymbol{A_o} \cup \boldsymbol{A_a}\}$ to s_k, compute $\{v_{kj}\}$, assign a fitness function to each v_{kj}, apply a procedure to remove non-optimal solutions from $\boldsymbol{S}$.	$\{v_{kj}: a_j \in \boldsymbol{A_d} \cup \boldsymbol{A_c}\}$, $\boldsymbol{A_o}$, $\boldsymbol{A_s}$	$\{v_{kj}: a_j \in \boldsymbol{A_o} \cup \boldsymbol{A_s}\}$ $\{s_p: p<k\}$	Genetic algorithms
7. Collaborative evaluation	The same as phase 6 but both the *computational* and the *interpretational* attributes are evaluated. Section 3.2 explains how the evaluation of interpretational objectives is supported.	$\{v_{kj}: a_j \in \boldsymbol{A_d} \cup \boldsymbol{A_c}\}$, $\boldsymbol{A_o}$, $\boldsymbol{A_a}$	$\{v_{kj}: a_j \in \boldsymbol{A_o} \cup \boldsymbol{A_a}\}$ $\{s_p: p<k\}$	Genetic algorithms in combination with 'askuser()' function

All DSS sets except the conceptual set ***C*** are assumed to be initially empty. Set ***C*** already contains the currently available world knowledge from which solution concepts and attributes are obtained (see phases 1,5). It is just not represented in any explicit format. The information flows resulting from execution of the phases are shown in Figure 3.

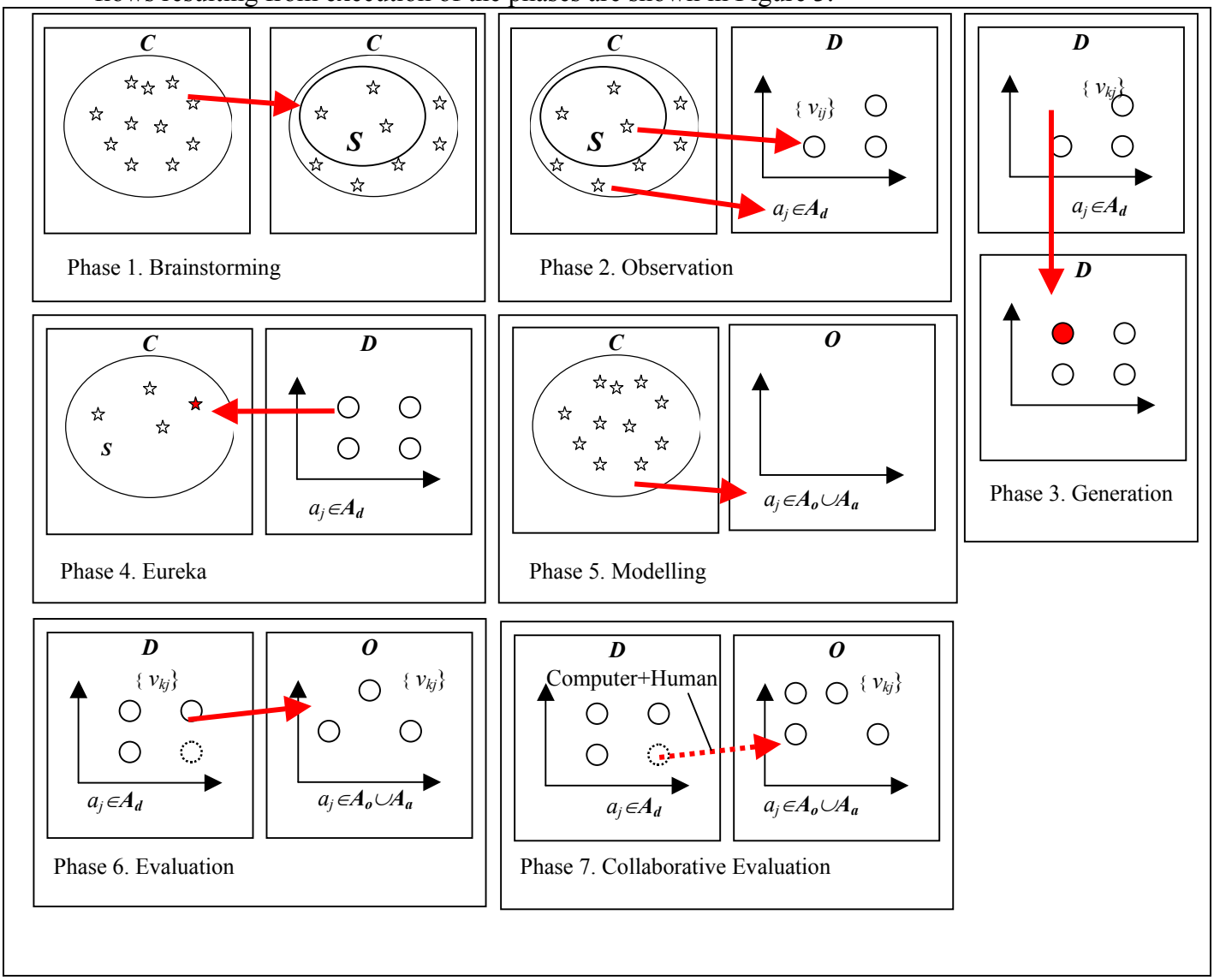

Figure 3. Information flows resulting from execution of the phases

3 Software support of the CCC method

The CCC method is supported by a software tool called ACCEL, which is abbreviated from Attributes Concepts Constraints Evaluation Language [9]. ACCEL is based on combination of notions widely used in modelling: a spreadsheet environment [10] and partial symbolic evaluation. This combination allows gradual build-up of both DSS sets and computational models.

On the basis of the spreadsheet environment, the design process as represented in ACCEL is a sequence of updates of a table. The table contains 0 or more rows; every row represents one concept. Every column in the table represents an attribute. A *cell* is characterized by a *row* (concept c_i) and by a *column* (attribute a_j). This cell represents the expression $v_{ij} = c_i.a_j$, which

is functional expression. An update of the table can be one of the following: adding a concept (=adding a row); adding an attribute (=adding a column), or *modifying* the contents of a cell (See Figure 4).

If $c_i.a_j$ can be evaluated; it also represents *the value* of this expression. It is assumed that the modification of a cell immediately invokes updating other dependent cells. Such dependencies are built during the modelling (see phase 5). If the expression in a cell cannot be completely evaluated, the user at least can see reasons for that and can correct or complete the model.

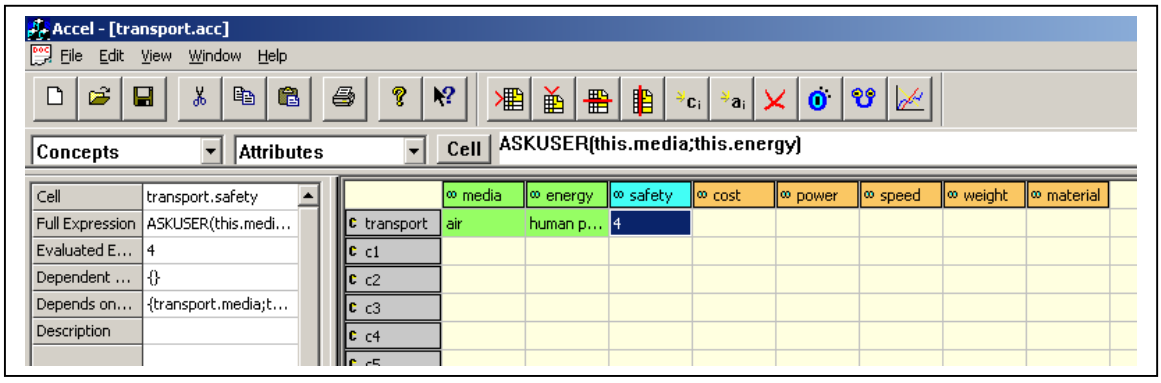

Figure 4. The representation of the design process in ACCEL.

3.1 Collaborative evaluation

During phase 7 of the CCC method user input is required in order to evaluate interpretational objectives (see 2.4). At all necessary moments the information is acquired from the user by means of an '*Askuser()*' function, which is a part of the semantics built-in ACCEL. This function is placed in a cell $c_i.a_j$, where c_i is a concept solution, which is being evaluated, a_j is an interpretational objective attribute. '*Askuser()*' has the following syntax: '*Askuser*($c_m.a_n$; $c_{m'}.a_{n'}$; $c_{m''}.a_{n''}$, ...)', where ($c_m.a_n$) is tuple of references to cells, which contain information necessary for the user to form an opinion about $c_i.a_j$. The information obtained from the user is taken according to the semantics of the expression and is kept in a history.

In order to form an opinion about $c_i.a_j$ not all information contained in c_i is necessary for the user. Often only few aspects of c_i will be sufficient for an opinion. Therefore, even if millions of different solutions have to be evaluated for a_j, there will be much less solutions, with different combinations of ($c_m.a_n$). As the result, the user input is necessary only for newly encountered combinations of ($c_m.a_n$). For all repeating combinations of ($c_m.a_n$) the information can be taken from the history. The history allows decreasing the amount of interactions with the user tremendously.

Example 3. Consider previous example 2, where we evaluated a_j=*"safety"* for various means for transportation. Each transportation mean (*tm*) might contain a large amount of information, which we can neglect during evaluations of the safety, e.i. size, material, etc. Let us assume that we are able to form a rough opinion about *tm.safety* if we are given information about energy source and the media of *tm*. In this case the cell *tm.safety* will contain an expression *"Askuser*(*tm.energy_source;tm.media*)*"* Even if there will be generated millions of solutions, the evaluation of the safety for them will require only 3*3=9 interactions with the user, since there are only 9 different combination for the given ranges $\boldsymbol{R}_{energy\_source}$ and $\boldsymbol{R}_{media}$ for which the user opinion is required. The necessary information is obtained by means of the user dialog, which is shown in Figure 5.

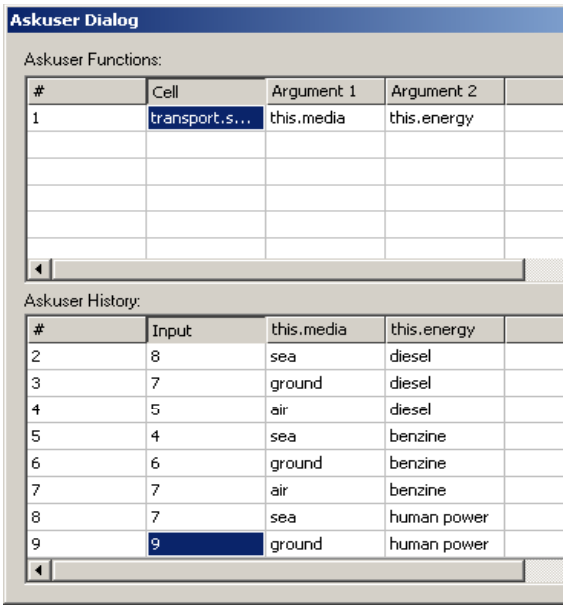

Figure 5. The Askuser Dialog contains the combinations of values for which the user input is required. The necessary user input is obtained either before or during the optimisation.

Notice that the mechanism of the '*Askuser()*' function may cause a direct inspiration for new creative ideas: the user is asked to evaluated (and hence, to interpret) a particular combination of attributes values that she at first may not have thought of. This corresponds to the words 'and back' in the title. ACCEL performs optimisation; in doing so, the '*Askuser()*' functions are evaluated, which in turn inspire to new creative ideas.

4 Summary and Conclusions

We have proposed the CCC method for collaborative generation and evaluation of solutions to ill-defined design problems. The method allows to obtain systematically DSS sets and to develop incrementally a computational model for the sake of automated evaluation and optimisation. The modelling activities associated with the method are simplified by combining the notions of spreadsheet representation and partial symbolic evaluation into a software tool ACCEL, which supports the method. ACCEL allows the user to do what-if analysis and to get optimised conceptual solutions by means of a built-in multi-objective optimisation engine. ACCEL is now publicly available and can be downloaded from our web site [9]

We have tried ACCEL in both educational and industrial settings for several projects [11]. In both cases we got different feedbacks, which were mostly positive. The present version Accel 3.1, is the result of incorporation of the feedbacks.

From our practice in industry we have experienced that people understand the method better if they work on their own problems. However, the introduction into the method and the software by means of examples easy to understand is essential.

We realise that more extended and systematic experiments have to be conducted in order to further assess practical validity and usefulness of the method and the software tool. However in this paper our main intention was to elaborate somewhat more theoretical aspects of the method, which are related to the usage of DSS in the context of ill-defined design problems.

In comparison with [6] the new ingredients of this paper are:

- Kinds for attributes,
- The collaborative procedure,
- Askuser() function.

References

[1] Cross N., "Engineering Design Methods: Strategies for Product Design", 3rd ed, 2000, pp.14-18.

[2] Keeney R. L., Raiffa H., "Decisions with multiple objectives : preferences and value tradeoffs", Cambridge University Press, 1993.

[3] Radermacher F. J., "Decision Support Systems: Scope and Potential", Decision Support Systems, Vol. 7, 1994, pp.315-328.

[4] Li S., "The Development of a Hybride Intelligent System for Developing Marketing Strategy", Decision Support Systems, Vol. 27, 2000, pp.395-409.

[5] Oeltjenbruns H. et.al., "Strategic Planing in Manufacturing Systems-AHP application to an equipment replacement decisions", Int. J. Production Economics, Vol. 38, 1995, pp.189-197.

[6] Ivashkov M. and van Overveld K., "An Operational Model for Design Processes", Proceedings of ICED '01, Vol.2, Glasgow, pp.139-146.

[7] De Bono E., "Serious creativity: using the power of lateral thinking to create new ideas", HarperCollins, London, 1993.

[8] Salamatov Y., "The Right Solution at The Right Time: A Guide to Innovative Problem Solving", Insytec B.V., 1999

[9] http://sts.bwk.tue.nl/Ivashkov/

[10] Badiru A. et.al., "A Multiattribute Spreadsheet Model for Manufacturing Technology Justification", Computers ind. Engng., Vol. 21, 1991, pp.29-33.

[11] Ivashkov M., van Overveld K., "Early Validation of a Design Method Based On Structured Reflection", Proceedings of International Design Conference Design 2002, Dubrovnik, May 14-17, 2002, Vol 1, pp. 343-348.

[12] Overveld K, et al. "Teaching Creativity in a Technological Design Context", IJEE, Vol 19-2, pp. 260-271.

For more information please contact:

M.Y. Ivashkov, M.Sc.,
Technical University of Eindhoven, F. Bouwkunde, Vert 06 H. 01,
5600 MB, Eindhoven, The Netherlands,
Tel: +31 40 2474772, e-mail: m.ivashkov@tue.nl / k.van.Overveld@wxs.nl,
URL: http://sts.bwk.tue.nl/Ivashkov/