Improvement of retrieval in Case-Based Reasoning for system design

Thierry Coudert, Élise Vareilles, Laurent Geneste, Michel Aldanondo

Improvement of retrieval in Case-Based Reasoning for system design

T. Coudert[1], E. Vareilles[2], L. Geneste[1], M. Aldanondo[2]
[1]LGP – Laboratoire Génie de Production, University of Toulouse – ENIT, Tarbes, France
[2]CGI – Centre de Génie Industriel, University of Toulouse – EMAC, Albi, France
(Thierry.Coudert@enit.fr, Elise.Vareilles@mines–albi.fr, Laurent.Geneste@enit.fr, Michel.Aldanondo@mines–albi.fr

Abstract – **The problematic addressed in this article is dealing with the improvement of retrieval in Case-Based Reasoning for system design. The retrieval activity is based on the evaluation of similarities between requirements (target) and the solutions (sources). However, similarities between features is often a subjective kind of knowledge difficult to formalize within companies. Based on an ontology of domain, the approach permits to retrieve compatible solutions rather than similar ones using a model of designer preferences. The requirements are modeled by means of constraints. When constraints are confronted to solutions in order to evaluate a compatibility measure, missing information within solutions with regard to requirements are taken into account using semantic similarities between concepts. A case study validates the proposals.**

Keywords – **Case–Based Reasoning, System design, Ontology, Preference modeling, semantic similarity**

I. INTRODUCTION

The proposals this article is dealing with focus on retrieval activity within a case–based reasoning process for system design. System design processes [1] [2] are composed of two macro-processes: requirements definition and solutions definition. Case–Based Reasoning (CBR) processes [3] [4] are mainly used as design processes. They are successful knowledge–based methods in industry. They have been widely used for system design in many domains (see e.g. [5] [6] [7] [8]) Requirements about a new system to develop are considered as a "problem" to solve and CBR systems permits to retrieve similar cases from a case base and to adapt them to provide new solutions. However, at the earliest step of design, a lot of uncertainties remain although prior suitable design solutions could be helpful. Therefore, some questions arise: how to formally define the problem for an efficient retrieval? How to define guidelines in order to define properly requirements and solutions? How to take into account the designer preferences? The retrieval step of a CBR methodology is based on the use of similarity measures between features or attributes for objects [5]. However, it is rather difficult to obtain such a subjective knowledge from experts within companies. Therefore, the problematic addressed in this article concerns the improvement of the retrieval process of CBR for systems' design. Firstly, an ontology is proposed in order to formalize knowledge and provide guidelines to designers. Secondly, the models of requirements and solutions are described. Thirdly, the retrieval process that take into account designer preferences as well as missing information within retrieved solutions is presented.

II. INFORMATION MODELING FOR SYSTEM DESIGN

A. Ontology of concepts for design

The proposed ontology gathers some pieces of knowledge by means of structured concepts. The knowledge about the domain is formalized into interrelated concepts. The aim of a concept is to guide designer teams in charge to collect requirements and finally, to develop solutions.

Thus, the knowledge embedded into a concept c is formalized by:
– a set (noted V_c) that gathers models of conceptual variables. Each one can be seen as a descriptor of the concept c. It corresponds to a general characteristic of an abstract object;
– a set (noted Σ_c) gathering models of conceptual constraints related to some models of conceptual variables. A model of conceptual constraint links one or many models of conceptual variables giving some authorized (or forbidden) combinations of values. A conceptual constraint is considered as a formal piece of knowledge embedded into a concept.

An example of ontology of concepts is represented in Fig. 1 (only concepts are represented). The proposed ontology is a hierarchical structure of concepts.

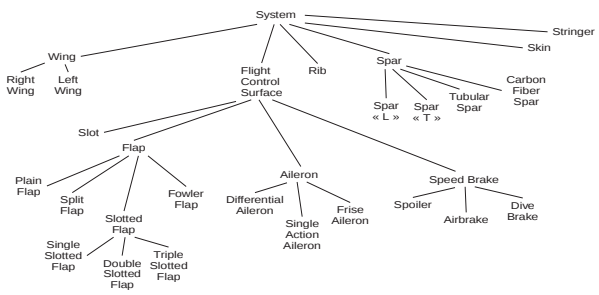

Fig. 1. Example of ontology of concepts

The root of the ontology is the most general concept named *System*. The concepts are linked by edges which represent relations of generalization/specialization. Any concept inherits all the characteristics of its parent.

B. System modeling

A System S (i.e. an object to design) is composed of a set of Requirements R and of one or many Solutions. Many solutions can be developed for a same System leading to many competitive alternatives.

1) *Requirements modeling*: associated to a system to design, the set of requirements is defined by means of: i) a Requirements Concept RC, ii) Requirements Variables associated to their domain and iii) Requirements Constraints. The Requirements Concept RC, coming from the ontology, represents the object to design at an abstract level. The different entities are represented in Fig. 2.

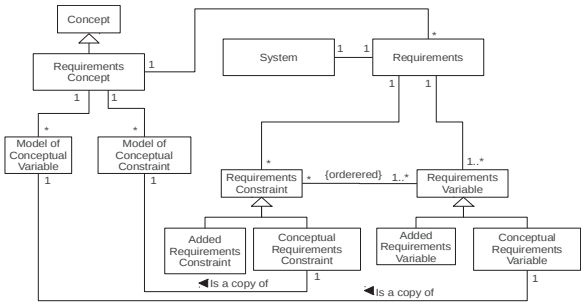

Fig.2. Requirements modeling

Therefore, by choosing a concept and associating it to the requirements (and furthermore to a system), the designer knows what are the Conceptual Requirements Variables, their domain as well as Conceptual Requirements Constraints. This information permits to guide the designer for the requirements elicitation and to formalize them by means of constraints.

2) *Solutions modeling*: a solution corresponding to a system to develop is represented by means of: i) a Solution Concept, ii) Solution Variables and their domain and, iii) values of Solution Variables (Fig. 3).

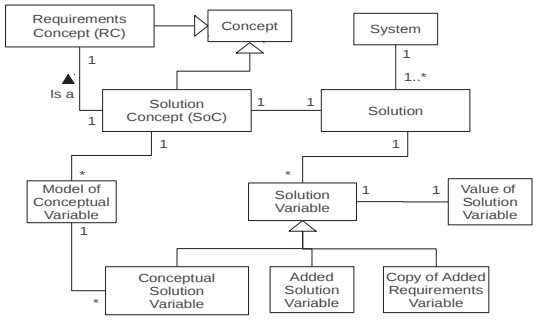

Fig. 3. Solutions modeling

A solution, within a system, is associated to a concept (SoC) which is an abstract view of the solution that must fulfill the system requirements. There is a constraint between the Requirements Concept RC and each Solution Concept SoC_i corresponding to a solution Sol_i: the concept SoC_i is either RC itself or one of its descendant into the ontology (noted: $SoC_i << RC$).

Some Solution Variables are copies of models of conceptual variables coming from SoC (they are named Conceptual Solution Variables). Added Requirements Variables are copied into each solution (they are named Copy of Added Requirements Variables). If necessary, new Solution Variables can be added by the designer to a solution in order to better characterize it (they are named Added Solution Variables). Each Solution Variable is associated to its domain (not represented).

III. REQUIREMENTS DEFINITION PROCESS: INTEGRATION OF DESIGNER PREFERENCES

From the customers or stakeholders needs, the Requirements Definition Process consists in:

- selecting the Requirements Concept (RC),
- and then, defining the Requirements Constraints.

It is considered in this article that constraints representing requirements are discrete constraints. A discrete constraint on a set of symbolic or numeric variables expressed in extension defines a set of allowed value n–tuples. Let be σ a discrete constraint which explicitly defines the allowed associations of values of a set of n discrete variables V_σ such that $V_\sigma=\{v_1, v_2, \cdots, v_n\}$. The set of domains of these n variables, noted D, is such that $D=\{D_{v1}, D_{v2}, \cdots, D_{vn}\}$. Let X_a be a set of p allowed n–tuples of values for V_σ such that $X_a=\{(x_{11}, x_{12},\cdots, x_{1n}),(x_{21}, x_{22},\cdots, x_{2n}),\cdots,(x_{p1}, x_{p2},\cdots, x_{pn})\}$ with x_{ij} a discrete symbolic or numeric value of the variable v_j in the n–tuple T_i. Let be V_σ^{value}, a n–tuple of values taken by the variables of V_σ. The crisp constraint σ is defined by $\sigma: V_\sigma^{value} \in X_a$.

Therefore, from this crisp set of allowed n–tuples, the designer can define preferences for non–allowed n–tuples. The preference function μ_σ is a mapping such that: $\mu_\sigma: D_{v1} \times D_{v2} \times \cdots \times D_{v_n} \rightarrow [0, 1]$. The steps which permit to define the preference function values are:

– Step 1: the value of μ_σ is defined for each allowed n–tuple T_i of X_a such that $\mu_\sigma(T_i)=1, \ \forall T_i \in X_a$.

– Step 2: for the non–allowed n–tuples (this set is noted X_{na} such that $X_{na}= (D_{v1} \times D_{v2} \times \cdots \times D_{v_n}) \setminus X_a$, the designer can express its preferences. A preference for a n–tuple T_j is a value between 0 and 1 representing how much the n–tuple T_j is preferred for the next retrieval step.

Similarities are not taken into account to define the function because in the case of symbolic values, it is quite difficult for experts to express a similarity measure

between two symbols because of subjectivity). It is more simple and efficient to ask to the designer how he wants to express the flexibility about a discrete constraint. The requirements remain crisp but a kind of flexibility is introduced for the next retrieval step.

Clearly, only one concept corresponding to the object to develop is required (RC). However, the designer can define a set of allowed concepts which will be used during the retrieval process. In order to define the preference function of a concept c_j with regard to RC ($\mu_{RC}(c_j)$), the semantic similarity measure proposed in [9] is used (equation 1). From this similarity measure, the designer choose whether the concept is allowed to be used during the retrieval process or not.

$$sim(c_1, c_2) = \frac{2 * depth_{System}(c_{com})}{depth_{System}(c_1) + depth_{System}(c_2)} \qquad (1)$$

The notations are:

- $depth_{System}(c)$: the distance (number of arcs) between the concept "*System*" (the root) and the concept c,
- c_{com} : the least common ancestor of RC and c_j in the ontology.

Finally, from the similarity measure, the designer defines the preference function value for the concepts ($\mu_{RC}(RC)=1$; $\mu_{RC}(c_j) = sim(RC, c_j)/c_j \neq RC$ if c_j is required for the retrieval step ; $\mu_{RC}(c_k)=0$ if c_k is not required.

III. SOLUTION DEVELOPMENT PROCESS: RETRIEVAL OF PRIOR SOLUTIONS

During the solution development process, the designer has:

- to choose into the ontology a Solution Concept (SoC) to develop,
- to define the solution assigning a value to the solution variables.

Accordingly with a Case–Based Reasoning methodology [1] [2], previously developed solutions (capitalized into a case base) can be suitable for fulfillment of new requirements. Therefore, it is necessary to retrieve similar (more exactly compatible) solutions with regard to the new requirements.

Thus, it is necessary to evaluate a compatibility measure of each preselected solution with regard to the requirements constraints. More the compatibility measure will be near to 1, more the solution is able to potentially fulfill the requirements and lower the adaptation effort will be. A compatibility measure has to be calculated for each constraint (local compatibility) and then an aggregation has to be done for the whole set of constraints (global compatibility).

1) *Local compatibility*: let σ be a constraint (involving n variables of the set V_σ) where the designer preferences have been integrated defining the preference function μ_σ. Let a solution Sol_k represented by a n–tuple of q values noted $V_{Sol_k}^{value}$ corresponding to the set of variables V_{Sol_k}. The solution Sol_k is associated to the concept SoC. The compatibility of Sol_k with regard to the constraint σ is given by the equation 2.

$$Comp(Sol_k, \sigma) = \begin{cases} 1 & \text{if } V_\sigma \subseteq V_{Sol_k} \text{ and } V_{Sol_k}^{value} \subseteq Xa \\ \mu_\sigma(V_{Sol_k}^{value}) & \text{if } V_\sigma \subseteq V_{Sol_k} \text{ and } V_{Sol_k}^{value} \not\subseteq Xa \\ \mu_{RC}(SoC) & \text{if } V_\sigma \not\subseteq V_{Sol_k} \end{cases} \qquad (2)$$

If all the variables involved into the constraint are defined into the solution and their values belong to the authorized n–tuples X_a, then the compatibility is maximum (1). If the variables values do not belong to X_a, the compatibility is given by the preference function. If at least one variable involved into the constraint is not used into the solution (missing variable), then three possibilities are offered: 1) Pessimistic way: the compatibility is equal to 0 ; 2) Optimistic way: the compatibility is equal to 1 ; 3) Our proposal: the compatibility is given by the preference function value corresponding to the concept SoC ($\mu_{RC}(SoC)$). That means that if a variable is missing into a solution with regard to a constraint, the cause is that the concept linked to the solution is far from the required concept RC. It is a compromise between pessimistic and optimistic viewpoints. The next step consists in aggregating the local compatibilities.

Global compatibility: the local compatibilities of the solution Sol_k with regard to a set of M constraints ($\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_M\}$) have to be aggregated in order to provide a global compatibility measure. Such aggregation can be done by means of a Minkowski function [10] (equation 3).

$$Comp(Sol_k, \Sigma) = \left(\sum_{i=1}^{M} (1/M) * \left(Comp(Sol_k, \sigma_i) \right)^\beta \right)^{1/\beta} \qquad (3)$$

The parameter β permits to tune the aggregation mechanism. ($\beta=1$: weighted average, $\beta \to \infty$: maximum). Then, considering the compatibility measure of each retrieved solution with regard to the whole set of constraints, the selection of compatible solutions can be done by the designer in order to decide how many solutions to develop and begin the adaptation phase.

3) *Adaptation*: The retrieved solutions can rarely be directly used as suitable solutions. The adaptation can occur by inclusion, removal, substitution or transformation of the variables values.

IV. CASE STUDY

1) *Case base content*: two solutions are capitalized into the case base: Sol_1 and Sol_2. The solution concepts are coming from the ontology in Fig. 1.

TABLE I
CASE-BASE CONTENT

SOLUTION: Sol_1	Solution Concept SoC= *"Single Slotted Flap"*			
Variables (V_{Sol_1})	length (l)	weight(wg)	material (mt)	x
Values ($V_{Sol_1}^{value}$)	1 500	100	Metal	60
SOLUTION: Sol_2	SoC= *"Differential Aileron"*			
Variables (V_{Sol_2})	length (l)	width (w)	weight (wg)	Material (mt)
Values ($V_{Sol_2}^{value}$)	1 400	75	150	Metal

2) *Customer Needs*: they can be expressed as "*The aileron length should be equal to 1000 mm and its weight should be light."*

3) *Requirements*: the choice of the concept *Aileron* gives the following knowledge (Table II).

TABLE II
CONCEPTUAL MODELS WITHIN THE *AILERON* CONCEPT (FROM THE ONTOLOGY)

Models of conceptual variables	$v_{Aileron}=\{Lenght(L), Width(W), Weight(Wg)\}$
Models of domains	$\Delta_{Aileron}=\{\{900,1000,1100,1200,1300,1400,1500\}, \{45,50,55,60,65,70,75\}, \{25,50,75,100,125,150,175,200,250,300,350,400,450,500,550\}\}$
Models of conceptual constraints	$\Sigma_{Aileron}=\{\Omega_1: L=20*W, \Omega_2: W\in\{25,50,75,100,125\}$

The choice of the Requirements Concept *Aileron* leads the designer to make copies of the conceptual variables L, W and Wg and the constraints Ω_1 and Ω_2 (copies are respectively *l*, *w, wg* and σ_1, σ_2). The need about a "light weight" leads to add the variable *mt* and to define the constraint σ_3. The need about the length (*l* = 1000) leads to add the constraint σ_4. The constraints are expressed by means of the allowed n–tuples of values X_{a1}, X_{a2}, X_{a3} and X_{a4}.

TABLE III
REQUIREMENTS CONCEPT, VARIABLES, DOMAINS AND CONSTRAINTS

Requirements Concept: RC = "*Aileron*"		
Conceptual variables + domains	length (l)	$d_l=\{900,1000,1100,1200,1300,1400,1500\}$
	width (w)	$d_w=\{45,50,55,60,65,70,75\}$
	Weight (wg)	$d_{wg}=\{25,50,75,100,125,150,175,200,250,300,350,400,450,500,550\}$
Conceptual Constraints	$\sigma_1: l=20*w \rightarrow Xa_1=\{(900,45),(1000,50),(1100,55),(1200,60),(1300,65),(1400,70),(1500,75)\}$	
	$\sigma_2: w\in[30.00,100.00] \rightarrow Xa_2=\{45,50,55,60,65,70,75\}$	
Added variables + domains	Material (mt)	$d_{mt}=\{Carbon\ Fiber, Metal\}$
Added require-ments constraints	$\sigma_3: mt\in\{Carbon\ Fiber\}=Xa_3$ $\sigma_4: l\in\{1000.00\}=Xa_4$	

4) *Designer preferences integration*: the allowed concepts with their preference degree are represented in Fig. 4. The designer chooses to extend the retrieval with the descendant concepts of RC and to add the concept "Single Slotted Flap". Their similarity with regard to "Aileron" is used to define the preferences.

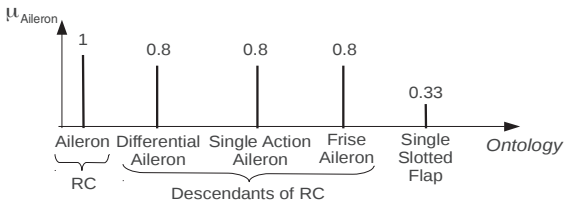

Fig. 4. Set of allowed concepts for retrieval

The allowed preferred n–tuples of values with regard to the constraints σ_1 and σ_3 are represented in the table IV with their respective preference degree.

TABLE IV
ALLOWED AND PREFERED N-TUPLES OF VALUES

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
l	900	1000	1100	1200	1300	1400	1500	1400	1300	1200
w	45	50	55	60	65	70	75	75	75	75
$\mu_{\sigma_1}(T_i)$	1	1	1	1	1	1	1	0.8	0.7	0.5

	T_1	T_2
mt	Carbon Fiber	Metal
$\mu_{\sigma_3}(T_i)$	1	0.2

The constraints σ_2 et σ_4 are defined such that:

- $\mu_{\sigma_2}(T_i)=1 \quad \forall T_i\in Xa_2; \mu_{\sigma_2}(T_i)=0$ Otherwise;
- $\mu_{\sigma_4}(T_i)=1 \quad \forall T_i\in Xa_4; \mu_{\sigma_4}(T_i)=0$ Otherwise.

5) *Selection of compatible solutions*: Sol_1 and Sol_2 are taken into account and are confronted to the constraints σ_1 to σ_4.

Compatibilities of Sol_1 with regard to the constraints: the variable w involved into the constraint σ_1 is missing within Sol_1 ($V_{Sol_1}=\{l, wg, mt\}; V_{\sigma_1}=\{l, w\}; V_{\sigma_1} \not\subset V_{Sol_1}$). Then, following the equation 2), the compatibility is $Comp(Sol_1, \sigma_1) = \mu_{Aileron}(Single\ Slotted\ Flap) = 0.33$. The constraint σ_2 involves only the variable w which is missing in the solution Sol_1: $Comp(Sol_1, \sigma_2) = 0.33$. The constraint σ_3 involves one variable mt which belongs to the solution Sol_1. Its compatibility is: $Comp(Sol_1, \sigma_3) = \mu_{\sigma_3}(metal) = 0.2$. The constraint σ_4 involves one variable l which is defined within the solution Sol_1. Its value is 1500 and the compatibility is: $Comp(Sol_1, \sigma_4) = \mu_{\sigma_4}(1500) = 0$.

Aggregation: the global compatibility of the solution Sol_1 with regards to the four constraints Σ is defined by the equation (4) with $\beta = 2$ and $M = 4$.

$$Comp(Sol_1, \Sigma) = \left(\sum_{i=0}^{4} (1/4) * \left(Comp(Sol_1, \sigma_i) \right)^2 \right)^{1/2} = \sqrt{0.25 * (0.33^2 + 0.33^2 + 0.2^2 + 0)} = 0.253 \quad (4)$$

Compatibilities of Sol_2 with regard to the constraints: Both variables l and w involved into the constraint σ_1 are defined within the solution Sol_2. The compatibility is ($Comp(Sol_2, \sigma_1) = \mu_{\sigma_1}(1400, 75) = 0.8$). The variable w takes the value 75 in Sol_2. The compatibility with regard to σ_2 is: $Comp(Sol_2, \sigma_2) = \mu_{\sigma_2}(75) = 1$. The variable mt takes the value "Metal" in Sol_2. The compatibility with regard to σ_3 is: $Comp(Sol_2, \sigma_3) = \mu_{\sigma_3}(Metal) = 0.2$. The variable l takes the value 1400 in Sol_2. The compatibility with regard to σ_4 is: $Comp(Sol_2, \sigma_4) = \mu_{\sigma_4}(1400) = 0$.

Aggregation: the global compatibility of the solution Sol_2 with regards to the whole set of flexible constraints Σ (with $\beta = 2$) is defined by the equation 5.

$$Comp(Sol_2, \Sigma) = \left(\sum_{i=0}^{4} (1/4) * \left(Comp(Sol_2, \sigma_i) \right)^2 \right)^{1/2} = \sqrt{0.25 * (0.8^2 + 1^2 + 0.2^2 + 0^2)} = 0.648 \quad (5)$$

In fine, the solution Sol_1 is not adapted with a global compatibility equal to 0.253. Furthermore, the concept "Single slotted flap" is pretty far from the requirements concept "Aileron" (similarity = 0.33). The solution Sol_2 is nearest to the requirements with a global compatibility equal to 0.648, obtained taking into account the designer preferences. Furthermore, the concept "Differential Aileron" is near to the required concept "Aileron" (similarity = 0.8). Therefore, Sol_2 is chosen by the designer for adaptation. This adaptation has to be done eliminating the non compatibilities with regard to the constraints σ_1 to σ_4.

V. CONCLUSION

The proposal described in this article deals with a retrieval mechanism based on CBR methodology for system design. It takes into account designer preferences instead of similarities to evaluate the compatibility of retrieved solutions that can be reused to fulfill new requirements. This approach is suitable because it is rather difficult to define a priori similarity measures between systems' features within companies. An ontology is used in order to capitalize knowledge about systems and to guide designers. From the ontology, it is possible to calculate automatically similarities between concepts. Such similarities are used to calculate compatibilities between solutions and requirements when some information are missing. The perspectives concern the integration of uncertainty/imprecision on solution models and to take into account such characteristics to evaluate compatibilities between solutions and requirements.

REFERENCES

[1] N.P. Suh, "The principles of design", Oxford University Press, New York, 1990.

[2] G. Pahl, W. Beitz, "Engineering Design: a Systematic Approach", Springer-Verlag, 1996.

[3] J. Kolodner, "Adaptation Methods and Strategies. Case-Based Reasoning", Chap. 11, Morgan Kaufmann, San Mateo, CA, 1993.

[4] A. Aamodt, E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches", *AI Communications*, vol. 7, no 1, pp. 39-59, 1994.

[5] Y. Avramenko, A. Kraslawski, "Similarity concept for case-based design in process engineering", *Computers and Chemical Engineering*, vol. 30, pp. 548–557, 2006.

[6] C. Gao, K. Huang, H. Chen, W. Wang, "Case–based reasoning technology based on TRIZ and generalized location pattern", *Journal of TRIZ in Engineering Design*, Vol. 2 (1), pp.40–58, 2006.

[7] C.J. Yang, J.L. Chen, "Accelerating preliminary eco-innovation design for products that integrates case-based reasoning and TRIZ method", *Journal of Cleaner Production*, vol. 19, pp. 998-1006, 2011.

[8] Y. Guo, J. Hu, Y. Peng, "A CBR system for injection mould design based on ontology: A case study," *Computer-Aided Design*, vol. 44, no. 6, pp. 496-508, June 2012.

[9] Z. Wu, M. Palmer, "Verb semantics and lexical selection," in *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–139, 1994.

[10] R. Bergmann, "Experience Management: Foundations, Development Methodology, and Internet–Based Applications," *Vol. 2432 of LNAI*, Springer, 2002.