Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2014), *28*, 49–69.

doi:10.1017/S0890060413000498

Case-based reasoning and system design: An integrated approach based on ontology and preference modeling

JUAN CAMILO ROMERO BEJARANO,[1,2] THIERRY COUDERT,[2] ELISE VAREILLES,[3] LAURENT GENESTE,[2] MICHEL ALDANONDO,[3] AND JOËL ABEILLE[2]

[1]Axsens, Toulouse, France
[2]Ecole Nationale d'Ingenieurs de Tarbes, University of Toulouse, Tarbes, France
[3]Mines-Albi, University of Toulouse, Toulouse, France

(Received September 7, 2012; Accepted July 20, 2013)

Abstract

This paper addresses the fulfillment of requirements related to case-based reasoning (CBR) processes for system design. Considering that CBR processes are well suited for problem solving, the proposed method concerns the definition of an integrated CBR process in line with system engineering principles. After the definition of the requirements that the approach has to fulfill, an ontology is defined to capitalize knowledge about the design within concepts. Based on the ontology, models are provided for requirements and solutions representation. Next, a recursive CBR process, suitable for system design, is provided. Uncertainty and designer preferences as well as ontological guidelines are considered during the requirements definition, the compatible cases retrieval, and the solution definition steps. This approach is designed to give flexibility within the CBR process as well as to provide guidelines to the designer. Such questions as the following are conjointly treated: how to guide the designer to be sure that the requirements are correctly defined and suitable for the retrieval step, how to retrieve cases when there are no available similarity measures, and how to enlarge the research scope during the retrieval step to obtain a sufficient panel of solutions. Finally, an example of system engineering in the aeronautic domain illustrates the proposed method. A testbed has been developed and carried out to evaluate the performance of the retrieval algorithm and a software prototype has been developed in order to test the approach. The outcome of this work is a recursive CBR process suitable to engineering design and compatible with standards. Requirements are modeled by means of flexible constraints, where the designer preferences are used to express the flexibility. Similar solutions can be retrieved even if similarity measures between features are not available. Simultaneously, ontological guidelines are used to guide the process and to aid the designer to express her/his preferences.

Keywords: Case-Based Reasoning; Design; Ontology; Preferences; Retrieval; System Engineering

1. INTRODUCTION

This paper focuses on a case-based reasoning (CBR) process for system design. The analysis of experience feedback (EF) of information regarding prior projects in system design permits users to make decisions very early regarding the feasibility of a new project (Girard & Doumeingts, 2004; Kam & Fischer, 2004). In such a context, CBR processes (Kolodner, 1993; Aamodt & Plaza, 1994) are mainly used to support design processes. They are knowledge-based methods used successfully in industry (see, e.g., Althoff & Weber, 2005; Liu & Ke, 2007; Armaghan & Renaud, 2012; Gu et al., 2012).

The work proposed in this article has been done considering a broader problematic defined within the ATLAS project consortium from 2008 to 2011. The ATLAS consortium involved five French academic institutions and two enterprises, funded by the French government and supported by the world competitiveness cluster Aerospace Valley. Therefore, the kind of systems this approach is dealing with are mainly in the aeronautic domain. The global approach proposed in the ATLAS project is based on the joint realization of a system design process and the associated project of design as well as the planning of this project (see Abeille et al., 2010; Coudert, Vareilles, Aldanondo, et al., 2011; Coudert, Vareilles, Geneste, et al., 2011). Moreover, the ATLAS project highlighted requirements about EF models and tools to aid engineering design. It also led to the development of a software prototype for the testing and the validation of the proposals.

Reprint requests to: Thierry Coudert, ENIT, 47 Avenue d'azereix, 65016 Tarbes Cedex, France. E-mail: thierry.coudert@enit.fr

Taking a CBR viewpoint, requirements for a new system to be developed are considered as a *problem* to solve, and CBR systems permit users to retrieve similar cases from a case base and to adapt them to provide new solutions. However, design processes are based on sound academic models as well as standards such as EIA-632, ISO-15288, and INCOSE. First, such standards and models are not integrated with CBR principles. Second, in CBR, a new problem is solved by identifying some common features between this initial problem and some previous solved problems. Most of the time, it is difficult to formally define the problem of design without ambiguity. At the earliest step of design, many uncertainties remain, requirements are difficult to formalize, and suitable prior design solutions are difficult to identify even when they could be helpful. Furthermore, the retrieval step within standard CBR processes is based on similarity measures between feature values. That means that this information is available or that methods exist to compute the values (Bergmann, 2002). When values are symbolic, it is generally considered that experts are able to provide similarity measures, but in practice, this is very difficult and time consuming. In such a context, some questions can arise for the designer: How can suitable prior cases be identified early in the process if uncertainty remains? How can a new problem be compared with prior solutions if similarity measures are not available? How can preferences be expressed during retrieval? Are there some guidelines to properly define requirements and solutions for systems with a clear and unambiguous semantic? How can knowledge about the design activity that can be reused be capitalized? When an engineering design standard is chosen for system development, how are CBR tools used? Some questions have been answered in the literature, but as far as we know, they have been answered separately. Thus, the three macrorequirements this paper is dealing with are the following:

- to define ontological guidelines to capitalize knowledge and to aid engineers to develop new systems;
- to propose a fully integrated CBR approach for system design based on suitable engineering design standards; and
- to take into account uncertainty, a lack of similarity measures, and designer preferences during the different steps of the integrated CBR approach.

The paper is organized in four sections. In Section 2, models, standards, and academic design processes as well as CBR process are presented. Then, in the light of the bibliographic background, discussions and definitions of the three requirements that the proposed method addresses are presented. In Section 3, the models for system design knowledge representation with regard to the requirements are presented. The recursive CBR (RCBR) process is detailed in Section 4. In Section 5, an example is presented to illustrate the proposed method. The experiments on the retrieval algorithm are also presented. Conclusions and perspectives are presented in Section 6.

2. MODELS, PROCESSES, STANDARDS, AND KNOWLEDGE MANAGEMENT (KM) FOR SYSTEM DESIGN AND BACKGROUND AND REQUIREMENTS FOR CBR

2.1. Methodologies and standards for engineering design

Design can be seen as a search process (Simon, 1969) corresponding to a project that aims to create or realize a new object or transform an existing one (Huysentruyt & Chen, 2010). Design is also considered as a knowledge discovery process in which information and knowledge of diverse sources are shared and processed simultaneously by a team of designers involved in the life phases of a product (Tang, 1997; Wang et al., 2006).

There are many existing design methodologies described in the literature (see, for instance, Suh, 1990; Pahl & Beitz, 1984; Dieter, 2000; Ullman, 2003). However, all the design processes are based on the following activities: knowing and understanding the customer's requirements, defining the design problem to solve, conceptualizing the solution(s), analyzing to optimize the chosen solution, and verifying the obtained solution with respect to the requirements (Suh, 1990).

From a systems engineering viewpoint, the EIA-632 standard provides some structuring processes for system design (see Martin, 2000) suitable in the aeronautic domain. The approach is based on the following premises:

- a system is composed of one or more products that are an integrated composite of hierarchical elements that have to meet the defined stakeholder requirements; and
- the engineering of a system is achieved by applying a set of processes to each element of the system hierarchy by a multidisciplinary team of people who have the required knowledge and skills.

When a decomposition is required, requirements are decomposed as well and dedicated to each subsystem to design. For each system to design, requirements are first verified and then validated. Then, the integration of subsystems can be achieved only if requirements dedicated to each subsystem have been verified and validated. That guarantees that the design of the system meets the overall system requirements. This approach is incrementally applied in an engineering life cycle framework. To tackle the product complexity, modular product architectures are used (Huang & Kusiak, 1998). That facilitates the creation of complex product architectures by developing hierarchical subsystems that can be designed independently (Mondragon et al., 2009). As mentioned in Chandrasegaran et al. (2013), the increasing complexity in products (or systems) leads to distributed and heterogeneous collaborative design environments. System engineering processes permit management of the concurrent design activities induced by such environments. Therefore, developing one system corresponds with one system engineering process.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

To split the system into many subsystems developed by specific teams and skills leads to many system engineering subprocesses having to be carried out (Abeille et al., 2010; Coudert, Vareilles, Geneste, et al., 2011). More recent standards are broader than EIA-632, covering the entire product life cycle (see, e.g., the standard ISO-15288; ISO, 2008) or the INCOSE Systems Engineering Handbook (Haskins, 2011). Such standards are fully compatible with EIA-632 and decompose the technical process for engineering a system into subprocesses: stakeholder requirements definition, requirements analysis, architectural design, implementation, integration, verification, transition, validation, operation, maintenance, and disposal. In this study, only the subprocesses "stakeholder requirements definition" to "validation" are concerned. Then, in order to be compatible with engineering standards in this article, the system design process is decomposed into a requirements definition process and a solution definition process. The requirements definition process addresses activities that define requirements from the customer or acquirer, from stakeholders, and from technical engineers. The solution definition process gradually and recursively leads to the technical solutions. The choice of a CBR process for KM is justified in the next section.

2.2. System design and knowledge

Engineering design processes use creativity, scientific principles, technical knowledge, and experience (Chen et al., 2008). In a recent survey, Chandrasegaran et al. (2013) describe explicit and tacit knowledge. Explicit knowledge is embedded in product documents, repositories, problem solving routines, computers algorithms, and so on. Tacit knowledge is a kind of knowledge *tied to experiences, intuition, unarticulated models, or implicit rules of thumbs* and is necessary to create new value in a product. This knowledge is generally retained by actors as personal experience that has to be extracted, capitalized, and exploited (Brandt et al., 2008; Foguem et al., 2008). Knowledge capitalization attempts to reuse, in a relevant way, a given domain knowledge previously stored and modeled (Dalkir, 2005). Knowledge exploitation attempts to disseminate knowledge to support current practices and to train future practitioners.

EF is a type of KM that permits experiential knowledge or lessons learned to be applied at an operational, tactical, or strategic level (Bergmann, 2002). EF is a bottom-up approach, where knowledge is built gradually from useful cases. The gradual transformation is done in three steps. The studied event and its context are described (information level), and then the analysis and solution are capitalized (experience level). The knowledge level is reached when lessons learned, procedures, invariants, and rules are inferred from past experiences (Foguem et al., 2008). During the definition of a new experience, prior capitalized experiences are taken into account in order to be reused. Different approaches exist: experiential learning (Kolb, 1984), lessons-learned systems (Weber et al., 2001), EF loops (Faure & Bisson, 1999; Rakoto et al., 2002; Jabrouni et al., 2009, 2011), or trace-based reasoning (Cordier et al., 2009; Settouti et al., 2009). In order to support such a process, tools as CBR are suitable for aiding the definition of experiences, their capitalization, and their future reutilization. According to Kolodner (1993), "[A] case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoned." These viewpoints lead us to consider EF as a powerful approach, contributing to KM and CBR as a relevant tool to support this approach. Thus, the proposed approach is based on EF principles to manage the knowledge about design processes. CBR is chosen as a tool to benefit from prior design solutions to fulfill new requirements for a system to be designed so that it closely matches the user requirements. The standard CBR process is presented in the next section.

2.3. CBR for system design

The CBR model (Kolodner, 1993; Aamodt & Plaza, 1994) considers that the solution to the most similar prior problem, adapted if necessary to take into account differences in problem descriptions, is selected as the proposed solution to the target problem (Leake & McSherry, 2005). Therefore, this methodology attempts to solve a new problem following a process of retrieval and adaptation of previously known solutions of similar problems. Usually, CBR systems are described by a cycle with main phases (Aamodt & Plaza, 1994; Finnie & Sun, 2003): problem definition, retrieval, reuse, revision, and retention. These five phases are defined below.

- *Problem definition:* When a problem arises, it is characterized so that it can be compared to problems in the case base.
- *Retrieval:* According to the new problem, the CBR system retrieves, from a case base, previous cases that are fairly similar to the new problem. The main challenge is to define how the CBR system compares the new case with prior ones, particularly when many uncertainties remain.
- *Reuse:* Most of the time, the solution of a retrieved case that fulfills the requirements of the new problem has to be adapted. The outcome is a solved case. It is generally considered as a key activity (Policastro et al., 2006). Some rules are generally defined for adapting prior solutions to new problems (Ruet & Geneste, 2002). These rules are difficult to define and to use because of the changing context of the different problems.
- *Revision:* The solved case has to be revised and is transformed into a revised case or, in other words, a suggested solution is transformed into a confirmed solution.
- *Retention:* The CBR system can learn the new case by its incorporation into the case base.

The requirements this paper is dealing with are presented in the next section.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

2.4. CBR requirements for system design

The approach presented in this article is based on the simultaneous fulfillment of three requirements. They are defined in the light of the bibliographic background and are discussed below.

2.4.1. Requirements for integration of CBR process and system engineering process

In the past decades, many works have been dedicated to case-based design. Comprehensive reviews can be found in Maher and Gomez de Silva Garza (1997), Goel and Craw (2006), and Saridakis and Dentsoras (2007). Many approaches use standard "function-based" methodologies, combining them with CBR methods. The approach proposed by Gomez De Silva Garza and Maher (1996) mixes function–behavior–structure (see Gero, 1990) and CBR. Chen et al. (2008) used a functional requirement-based model mixed with CBR. Janthong et al. (2010) combined axiomatic design and CBR to provide an integrated methodology. Other approaches combine the TRIZ method (Altshuller, 1996), with CBR (Gao et al., 2006; Cortes-Robles et al., 2009; Yang & Chen, 2011). Computer-aided design systems can also benefit from CBR methodologies (Lee & Luo, 2002; Mileman et al., 2002; Qin & Regli, 2003; Woon et al., 2005; Mok et al., 2008; Guo et al., 2012).

Regarding the recursive aspect of the system engineering process (Section 2.1), a review of approaches based on a partonomic hierarchy of cases is presented in Maher and Gomez de Silva Garza (1997). In the works of Stahl and Bergmann (2000), a RCBR is proposed for solving a product configuration problem. The initial problem is decomposed into a hierarchy of subproblems, and the system retrieves a solution. If there are unsolved subproblems, the system is recursively carried out to find subsolutions. However, the problem structure has to be completely defined at the beginning of the process. Within computer-aided design systems, approaches for assembly retrieval take into account composite products (see a panorama in Chen et al., 2012). They are based on a rough query with vague components and main relationships. However, at the beginning of the engineering process of a system, the future structure of a system can be unknown and designed later, if necessary, by other teams of designers with their own skills. Therefore, the first requirement this paper is dealing with is expressed as follows:

Requirement R.1. The CBR process should be integrated with modular and hierarchical engineering design processes: systems should be reused, level after level, following a system engineering standard. The decomposition of a system into subsystems leading to define subrequirements should be used to retrieve similar solutions from the case base. These solutions should be adapted to fulfill the subsystems requirements. This process should be recursively carried out each time a decomposition is required. ■

2.4.2. Requirements for integration of designer's preferences within requirements definition process

When designers use a CBR approach, the problem representation can be incomplete, uncertain, imprecise (Dubois et al., 1997; Nanda et al., 2007; Chang et al., 2008), and subject to preferences (Junker & Mailharro, 2003; Benferhat et al., 2006; Domshlak et al., 2011). Many case-based design approaches are based on the assumption that experts are able to define a similarity measure between two values for a feature, an attribute, or an object. Then, aggregation mechanisms provide similarity measures between a target and a source case (Bergmann, 2002). However, in an industrial context, experts encounter difficulties in defining such similarities because of the subjective nature of this task (Sun et al., 2008). Most of the CBR approaches for design are based on the assumption that the requirements are well identified and known (an exhaustive list of feature-value pairs), but most of the time, this situation is not realistic (Xuanyuan et al., 2011). During the requirements definition phase, some features are not yet identified and requirements can be ill structured or partially defined. They are refined through the design process as the designer's understanding of the problem is improved (Gomez De Silva Garza & Maher, 1996). In such a context, fuzzy CBR approaches can be helpful (see, e.g., Dubois et al., 1997, 1998; Ruet & Geneste, 2002; Sun et al., 2008; Wu et al., 2008; Zarandi et al., 2011). The fuzzy set theory is used to model uncertainties or preferences on attributes values and/or fuzzy similarity measures (see, e.g., Wang, 1997; Liu, 2005; Avramenko & Kraslawski, 2006; Negny & Le-Lann, 2008; Negny et al., 2010). However, uncertainty, preference, or similarity has to be modeled by the designer for each attribute describing the problem.

In system engineering, requirements represent the boundaries of the problem to solve or the goals that should be reached by the solution. In this paper, they are modeled by means of a set of constraints (e.g., such models are described in Thornton, 1996; Chenouard et al., 2009). A constraint describes the allowed or forbidden values of a set of variables. The requirements have to be very close to the customer needs, but if they are too crisp, some suitable cases can be mismatched during the retrieval step. Therefore, designers in charge of requirements definition should be able to introduce flexibility to retrieve a sufficient number of diverse cases for reuse. This flexibility can be expressed using flexible constraints (Dubois et al., 1996) rather than on the attributes values that define the problem. That permits the designer to: take into account the unavailability of similarity measures between attributes values, efficiently express her/his preferences and model uncertainties directly on the constraints that model the requirements, and to enlarge the scope of the retrieval. Therefore, the second requirement is defined as follows:

Requirement R.2. To take into account uncertainty and the unavailability of similarity measures, and to enlarge the scope of retrieval, flexible constraints defined from the customer's needs and the designer's preferences should be used in order

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

to model the requirements. Compatibility measures between these flexible constraints and solutions should be computed in order to guide the designer to retrieve similar and suitable solutions. ■

2.4.3. Requirement for the use of an ontology

For a panorama of the use of ontology in design, the reader can refer to Uschold and Gruninger (1996), Kim et al. (2006), and Brandt et al. (2008). The explicit specification of a conceptualization is called an "ontology," that is, a consensually determined, structured set of terms ("concepts") shared by a community of experts to express semantic information (Studer et al., 1998). An ontology may provide a formal semantic representation of the objects for case representation in CBR methodologies as in Lau et al. (2009). To be shared and understood by different actors, requirements and solutions should manipulate some standard concepts with a common and unambiguous understanding. Therefore, ontologies can be used to capitalize such knowledge and guide the requirements definition task and the solution definition task. Thus, defining a new case can be performed with less ambiguity using such an ontology for supporting requirements definition (Darlington & Culley, 2008), for preferences modeling as in Cao et al. (2011), and for solutions modeling. Therefore, a concept should be associated to the requirements and a concept should be associated to each solution within a case.

In order to take into account the differences between the requirements and a solution during retrieval (number and type of features, conceptual gap, etc.) the similarity should be first evaluated at the conceptual level, comparing both concepts. There are many efficient methods to evaluate the similarity between two concepts described in the literature (Wu & Palmer, 1994; Cordi et al., 2005; Batet et al., 2011). Based on such similarity measures, the designer should be able to express her/his preferences on the concepts. Therefore, only the solutions with a sufficient conceptual similarity and/or preference should be selected for a deeper comparison. The third requirement is then expressed as follows:

REQUIREMENT R.3. The CBR process should be based on an ontology to assist the requirements definition process, facilitate the standardization and reusability for solution development processes, and facilitate the retrieval step. Associating concepts to the requirements and to the solutions should be done in order to use the knowledge embedded in these concepts during the design process. The retrieval step should be split in two consecutive phases: the selection of solutions based on conceptual similarity measures and/or preferences, and the deep comparison between requirements and the preselected solutions. ■

As far as we know, there is no approach in the literature that fulfills simultaneously these three requirements in the domain of case-based system engineering. The next sections aim at fulfilling them. The knowledge and information formalisms used for system design are presented in the next section.

3. SYSTEM KNOWLEDGE REPRESENTATION BASED ON AN ONTOLOGY OF CONCEPTS

3.1. Proposed ontology

For the proposed method, a concept is a part of abstract knowledge suitable for design. A concept represents general characteristics about an object to be designed at a very abstract level. The purpose of a concept is to guide design teams in collecting requirements and, ultimately, to develop solutions. By offering a panel of well-structured and unambiguous concepts with a clear semantic within an ontology, the work of designers is improved. Designers are informed about descriptors of objects they have to design and of allowed and forbidden changes to the object characteristics.

Thus, the knowledge embedded into a concept c is formalized by

- a set (denoted by v_c) of models of conceptual variables. Each model of variable can be considered as a descriptor of the concept c. It corresponds to a general characteristic of an abstract object.
- a set (denoted by Δ_c) of models of domains (one model of domain for each model of a conceptual variable). A model of a domain represents the authorized values of a model of a conceptual variable.
- a set (denoted by Σ_c) of models of conceptual constraints related to some models of conceptual variables. A model of a conceptual constraint is a formal piece of knowledge that links one or many models of conceptual variables giving some authorized (or forbidden) combinations of values.

The class diagram representing a concept and its characteristics is represented in Figure 1.

The proposed ontology is a hierarchical structure of concepts representing a taxonomy. The root of the ontology is the most general concept, named the *System*. The *System* concept has no parents. The concepts are linked by edges that represent relations of generalization/specialization. Any concept inherits all the characteristics of its parents. Some models are inherited from the ancestors and other ones are specific.

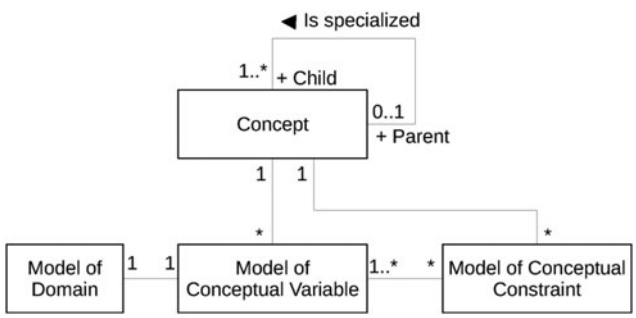


Fig. 1. A class diagram representing the characteristics of a concept in the ontology.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

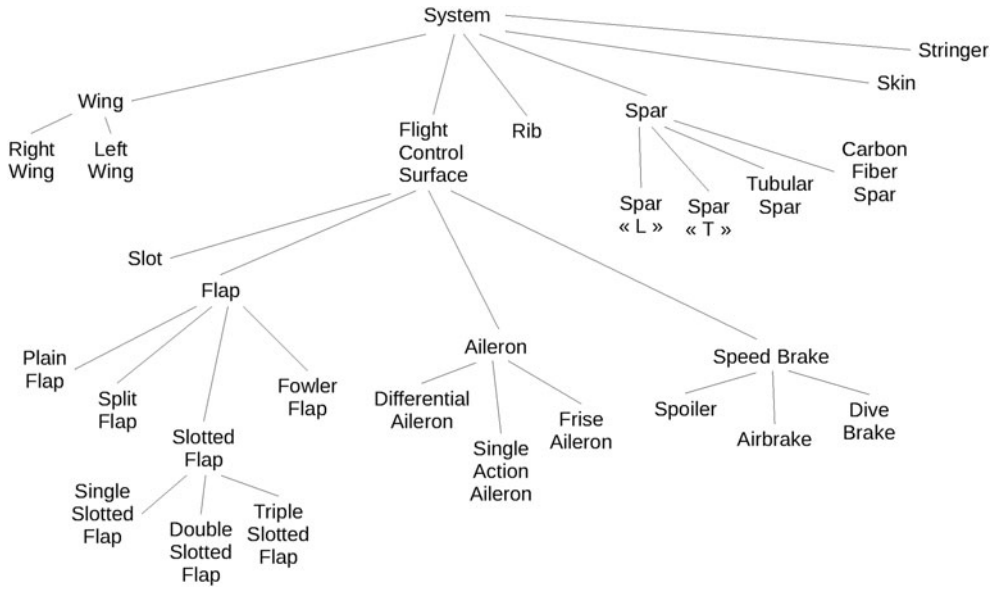


Fig. 2. An example of an ontology of concepts.

Let C_1 and C_2 be two concepts of the ontology such that C_2 is a descendent of C_1. All the models of variables, domains, and constraints of C_1 are duplicated into C_2. Furthermore, C_2 can contain specific models.

It is important to notice that the relations between concepts are not represented (except the hierarchical links). However, because the knowledge is embedded in the concepts (models of variables, domains, and constraints), the term *ontology* is maintained. An example of a partial ontology of concepts is represented in Figure 2.

An ontology has to be built and maintained by experts of the domain. The reader can refer to Uschold and Gruninger (1996), Richards and Simoff (2001), and Darlington and Culley (2008) for an overview of such a task. This aspect is not described in this paper, and it is assumed that a KM process permits designers to define, maintain, and use the ontology. For the proposed method, a concept is used to guide the definition of requirements (Section 4.2.1), to permit the retrieval of compatible solutions (Section 4.2.2), and to guide the definition of solutions (Section 4.2.3).

3.2. System modeling

Derived from EIA-632 standards, the following model is used to define the required entities of design (Fig. 3). A *System* is composed of a set of *Requirements* and of one or many *Solutions*. Many *Solutions* can be developed for the same *System*, leading to many competitive alternatives.

A *Solution* can be composed of two or more subsystems. If a system is too complex to be designed without decomposition (the problem of capacities or competencies of the design team, for instance), it is split into as many subsystems as required. For each subsystem, the requirements have to be defined from higher level requirements. A hierarchical and composite solution is then obtained by an activity of integration of subsystems or, more exactly, of solutions corresponding to the subsystems. The descriptions of the requirements and the solutions are refined in the next sections.

3.2.1. Requirements modeling

For a system to be designed, the set of requirements is defined by means of (1) a requirements concept (RC), (2) requirements variables associated with their domain, and (3) requirements constraints. The RC represents the object to be designed at an abstract level. Requirements variables are either copies of models of variables coming from the RC (named conceptual requirements variables) or new requirements variables added by the designer to better characterize the requirements (named added requirements variables). Variables domains are either copies of models of domains (named conceptual domains) or domains of new added requirements variables (named added domain). The requirements constraints are either copies of models of conceptual constraints (named conceptual requirements constraints) or new requirements constraints added by the designer to represent a specific customer need (named added requirements constraints). A re-

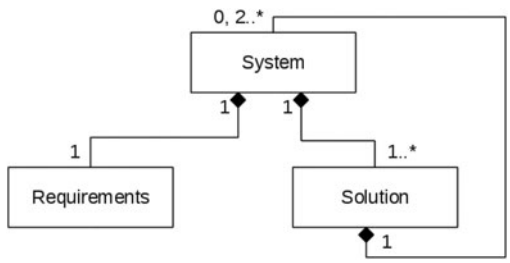


Fig. 3. The metamodel of the design entities.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

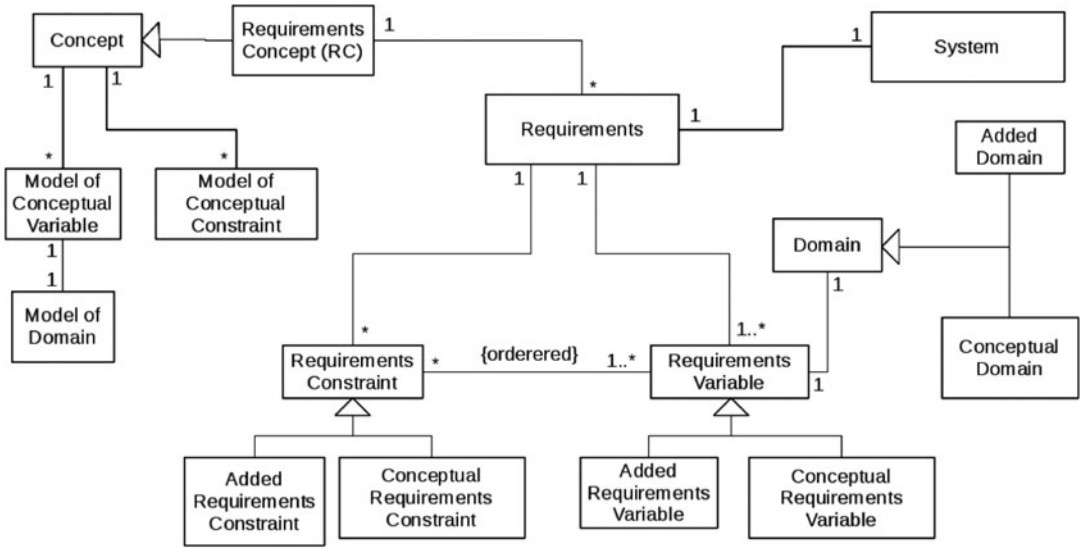


Fig. 4. The model of requirements.

quirements constraint is associated with one or many ordered requirements variables, and a requirements variable can be involved in many requirements constraints. The different entities are represented in the class diagram in Figure 4.

Therefore, by choosing a concept and associating it with the requirements (and, furthermore, to a system), the designer knows what are the conceptual requirements variables and their domain as well as the conceptual requirements constraints. This information guides the designer in eliciting the requirements and in formalizing them by means of constraints. Because the characteristics of a requirements concept are generic and abstract, the designer can add other requirements variables and constraints to properly define the system requirements. Based on the ontology, this requirements model fulfills the requirement R.3 (Section 2.4). The description of solutions is given in the following section.

3.3. Solutions modeling

A solution corresponding to a system to be developed is represented by means of a solution soncept, solution variables and their domains, solution constraints to be satisfied, and values of solution variables. The different entities are represented in the class diagram in Figure 5.

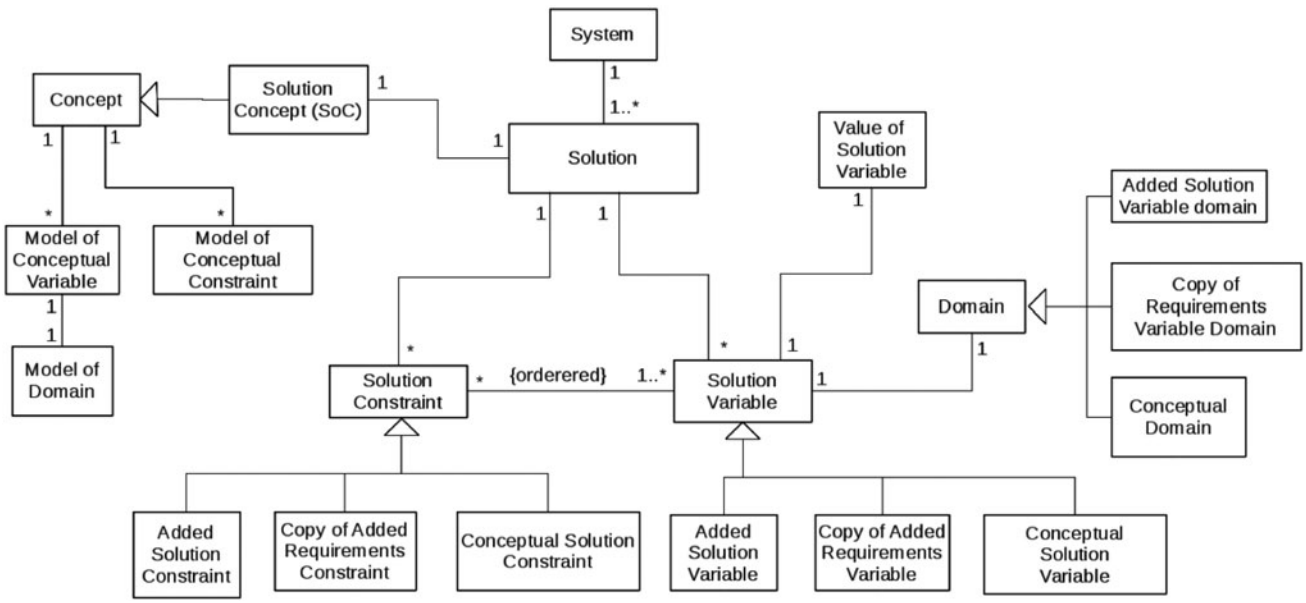


Fig. 5. The solution model.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

1. *Solution concept:* A solution, within a system, is associated with a concept, named the solution concept (SoC), which is an abstract view of the solution. Each solution is associated with its own solution concept. There is a constraint between the RC and each solution concept SoC_i corresponding to a solution Sol_i : the concept SoC_i is either the concept RC itself or one of its descendents into the ontology (denoted by $SoC_i \ll$ RC in this paper). However, SoC can be the same as RC if it is not possible to find the required specialized concept in the ontology.
2. *Solution variables:* Some solution variables are copies of models of conceptual variables coming from SoC (conceptual solution variables). Added requirements variables are copied into each solution (copy of added requirements variables). If necessary, new solution variables (associated with their domain) can be added by the designer to a solution to better characterize it (added solution variables). Therefore, within a solution, the set of solution variables contains copies of all requirements variables used to characterize the requirements and variables specific to the solution (copies of models of conceptual variables that are specific to SoC and added solution variables).
3. *Solution constraints:* Within a solution, the set of solution constraints contains the copies of models of conceptual constraints coming from SoC (conceptual solution constraints), the copies of added requirements constraints, and the added solution constraints. A constraint can be added by the designer in a solution when it is derived from the solution itself. For instance, the choice of a material can impose a new constraint on the solution dimensions. This new constraint is derived from the design solution. Therefore, a solution contains the copies of all the requirements constraints and the constraints that are specific to the solution. Some constraints are conceptual ones, and other ones are added by the designer. By embedding the entire set of constraints into a solution, it is possible to develop a solution that will ensure all these constraints are satisfied.
4. *Values of solution variables:* Within a solution, during the design process, the designer has to determine for each solution variable a value that must belong to its domain and satisfy all the solution constraints (thus, the requirements constraints). Therefore, an n-tuple of values represents the solution. To validate it, the n-tuple of values must satisfy all the constraints.

3.4. Case modeling

The results of each system development activity have to be capitalized in a case base for further reuse. Systems can be decomposed into subsystems during the solution development. Therefore, the structure of cases is also hierarchical as proposed in Macedo and Cardoso (1998). The cases are nested as well as systems. A case gathers the information about the system, its set of requirements, and its set of solutions. Thus, if a solution to reuse is composed of many subsystems, the CBR process is carried out for each subsystem in order to be sure that subcases can be reused. This point is important because, even if the complete hierarchy of systems and subsystems is reused, it ensures that, at each level, the requirements dedicated to a system are fulfilled by a solution. Even when reusing, some changes can occur in requirements following norms or stakeholders evolutions. Some subassemblies can be obsolete or their suppliers may not supply them anymore. Some adaptations then have to be made to existing solutions. An example of nested cases is given in Figure 6.

The definitions of the ontology and the models for requirements and solutions fulfill requirement R.3 (Section 2.4).

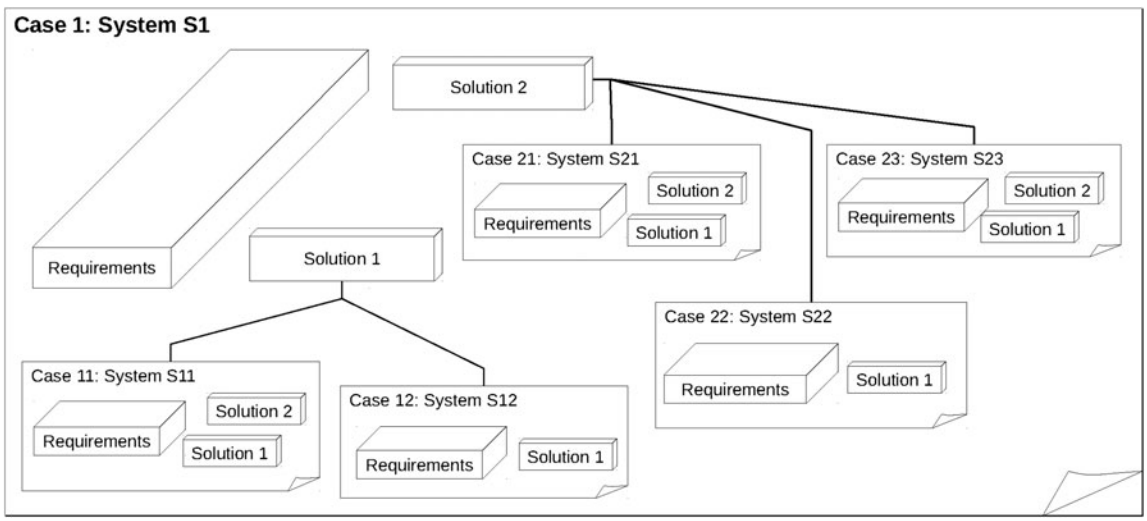


Fig. 6. An example of the two-level nested-cases model.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

4. FROM THE EIA-632 STANDARD TOWARD A RCBR PROCESS FOR SYSTEM DESIGN

4.1. RCBR process

Considering that CBR is suitable for system design, an integrated CBR process is proposed for system design activities. In Figure 7, three processes are represented: EIA-632, CBR, and the proposed RCBR process for system design. In the proposed method, the *requirements definition process* defines the target problem by means of models that represent requirements. The *compatible cases retrieval process* permits searching among the past solutions (source cases) those that are more or less compatible with the new (flexible) requirements (i.e., the target case). Then, the *solution definition process* has to be carried out with the retrieved compatible cases. This process permits the definition of (several) solution(s) from the retrieved solutions. During this process, if a decision to split the considered system into subsystems is taken because of its inherent complexity, a decomposition is performed. For each subsystem, a new RCBR process is then carried out. Once solutions are defined, the RCBR process ends by verifying, validating, and capitalizing the experiences of the development.

Considering the example of Figure 6, at the higher level, the main RCBR process develops the system S1. It carries out the RCBR process five times for the development of the subsystems (S11, S12, S21, S22, and S23). The development of a solution is split into three parts: the solution development, the subsystems development, and the integration of the subsystems. The example is summarized in Figure 8.

4.2. Detailed description of the RCBR subprocesses

4.2.1. Requirements definition process: An ontology and preference-based approach

The set of needs expressed by the customer and/or other stakeholders has to be translated into technical requirements. At the earliest stage, the RC corresponding to the future system to be developed is chosen in the ontology. The designer (or the team in charge of requirements definition) has to gather as much information as possible, taking into account the knowledge embedded in the ontology and her/his own preferences. The requirements definition process is represented in Figure 9 using the Business Process Model and Notation formalism.

The inputs of the process are either a set of customer/stakeholder needs or a set of subsystem requirements. For the latter, the requirements definition process is realized for the development of a subsystem that will be integrated into a higher level system. In that case, the designer has to elicit the requirements provided by the designer of the higher level (task T_1). The next task consists of choosing the RC among all the concepts in the ontology (this set of concepts is named C) that corresponds to the new system to design (task T_2). This choice initiates the requirements definition task (task T_3) from conceptual models. During T_3, the designer can add variables and constraints other than the conceptual ones. The outcomes of T_3 are crisp requirements, that is, a set of crisp requirements constraints (named $\sigma = \{\sigma_i\}$), and the RC (RC has been defined during T_2). The next task is the requirements constraints relaxation (task T_4). This task takes into account designer preferences and/or similarity measures coming from the ontology to provide flexible requirements constraints (such a constraint is named a *soft fuzzy constraint* in Dubois et al., 1996) and similar or preferred concepts to the retrieval mechanism.

Finally, the outcomes of the requirements definition process are the following:

- C_c: the fuzzy set of compatible concepts,
- μ_{C_c}: the membership function related to C_c, the fuzzy set of compatible concepts in C to be used during the retrieval task. The membership function $\mu_{C_c}(c)$ character-

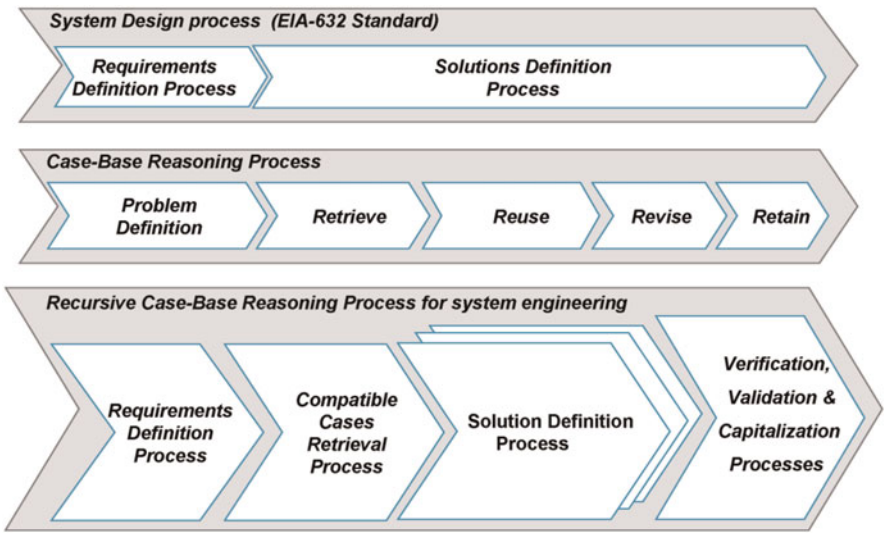


Fig. 7. The system design process, case-based reasoning (CBR) process, and recursive CBR process for design.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

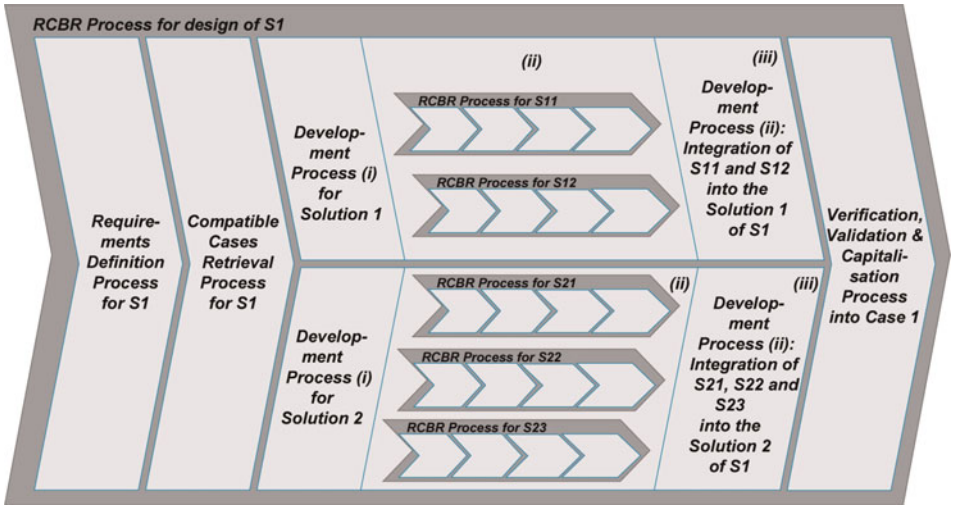


Fig. 8. An example of the two-level nested-cases model and the associated recursive case-based reasoning (RCBR) processes.

izes the degree to which a concept c of the ontology ($c \in C$) is compatible with the new requirements and such that $\mu_{C_c}: C \rightarrow [0, 1]$,

- σ: the set of crisp constraints that model the requirements,
- μ_σ: the set of membership functions related to the set of requirements constraints σ, such that $\mu_\sigma = \{\mu_{\sigma_i}\}$, where μ_{σ_i} is the membership function related to a requirements constraint σ_i. It characterizes the fuzzy set of compatible values (or n-tuples of values) taken by the requirements variables with regard to the constraint σ_i,
- $\tilde{\sigma}$: the set of flexible constraints.

Steps to evaluate the function μ_{Cc}.

1. *Step 1:* The concept RC is defined as fully compatible: $\mu_{Cc}(\text{RC}) = 1$.
2. *Step 2:* The compatibilities of the descendent concepts of RC in the ontology are computed by evaluating their membership functions from semantic similarities. Different methods to evaluate conceptual similarities between two concepts of a taxonomy or an ontology are described, for instance, in Wu and Palmer (1994), Cordi et al. (2005), and Batet et al. (2011). Among all the ap-

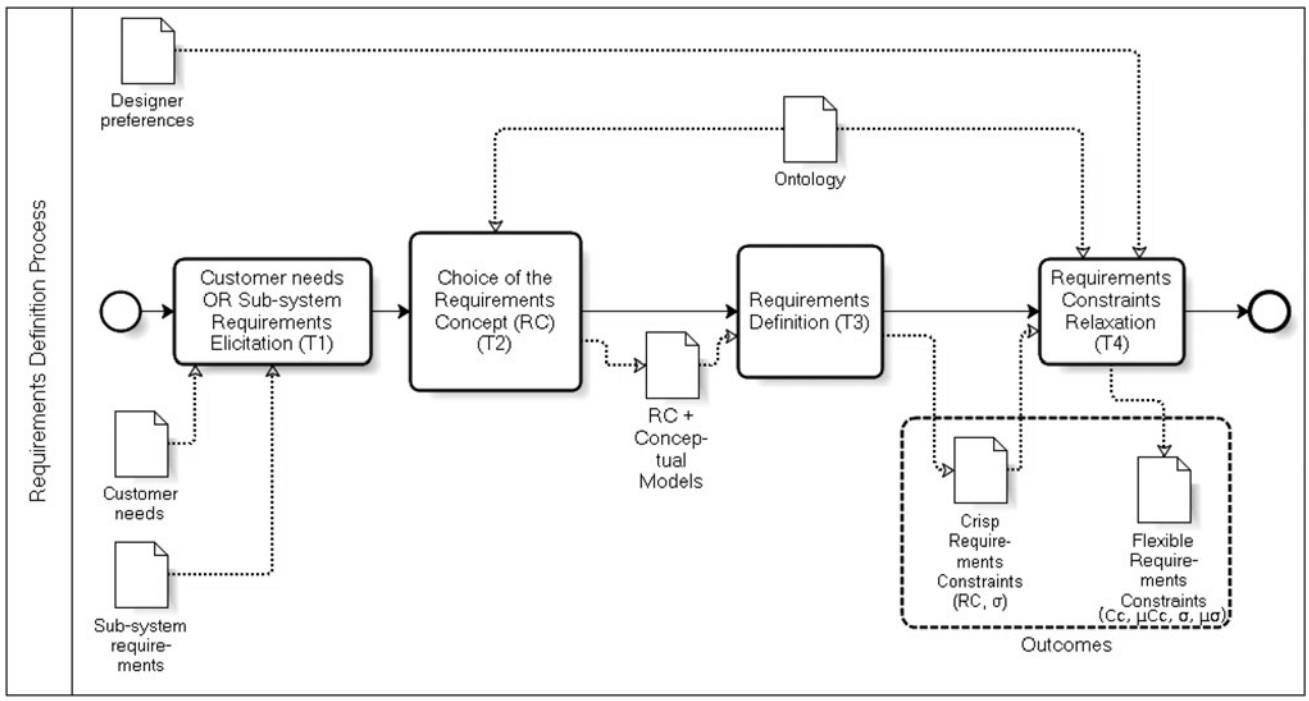


Fig. 9. The requirements definition process.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

proaches, a very simple and efficient one is the measure of Wu and Palmer, which is based on the distance (expressed in terms of the number of arcs) between the two concepts being compared and the depths of the concepts in the ontology (with regard to the *root* concept). The similarity between two concepts C_1 and C_2 is defined by Eq. (1):

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

such that

- $depth_{system}(c_i)$ is the distance (i.e., the number of arcs) between the concept named *System* and the concept c_i and
- c_{com} is the least common ancestor of c_1 and c_2 in the ontology.

The efficiency of the measure is based on a well-structured hierarchy of concepts. Therefore, the similarity measure of Wu and Palmer is used to define the membership functions of all the descendents of RC (this set is named descendents of RC, or DoRC). The membership function of each concept c_i, a descendent of RC, is defined by Eq. (2):

$$\mu_{C_c}(c_i) = sim(\text{RC}, c_i) \quad \forall c_i \in \text{DoRC}. \quad (2)$$

3. *Step 3:* For the remaining concepts of the ontology (i.e., the set named C^- such that $C^- = C\backslash(\text{RC})$, the designer has the option to express her/his preferences by setting the degree to which she/he allows each concept to be used during the retrieval step. The more a concept c_j is preferred by the designer, the closer its degree of preference, denoted by $pref(c_j)$), is to 1. The membership function of a preferred concept c_j is defined by Eq. (3):

$$\mu_{C_c}(c_j) = pref(c_j) \quad \forall c_j \in C^-. \quad (3)$$

The similarity measure of Wu and Palmer between the concept RC and the concept c_j can be used as a guideline to evaluate the preference. A similarity measure close to 1 indicates to the designer that a greater preference should be given to the concept c_j.

4. *Step 4:* For a concept $c_j (c_j \in C^-)$ with a degree of preference different from 0 and fixed by the designer, the compatibility of each descendent of C_j in the ontology (named C_{jm}, such that $C_{jm} \in \text{Doc}_j$ and $C_{jm} \notin$ (DoRC $\cup$ RC) where Doc_j is the set of *Descendent of* c_j) is computed by multiplying the similarity between c_j and c_{jm}, and the degree of preference given to c_j. The membership function corresponding to a concept c_{jm} is defined by Eq. (4):

$$\mu_{C_c}(c_{jm}) = sim(c_j, c_{jm}) \times pref(c_j), \quad \forall c_{jm} \in \text{Doc}_j, c_{jm} \notin (\text{DoRC} \cup \text{RC}). \quad (4)$$

Definition of the fuzzy set of compatible values with regard to a discrete constraint. It is considered in this article that constraints representing requirements are discrete constraints (Montanari, 1974; Gelle et al., 2000).

A discrete constraint on a set of symbolic or numeric variables defines a set of allowed n-tuples of values. Let σ_i be a discrete constraint that explicitly defines the allowed associations of values of a set of n discrete variables, denoted by V_{σ_i}, such that $V_{\sigma_i} = \{ v_1, v_2, \ldots, v_n \}$. The set of domains of these n variables, denoted by D, is such that $D = \{D_{v1}, D_{v2}, \ldots, D_{vn}\}$. Let X_a be a set of p allowed n-tuples of values for μ_{σ_i} such that $X_a = \{(x_{11}, x_{12}, \ldots, x_{1n}), (x_{21}, x_{22}, \ldots, x_{2n}), \ldots, (x_{p1}, x_{p2}, \ldots, x_{pn})\}$ with x_{ij} a discrete symbolic or numeric value of the variable v_j in the n-tuple T_i. Let $V_{\sigma_i}^{value}$ be an n-tuple of values corresponding to the variables of V_{σ_i}. The crisp constraint σ_i is defined by $\sigma_i : V_{\sigma_i}^{value} \in X_a$.

Therefore, from this crisp set of allowed n-tuples, the membership function μ_{σ_i}, which defines the fuzzy set of allowed n-tuples, is defined using the preferences of the designer. The membership function μ_{σ_i} is a mapping such that $\mu_{\sigma_i} : D_{v1} \times D_{v2} \times \cdots \times D_{vn} \rightarrow [0, 1]$. The steps to define the membership function μ_{σ_i} are the following:

1. *Step 1:* The value of the membership function $\mu_{\sigma i}$ is defined for each allowed n-tuple T_i of X_a such that $\mu_{\sigma_i}(T_i) = 1, \forall T_i \in X_a$.
2. *Step 2:* For the disallowed n-tuples (this set is denoted as X_{na} such that $X_{na} = (Dv_1 \times Dv_2 \times \cdots \times Dv_n)\backslash X_a$, the designer can express her/his preferences. A preference for a n-tuple T_j (denoted by $Pref(T_j)$) is a value between 0 and 1 representing how much the n-tuple T_j is preferred for the next retrieval step. The membership function corresponding to a disallowed compatible n-tuple T_j is defined by the Eq. (5):

$$\mu_{\sigma_i}(T_j) = Pref(T_j), \quad \forall T_j \in X_{na}. \quad (5)$$

Similarities are not taken into account to define this membership function because in the case of symbolic values, it is quite difficult for experts to express a similarity measure between two symbols. It is simpler and more efficient to ask the designer how she/he wants to express the flexibility of a discrete constraint. The preference modeling is the first step for the fulfillment of the requirement R.2 (Section 2.4).

4.2.2. *Compatible cases retrieval process*

From the compatible concepts and the flexible discrete constraints expressed during the requirements definition process, the compatible cases retrieval process can be carried out. The aim is to identify, from the case base, the compatible solutions with regard to the set of compatible concepts and with the flexible constraints. The retrieval process is composed of two sequential tasks: the preselection and the selection of compatible solutions. The compatible cases retrieval process is represented in Figure 10.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

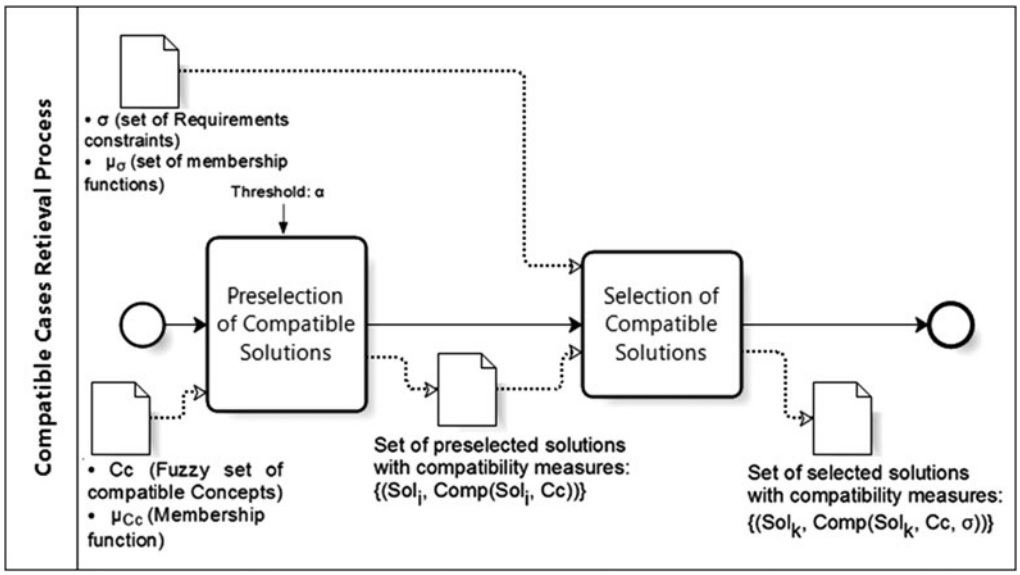


Fig. 10. The compatible cases retrieval process.

a. *The preselection of compatible solutions:* With regard to the set of compatible concepts C_c and a threshold given by the designer is a conceptual retrieval mechanism. For each solution Sol_i in the case base, the preselection process evaluates how much the Solution Concept ($Concept(Sol_i)$) linked to the solution is compatible with the set of concepts C_c. The compatibility, denoted by $Comp(Sol_i, C_c)$, is given by Eq. (6):

$$Comp(Sol_i, C_c) = \mu_{C_c}(Concept(Sol_i)). \quad (6)$$

Therefore, each solution in the case base having a compatibility greater than or equal to a threshold α ($0 \leq \alpha \leq 1$) given by the designer is preselected for the next step of retrieval and added to the set *PRE* conjointly with its compatibility measure [Eq. (7)].

$$PRE = \{(Sol_i, Comp(Sol_i, C_c))\} / Comp(Sol_i, C_c) \geq \alpha. \quad (7)$$

Then, from an n-tuple of values corresponding to a preselected solution of the set *PRE*, the retrieval mechanism has to

- evaluate the compatibility of the solution with regard to each flexible constraint (local compatibility);
- aggregate the local compatibilities in order to obtain the compatibility of the whole solution with regard to all flexible constraints (global compatibility); and
- select compatible solutions.

b. *Compatibility with regard to a flexible discrete constraint:* Let $\tilde{\sigma}_i$ be a flexible discrete constraint (on n variables of the set V_{σ_i}), where the preferences of the designer have been integrated by defining the membership function μ_{σ_i}. Let a solution Sol_k be represented by a set of q values denoted by $V_{Sol_k}^{value}$ corresponding to the set of variables V_{Sol_k}. The compatibility of Sol_k with regard to the constraint $\tilde{\sigma}_i$ is given by Eq. (8). If all the variables involved in the constraint are defined in the solution and their values belong to the authorized n-tuples X_a, then the compatibility is maximum. If the variable values do not belong to X_a, the compatibility is given by the designer's preference. If at least one variable involved in the constraint is not used in the solution, then the compatibility is equal to 0.

$$Comp(Sol_k, \tilde{\sigma}_i) = \begin{cases} 1 & \text{if } V_{\sigma_i} \subseteq V_{Sol_k} \text{ and } V_{Sol_k}^{value} \subseteq X_a, \\ \mu_{\sigma_i}(V_{Sol_k}^{value}) & \text{if } V_{\sigma_i} \subseteq V_{Sol_k} \text{ and } V_{Sol_k}^{value} \not\subseteq X_a, \\ 0 & \text{Otherwise.} \end{cases} \quad (8)$$

c. *Compatibility of a solution with regard to the whole set of flexible constraints:* The local compatibilities of the solution Sol_k with regard to a set $\tilde{\sigma}$ of M flexible constraints ($\tilde{\sigma} = \{\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_M\}$) have to be aggregated in order to provide a global compatibility measure. This aggregation can be performed, for instance, by means of a Minkowski function (Bergmann, 2002). Then, the global compatibility of the solution Sol_k with regard to the set $\tilde{\sigma}$ is given by Eq. (9) (Coudert, Vareilles, Aldanondo, et al., 2011).

$$Comp(Sol_k, \tilde{\sigma}) = \left(\sum_{i=1}^{m} (1/m) \times (Comp(Sol_k, \tilde{\sigma}_i))^{\beta} \right)^{1/\beta}. \quad (9)$$

The parameter β permits the designer to tune the aggregation mechanism ($\beta = 1$: weighted average, $\beta \to \infty$: maximum).

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

Then, considering the compatibility measure of each preselected solution with regard to the entire set of constraints, the selection of compatible solutions can be made by the designer to decide how many solutions to develop and the solution definition processes (one process for each solution to be developed) can begin. This retrieval process fulfills the requirement R.2 (Section 2.4).

4.2.3. Solution definition process

From the set of approximately compatible solutions coming from the compatible cases retrieval process, one or many solutions corresponding to the new requirements have to be developed. The retrieved solutions can rarely be directly used as suitable solutions. They usually require adaptations to be applied to new requirements. The adaptation process may be as simple as the substitution of a component. This is expressed by changing the value of the variable corresponding to the component or the feature. The adaptation can be as complex as the complete modification of the solution structure. The adaptation can occur by inclusion, removal, substitution, or transformation of the variable values (Policastro et al., 2008). Within a routine design context, the designer will make a copy of fully compatible solutions. In an innovative design context, the designer will make copies of several elements or ideas and will totally change other ones. For the latter, the proposed RCBR process is helpful because the designer knows which elements are the prior compatible solutions and which elements are the obsolete components to replace within the new solution.

a. *Description of the solution definition process:* The solution definition process is represented in Figure 11 for the development of one solution Sol_{new}. An empty solution is created. Then, from the set of retrieved solutions and their compatibility measures given by the compatible cases retrieval process, the designer selects one solution Sol_r to reuse. Three scenarios are presented.

Scenario 1: The solution has to be developed from scratch because the retrieved solutions are not considered suitable. The designer chooses a SoC in the ontology and develops the new solution. The knowledge embedded in the concept SoC is used as a guideline. This task is finished when the designer has given a value to each variable. However, two possibilities remain:

- If the designer chooses to decompose the solution into n subsystems ($n \geq 2$) because of the complexity, the entire RCBR process is carried out RCBR times for the development of each subsystem solution at the lower level. The designer has to provide the subsystem requirements dedicated to each subsystem. When developed, the subsystem solutions are integrated to finalize the system solution by giving values to the variables.
- If the designer can develop the solution without decomposition, the process is finished.

Scenario 2: One solution Sol_r is selected for reuse. Then, Sol_r is copied into Sol_{new}. Thus, the solution Sol_{new} is confronted with the set of crisp requirements constraints σ. If each requirement constraint of σ is satisfied by Sol_{new}, that means that no adaptation effort is needed and the development of the solution Sol_{new} is finished.

Scenario 3: As in scenario 2, Sol_r is copied into Sol_{new}, but the solution Sol_{new} does not satisfy the

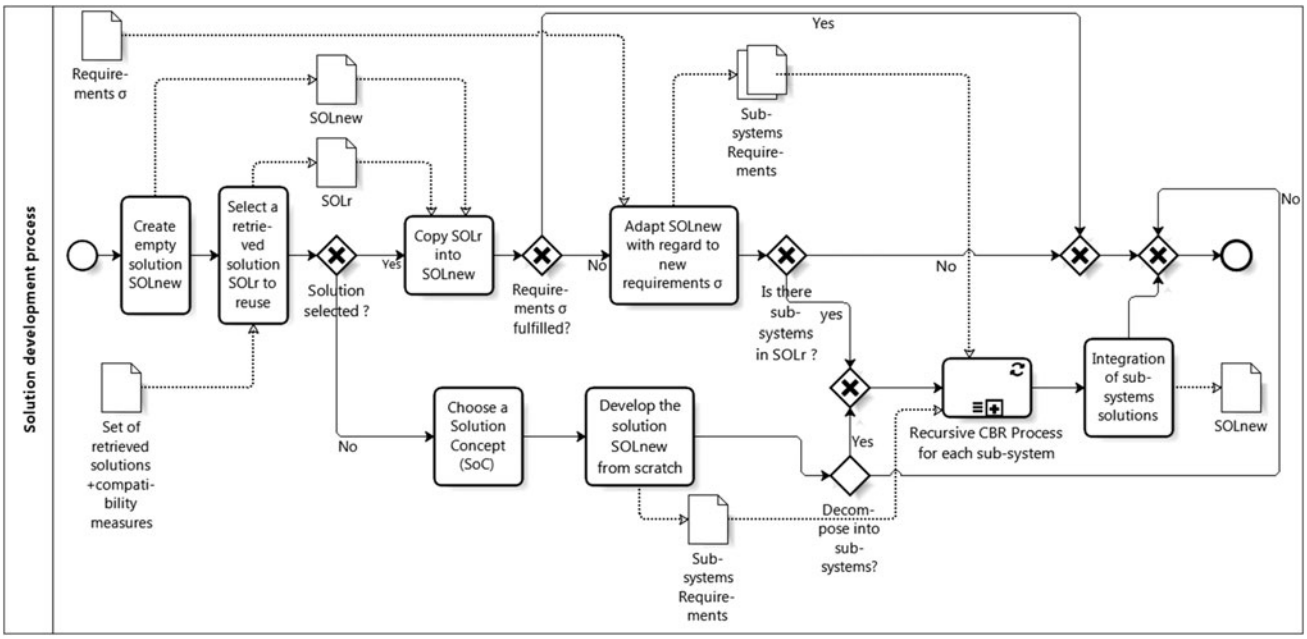


Fig. 11. The solution definition process.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

requirements constraints. Thus, the solution Sol_{new} is adapted in accordance with the new requirements. If the solution Sol_{new} is composed of subsystems, the designer has to adapt the subsystem requirements and transmit them to the lower level RCBR process to eventually reuse the subsystem solutions. When each subsystem has been developed within its own RCBR process, the subsystem solutions have to be integrated. If the solution Sol_{new} is not composed of subsystems, then the development is finished after the adaptation task.

b. *Description of the solution copy and adaptation:* When a solution Sol_r is copied into an empty solution Sol_{new}, the following information is transferred:

- the solution concept of Sol_r is copied and linked to Sol_{new},
- all solution variables (and their domain) of Sol_r are copied into Sol_{new} (added solution variables, added requirements variables, and conceptual solution variables),
- all solution constraints are copied (added solution constraints, added requirements constraints, and conceptual solution constraints).
- the change of the concept linked to the solution Sol_{new} by a close concept within the ontology (thus, conceptual variables and constraints can be added or removed), and
- the addition of variables and values.

4.2.4. *Verification, validation, and capitalization process*

The final process concerns the verification/validation of the entire set of solutions that have been developed with the goal of fulfilling the requirements. For each solution, it is necessary to evaluate the *n*-tuple of values against the requirements constraints. Each constraint must be satisfied to validate the solution. If some values violate a constraint, they have to be changed. A solution is validated when it is certain that the values satisfy the entire set of requirements constraints and, furthermore, the solution constraints (requirements constraint + added solution constraints). When a solution is validated, it can be capitalized with its case in the case base for future reuse. Note that a nonvalidated solution can also be capitalized. In another context, such a solution could be validated.

The proposed solution definition process fulfills requirement R.3. There are several important notes:

- Even if the proposed RCBR process is presented as sequential activities, each time it is possible to backtrack and modify information. Particularly during the solution definition process, new requirements can be added by the designer, or some exceptions can be required regarding a particular constraint that is impossible to satisfy. In such cases, the activities are stopped, the requirements can be modified (demand for derogation, for instance), newly retrieved solutions can be obtained from the case base, and the solution definition process can restart within its new context. Such a function is not described in this paper.
- If a suitable concept (RC or SoC) is not found in the ontology, it can be added by the designer. However, it is important to note that a KM process is then required in order to verify and validate the new concept. A number of questions have to be treated. Is it well founded? Is it placed at the right place in the ontology? Are the models within the concepts complete? However, this KM activity is not described in our approach.

5. ILLUSTRATIVE EXAMPLE, TESTS, AND DISCUSSION

First, an academic example of a system design following the RCBR process is described to highlight and validate the proposed approach. The representation highlights information and knowledge. Second, the ATLAS software used in order to implement the propositions is briefly presented. Third, the experiments performed to test the compatible cases retrieval process are presented, and fourth, a discussion about the results and the contribution is provided.

5.1. Description of the example

a. *Case base content:* To simplify, only three cases of the case base are represented for this example. Furthermore, only the solutions are represented: Sol_1, Sol_2, and Sol_3 (Table 1). The concepts are from the ontology of Figure 2 (Section 3.1).

b. *Customer needs:* The needs are expressed as "*The aileron length should be equal to 1000 mm and its weight should be light.*"

c. *Requirements:* From the customer's need, the designer chooses the concept *Aileron* in the ontology of Figure 2. The knowledge embedded into the concept *Aileron* is represented in Table 2.

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 named respectively, *l*, *w*, and *wg* for the variables and σ_1 and σ_2 for the constraints). The need for "light weight" leads to the addition of the variable *mt* and to the definition of the constraint σ_3. The need for the length ($l = 1000$) leads to the addition of the constraint σ_4. The four constraints are expressed by means of the allowed *n*-tuples of values Xa_1, Xa_2, Xa_3, and Xa_4 (Table 3).

d. *Flexible requirements for retrieval:* The flexible requirements are represented in Figure 12. The constraints $\tilde{\sigma}_2$ and $\tilde{\sigma}_4$ are defined such that $\mu_{\sigma_2}(T_i) = 1, \forall T_i \in Xa_2$; $\mu_{\sigma_2}(T_i) = 0$, *Otherwise* and $\mu_{\sigma_4}(T_i) = 1\ \forall T_i \in Xa_4$; $\mu_{\sigma_4}(T_i) = 0$, *Otherwise*.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

Table 1. *Description of three solutions*

	Solution Concept *Single Slotted Flap*				
Variables (V_{Sol_1})	Length (l)	Weight (wg)	Material (mt)	x	y
Sol_1 values ($V^{value}_{Sol_1}$)	1500	100	Metal	60	2.36
	Solution Concept *Differential Aileron*				
Variables (V_{Sol_2})	Length (l)	Width (w)	Weight (wg)	Material (mt)	α
Sol_2 values ($V^{value}_{Sol_2}$)	1400	75	150	Metal	π
	Solution Concept *Tubular Spar*				
Variables (V_{Sol_3})	Length (l)	Width (w)	Weight (wg)	Material (mt)	
Sol_3 values ($V^{value}_{Sol_3}$)	15,000	150	3000	Metal	

e. *Compatible cases retrieval process/preselection of compatible solutions:* The compatibilities of the solutions with regard to the set of compatible concepts are represented in Table 4.

Let α be the threshold such that $\alpha = 0.3$ given by the designer. Therefore, Sol_3 is not preselected because its concept (*Tubular spar*) is totally different than the requirements concept (*Aileron*).

f. *Compatible cases retrieval process/selection of compatible solutions:* Sol_1 and Sol_2 are taken into consideration and are evaluated against the flexible constraints σ_1, σ_2, σ_3, and σ_4.

Compatibilities of Sol_1 with regard to the constraints. The variable w involved in the constraint σ_1 is missing within Sol_1 ($V_{Sol_1} = \{1, wg, mt\}$; $V_{\sigma_1} = \{1, w\}$; $V_{\sigma_1} \not\subset V_{\mathrm{Sol}_1}$). Then, following Eq. (8), the compatibility is (Comp (Sol_1, $\tilde{\sigma}_1$) = 0). The constraint involves only the variable w, which is missing in the solution Sol_1: Comp (Sol_1, $\tilde{\sigma}_2$) = 0. The constraint σ_3 involves one variable *mt*, which belongs to the solution Sol_1. Its value is "metal," and the compatibility is Comp (Sol_1, $\tilde{\sigma}_3$) = $\mu_{\sigma_3}(metal) = 0.2$. The constraint σ_4 involves one variable l, which is defined within the solution Sol_1. Its value is equal to 1500, and the compatibility is Comp (Sol_1, $\tilde{\sigma}_4$) = $\mu_{\sigma_4}(1500) = 0$.

Aggregation: The global compatibility of the solution Sol_1 with regard to the entire set of flexible constraints $\tilde{\sigma}$ is defined by Eq. (10) with $\beta = 2$ and $M = 4$:

$$Comp(Sol_1, \tilde{\sigma}) = \left(\sum_{i=0}^{4}(1/4) \times (Comp(Sol_1, \tilde{\sigma}_i))^2\right)^{1/2} = \sqrt{0.25 \times (0 + 0 + 0.2^2 + 0)} = 0.1. \quad (10)$$

Table 2. *Conceptual models of* aileron

Conceptual Models Within Aileron Concept[a]	
Model of conceptual variables	$\upsilon_{\text{aileron}}$ = {Length (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}}
Models of conceptual constraints	Σ_{aileron} = {Ω_1: $L = 20 \times W$}, Ω_2: $W \in$ {25, 50, 75, 100, 125}

[a]From the antology of Figure 2.

Table 3. *Requirements for the system to develop*

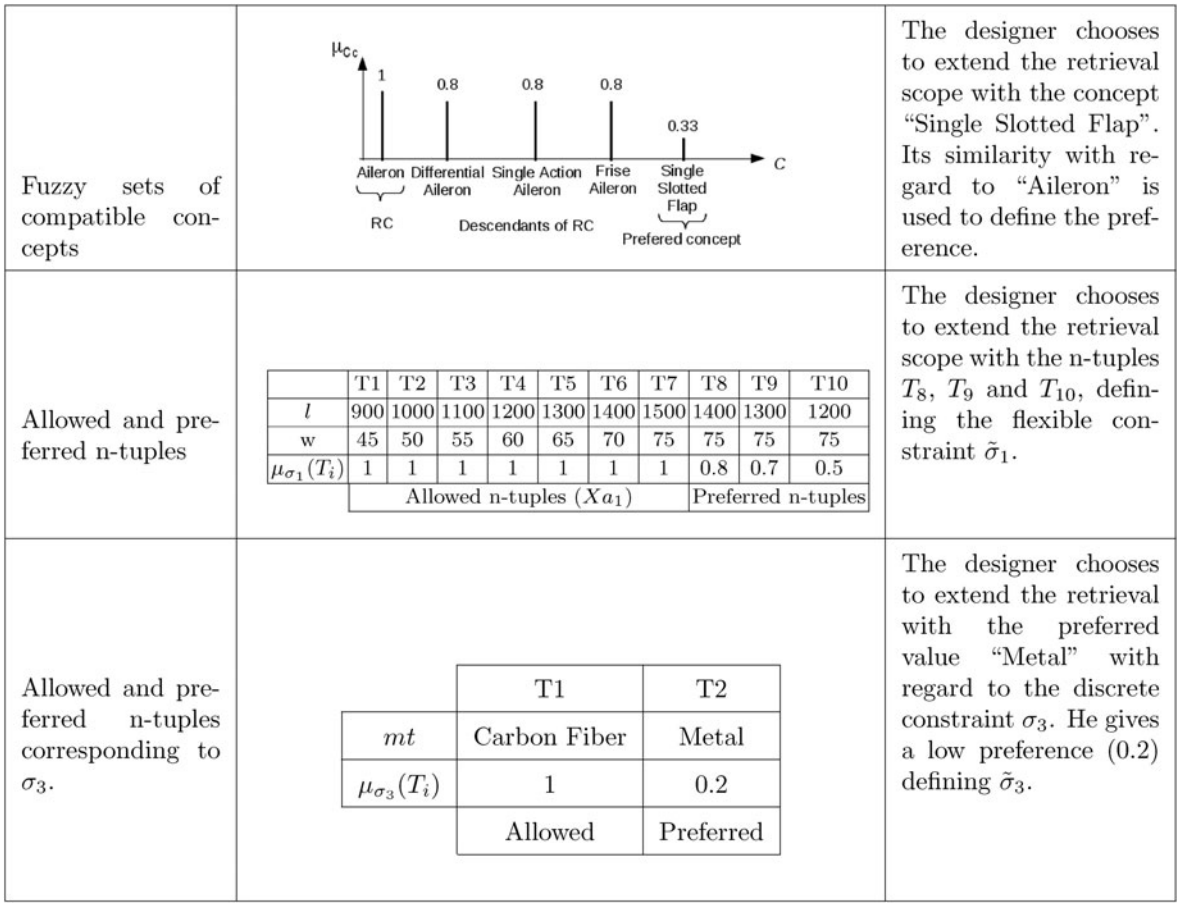
Requirements Concept: RC = *Aileron*	
Conceptual Variables and Domains	
Length (l)	d_l = {900, 1000, 1100, 1200, 1300, 1400, 1500}
Width (w)	d_w = {45, 50, 55, 60, 65, 70, 75}
Weight (wg)	d_{wg} = {25, 50, 75, 100, 125, 150, 175, 200, 250, 300, 350, 400, 450, 500, 550}
Conceptual Constraints	
	σ_1: $l = 20 \times w \rightarrow Xa_1$ = {(900, 45), (1000, 50), (1100, 55), (1200, 60), (1300, 65), (1400, 70), (1500, 75)}
	σ_2: $w \in$ [30.00, 100.00] $\rightarrow Xa_2$ = {45, 50, 55, 60, 65, 70, 75}
Added Variables and Domains	
Material (mt)	d_{mt} = {*Carbon Fiber, Metal*}
Added Requirements Constraints	
	σ_3: $mt \in$ {*Carbon Fiber*} = Xa_3
	σ_4: $l \in$ {1000.00} = Xa_4

Compatibilities of Sol_2 with regard to the constraints. Variables l and w involved in the constraint σ_1 are defined within the solution Sol_2. The compatibility is (Comp (Sol_2, $\tilde{\sigma}_1$) = $\mu_{\sigma_1}(1400, 75) = 0.8$). Variable w takes the value 75 in Sol_2. The compatibility with regard to σ_2 is Comp (Sol_2, $\tilde{\sigma}_2$) = $\mu_{\sigma_2}(75) = 1$. The variable *mt* takes the value "Metal" in Sol_2. The compatibility with regard to σ_3 is Comp (Sol_2, $\tilde{\sigma}_3$) = $\mu_{\sigma_3}(Metal) = 0.2$. Variable l takes the value 1400 in Sol_2. The compatibility with regard to σ_4 is Comp (Sol_2, $\tilde{\sigma}_4$) = $\mu_{\sigma_4}(1400) = 0$.

Aggregation: The global compatibility of the solution Sol_2 with regard to the entire set of flexible constraints $\tilde{\sigma}$ (with $\beta = 2$) is defined by the Eq. (11):

$$Comp(Sol_2, \tilde{\sigma}) = \left(\sum_{i=0}^{4}(1/4) \times (comp(Sol_2, \tilde{\sigma}_i))^2\right)^{1/2} = \sqrt{0.25 \times (0.8^2 + 1^2 + 0.2^2 + 0^2)} = 0.648. \quad (11)$$

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press



	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
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
	Allowed n-tuples (Xa_1)							Preferred n-tuples		

	T1	T2
mt	Carbon Fiber	Metal
$\mu_{\sigma_3}(T_i)$	1	0.2
	Allowed	Preferred

Fig. 12. Flexible requirements.

The results of the compatible cases retrieval process are represented in Table 5.

Analyzing these results, one can observe that the solution Sol_1 is not adapted because it has a global compatibility equal to 0.1. Furthermore, the concept *Single slotted flap* is very far from the RC *Aileron* (similarity = 0.33). The solution Sol_2 is the nearest to the requirements with a global compatibility equal to 0.648 and was obtained by 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 achieved by eliminating the incompatibilities with regard to the constraints σ_1, σ_3, and σ_4.

g. *Adaptation:* The solution Sol_{new} is created from the solution Sol_2 (Table 6). The values of the pair (i.e., couple) of solution variables (l, w) are reduced to the n-tuple (1000, 50) and the couple (wg, mt) is transformed into (60, Carbon Fiber). The variable α is specialized to the concept *Differential Aileron*; thus, it is removed from the solution Sol_{new}. Clearly, in this academic example, the work of the designer to provide a new solution by adapting the prior one is not highlighted. However, her/his work can be facilitated through the benefit of EF from prior designs offered by the proposed method.
h. *Validation, verification and capitalization:* Each constraint of the requirements has to be checked with the

Table 4. *Compatibilities of the solutions with regards to Cc*

Solution	Solution Concept	Compatibility With Regard to *Cc*
Sol_1	Single slotted flap	$Comp(Sol_1, Cc) = \mu_{Cc}(Single\ slotted\ flap) = 0.33$
Sol_2	Differential aileron	$Comp(Sol_2, Cc) = \mu_{Cc}(Differential\ Aileron) = 0.8$
Sol_3	Tubular spar	$Comp(Sol_3, Cc) = \mu_{Cc}(Tubular\ spar) = 0$

Table 5. *Results of the compatible cases retrieval process*

	σ_1	σ_2	σ_3	σ_4	Global Compatibility With Regard to Constraints	Compatibility With Regard to Requirements Concept
Sol_1	0	0	0.2	0	0.1	0.33
Sol_2	0.8	1	0.2	0	0.648	0.8

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

Table 6. *Result of the adaptation of Sol_2 into Sol_{new}*

	Solution Concept *Aileron*			
Variables ($V_{Sol_{new}}$)	Length (l)	Width (w)	Weight (wg)	Material (mt)
Sol_{new} values ($V^{value}_{Sol_{new}}$)	1000	50	60	Carbon fiber

variable values within the solution. One can observe that each constraint is satisfied by the possible values, and then the solution Sol_{new} is validated and capitalized into the case base, encapsulated within a new case with its requirements. This solution will be reused in future design processes.

5.2. Implementation within the ATLAS software

As presented in the Introduction, one of the outcomes of the ATLAS project was the prototype of a software application. A web-based application (ATLAS software) has been developed using the Ruby-On-Rails Framework (see a screen shot in Fig. 13). It permitted researchers to test and validate the proposed process. By means of this software, it is possible to carry out the RCBR process for developing a system of systems. Each system can be composed of many solutions. The software allows a user to define requirements and to retrieve compatible solutions in the case base by integrating the designer preferences. The semantic similarity of Wu and Palmer (1994) has also been implemented. Compatible solutions are identified, and a compatibility degree is evaluated using the proposed approach described in Section 4.2. Then a solution can be reused and adapted to the new requirements (simple copy or copy plus modifications). However, in the current version of the prototype, the requirements constraints are unary constraints (they involve only one variable; Gelle et al., 2000). This software prototype has been tested by an industrial partner of the ATLAS project in order to validate the approach.

5.3. Test of the compatible cases retrieval process

In order to test the compatible cases retrieval algorithm used in the proposed approach, a random generator has been developed in Ruby language. For each experiment, a random ontology has been generated. The parameters are the ontology's depth and the number of descendents of each concept (minimum and maximum). For each concept, some variables and constraints are randomly generated and others are inherited from its ancestors. A case base is also generated: each system is composed of two solutions, and the number of systems is 10^3, 10^4, or 10^5. New requirements are also generated randomly: the RC, the added requirements variables, and the added requirements constraints. The designer's preferences (i.e., the preferred n-tuples) are also randomly added to each requirements constraint. For each test, 10 experiments have been run, and the results show the values corresponding to the minimum/mean/maximum of the indicators. The com-

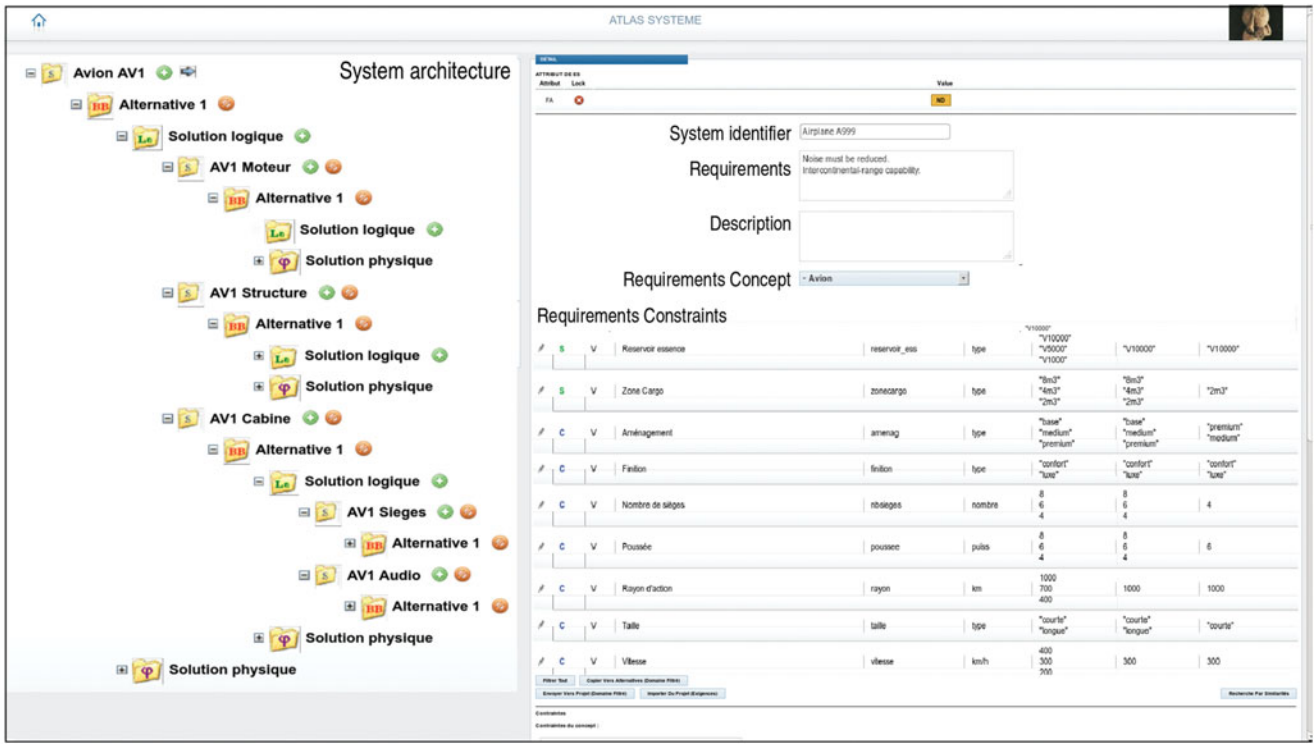


Fig. 13. ATLAS Software.

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

Table 7. *Experimental results*

Ontology Depth	6			15		
No. of Descendents/Concept	0/4/8			0/1.5/3		
No. of concepts	$580/9000/2\times10^4$	$10^3/12\times10^3/2\times10^4$	$2000/10^4/2\times10^4$	$3\times10^4/76\times10^3/10^5$	$10^4/4\times10^4/10^5$	$6000/6\times10^4/10^5$
No. solutions	2×10^3	2×10^4	2×10^5	2×10^3	2×10^4	2×10^5
No. of solution variables	3/16/30	2/17/32	1/16/33	6/34/56	4/33/60	1/34/61
No. of solution constraints	2/9/14	1/9/14	1/9/14	6/18/23	3/18/23	2/18/23
No. of requirements variables	7/13/18	11/15/20	12/16/20	22/32/40	18/28/35	23/32/38
No. of requirements constraints	4/8/10	6/8/10	6/8/10	12/17/19	13/16/19	13/16/18
No. of preselected solutions	13/88/377	118/540/2100	1250/4000/17500	1/20/64	67/500/2000	$1000/4000/15\times10^3$
Compatibility	0.47/0.73/1.0	0.53/0.73/1.0	0.47/0.75/1.0	0.62/0.78/1.0	0.6/0.78/1.0	0.55/0.75/1.0
Retrieval time (s)	0.009/0.02/0.04	0.1/0.2/0.4	0.9/2.4/12	0.02/0.04/0.15	0.2/0.45/1.3	2/7/15

patible cases retrieval process parameters are tuned as follow: the threshold α is equal to 0.5 and β is equal to 2. The results of the different experiments are synthesized in Table 7.

These results show that the algorithm is able to provide retrieved solutions to the designer even for large case bases (10^5 cases). The preselection of compatible solutions using the threshold of 0.5 permits retrieved solutions with a compatibility $\geq$0.47. The computing time of solutions compatibility measures is reduced because only the preselected solutions are taken into account. In order to verify this point, a set of 10 experiments has been run with 7×10^4 concepts (mean value), 2×10^5 solutions, and $\alpha = 0.0$. The mean time to perform the retrieval activity on the whole case base (2×10^5 solutions) is 33 times higher than for the worst experiments obtained in Table 7 (right column). The mean compatibility is equal to 0.36.

5.4. Discussion

The requirements of the example described in the Section 5.1 was to be simple and easy to understand. Its role is mainly to highlight the ideas developed in the article. Furthermore, the experiments that have been carried out and are synthesized in Section 5.3 allow evaluation of the performances of the compatible cases retrieval process. They show that the algorithm developed for the retrieval can be used in a real context where the designer can retrieve frequent solutions from the case base. The use of the double stage retrieval (a conceptual similarity-based selection followed by a deeper comparison of a reduced number of cases) allows limiting of the retrieval time. Based on this example and on the propositions developed in Sections 3 and 4, the main contributions of this paper are synthesized below. Some limitations are also discussed.

The three requirements defined in Section 2.4 are fulfilled simultaneously by the proposed approach for system engineering.

1. It is fully compatible with the system engineering standards. A company that wishes to implement a system engineering standard following ISO-15288 or INCOSE recommendations giving guidelines to the development processes of systems can carry out the RCBR process because it is suitable to system development.
2. Simultaneously, the proposed approach permits the designer to express her/his preferences directly by defining flexible constraints. Thus, the comparison of the problem (i.e., the requirements modeled by a set of flexible constraints) and a solution of the case base is done by calculating a global compatibility measure that is not based on the aggregation of similarities between attributes values. Furthermore, it is not necessary for the designer to define the preferences on each variable value.
3. Finally, the CBR approach is guided by an ontology of concepts all along the case-based design process. The concepts contain knowledge and represent at an abstract level the objects that can be developed. This knowledge is expressed by means of variables, domains, and constraints. First, the requirements definition can be done at a conceptual level by choosing a concept in the ontology. The knowledge embedded into this concept is suitable to aid the designer defining the requirements. Second, the retrieval is a double stage activity that allows acceleration of the retrieval activity: a first step permits a designer to quickly select solutions with a concept that is sufficiently similar to the RC (and/or preferred by the designer). The second step permits the designer to calculate the compatibility of selected solutions. In the third step, during the solution definition process, the concept associated to the solution to develop can help the designer defining the characteristics of the system that fulfills the requirements.

Numeric requirements constraints intentionally expressed (e.g., σ: $2\times\ln(v_1) + 3\times v_2 = 50$) have to be discretized and extensionally expressed, defining the set of allowed n-tuples of values. Then, the designer can express his preferences on each allowed n-tuple. This is not yet implemented in our approach. When a requirement constraint involves a lot of variables, the definition of preferred n-tuples can be difficult for

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

the designer because the expression of the preferences involves a lot of combinations. The adaptation of solutions is a difficult step of CBR approaches. There are no aiding tools proposed in our approach to help designers to adapt solutions. Only the tacit knowledge of the designer is used.

6. CONCLUSION

In this article, using existing academic and industrial standards and existing standard CBR methodologies, an integrated CBR process for system design has been proposed. This process is fully compatible with system engineering requirements. For each step of this process, methods have been proposed to take into account designer preferences at the earliest phases of a design process. Furthermore, to aid designers and to formalize knowledge for design, an ontology has been defined that formalizes guidelines suitable to the proposed RCBR process. For the requirements definition, the retrieval of compatible cases, and the solutions definition, the knowledge embedded in the concepts of the ontology is exploited, leading to improve standardization. This facilitates the future reuse of the acquired knowledge for system design as well as the definition of the corresponding information. The retrieval mechanism is a double stage and preference-based process. The requirements concept corresponding to requirements at a conceptual level is used in order to identify solutions within the case base. Then, requirements constraints are used in order to define compatible cases with a compatibility measure. Preferences of the designer are used at each stage. They permit to the designer to give flexibility to the retrieval process, and moreover, they replace similarity measures generally used in CBR tools and sometimes difficult to obtain from experts. At the first stage, a set of preferred concepts is given by the designer with a preference degree. At the second stage, preferences are expressed for the requirements constraints. The designer expresses how she/he prefers to use some *n*-tuples of values, by defining a preference degree that is exploited during the retrieval step. Our process is compatible with system engineering requirements and processes where systems of systems have to be recursively developed.

The ATLAS software permits designers to show the feasibility of the approach and has been evaluated by industrial partners of the ATLAS project. The experiments have been carried out using a testbed that automatically generates an ontology and cases have shown that the double stage compatible cases retrieval process is efficient even for large case bases.

Extensions of this work concern first the integration of the proposed RCBR process into the project management process. The RCBR process may be integrated into project planning to manage system design projects and reuse prior planning information (e.g., resources, tasks, and durations). Initial models have been proposed be Abeille et al. (2010) and Coudert, Vareilles, Geneste, et al. (2011). Second, the approach may be improved in order to better take into account numeric constraints without discretization and to propose aiding tools to the designer for expressing its preferences when requirements constraints involve a lot of variables. Third, constraint satisfaction problem filtering tools may be used in order to reduce the solution space very early. We have proposed first methods for coupling such CBR and constraint satisfaction problem in Vareilles et al. (2012).

REFERENCES

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications 7(1)*, 39–52.

Abeille, J., Coudert, T., Vareilles, É., Geneste, L., Aldanondo, M., & Roux, T. (2010). Formalization of an integrated system/project design framework: first models and processes. In *Complex Systems and Management* (Aiguier, M., Bretaudeau, F., & Krob, D., Eds.), pp. 207–217. Berlin: Springer.

Althoff, K.-D., & Weber, R. (2005). Knowledge management in case-based reasoning. *Knowledge Engineering Review 20(3)*, 305–310.

Altshuller, G. (1996). *And Suddenly the Inventor Appeared: Triz, the Theory of Inventive Problem Solving*. Worcester, MA: Technical Innovation Center.

Armaghan, N., & Renaud, J. (2012). An application of multi-criteria decision aids models for case-based reasoning. *Information Sciences 210*, 55–66.

Avramenko, Y., & Kraslawski, A. (2006). Similarity concept for case-based design in process engineering. *Computers & Chemical Engineering 30(3)*, 548–557.

Batet, M., Sánchez, D., & Valls, A. (2011). An ontology-based measure to compute semantic similarity in biomedicine. *Journal of Biomedical Informatics 44(1)*, 118–125.

Benferhat, S., Dubois, D., Kaci, S., & Prade, H. (2006). Bipolar possibility theory in preference modeling: representation, fusion and optimal solutions. *Information Fusion 7(1)*, 135–150.

Bergmann, R. (2002). *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Berlin: Springer.

Brandt, S.C., Morbach, J., Miatidis, M., Theißen, M., Jarke, M., & Marquardt, W. (2008). An ontology-based approach to knowledge management in design processes. *Computers & Chemical Engineering 32(1–2)*, 320–342.

Cao, D., Li, Z., & Ramani, K. (2011). Ontology-based customer preference modeling for concept generation. *Advanced Engineering Informatics 25(2)*, 162–176.

Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horvth, I., Bernard, A., Harik, R.F., & Gao, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design 45(2)*, 204–228.

Chang, X., Sahin, A., & Terpenny, J. (2008). An ontology-based support for product conceptual design. *Robotics and Computer-Integrated Manufacturing 24(6)*, 755–762.

Chen, X., Gao, S., Guo, S., & Bai, J. (2012). A flexible assembly retrieval approach for model reuse. *Computer-Aided Design 44(6)*, 554–574.

Chen, Y.-J., Chen, Y.-M., Chu, H.-C., & Kao, H.-Y. (2008). On technology for functional requirement-based reference design retrieval in engineering knowledge management. *Decision Support Systems 44(4)*, 798–816.

Chenouard, R., Granvilliers, L., & Sebastian, P. (2009). Search heuristics for constraint-aided embodiment design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23*, 175–195.

Cordi, V., Lombardi, P., Martelli, M., & Mascardi, V. (2005). An ontology-based similarity between sets of concepts. In *Proc. Workshop dagli Oggetti agli Agenti (WOA)* (Corradini, F., Paoli, F.D., Merelli, E., & Omicini, A., Eds.), pp. 16–21. Bologna: Pitagora Editrice.

Cordier, A., Mascret, B., & Mille, A. (2009). Extending case-based reasoning with traces. In *Grand Challenges for Reasoning from Experiences*, workshop at IJCAI'09.

Cortes-Robles, G., Negny, S., & Le-Lann, J.M. (2009). Case-based reasoning and TRIZ: a coupling for innovative conception in chemical engineering. *Chemical Engineering and Processing: Process Intensification 48(1)*, 239–249.

Coudert, T., Vareilles, É., Aldanondo, M., Geneste, L., & Abeille, J. (2011). Synchronization of system design and project planning: integrated model and rules. *5th IEEE Int. Conf. Software, Knowledge, Information, Industrial Management and Applications (SKIMA' 2011)*, pp. 1–6.

Coudert, T., Vareilles, É., Geneste, L., Aldanondo, M., & Abeille, J. (2011). Proposal for an integrated case based project planning. In *Complex Sys-*

https://doi.org/10.1017/S0890060413000498 Published online by Cambridge University Press

tems Design and Management (Hammami, O., Krob, D., & Voirin, J.-L., Eds.), pp. 133–144. Berlin: Springer.
Dalkir, K. (2005). *Knowledge Management in Theory and Practice*. Amsterdam: Elsevier/Butterworth Heinemann.
Darlington, M.J., & Culley, S.J. (2008). Investigating ontology development for engineering design support. *Advanced Engineering Informatics 22(1)*, 112–134.
Dieter, G. (2000). *Engineering Design: A Materials and Processing Approach*. New York: McGraw–Hill.
Domshlak, C., Hüllermeier, E., Kaci, S., & Prade, H. (2011). Preferences in AI: an overview. *Artificial Intelligence 175(7–8)*, 1037–1052.
Dubois, D., Esteva, F., Garcia, P., Godo, L., de Mantaras, R.L., & Prade, H. (1997). Fuzzy modelling in case-based reasoning and decision. *Proc. ICCBR-97, Case-Based Reasoning Research and Development* (Leake, D.B., & Plaza, E., Eds.), pp. 599–610. New York: Springer–Verlag.
Dubois, D., Fargier, H., & Prade, H. (1996). Possibility theory in constraint satisfaction problems: handling priority, preference and uncertainty. *Applied Intelligence 6(4)*, 287–309.
Dubois, D., Prade, H., Esteva, F., Garcia, P., Godo, L., & Lopez de Mantaras, R. (1998). Fuzzy set modelling in case-based reasoning. *International Journal of Intelligent Systems 13(4)*, 345–373.
Faure, A., & Bisson, G. (1999). Modeling the experience feedback loop to improve knowledge base reuse in industrial environment. In *12th Workshop on Knowledge Acquisition, Modeling and Management, KAW* 99. Banff, Canada.
Finnie, G.R., & Sun, Z. (2003). R^5 model for case-based reasoning. *Knowledge-Based Systems 16(1)*, 59–65.
Foguem, B.K., Coudert, T., Béler, C., & Geneste, L. (2008). Knowledge formalization in experience feedback processes: an ontology-based approach. *Computers in Industry 59(7)*, 694–710.
Gao, C., Huang, K., Chen, H., & Wang, W. (2006). Case-based reasoning technology based on TRIZ and generalized location pattern. *Journal of TRIZ in Engineering Design 2*, 40–58.
Gelle, E., Faltings, B., Clément, D.E., & Smith, I.F.C. (2000). Constraint satisfaction methods for applications in engineering. *Engineering With Computers (London) 16(2)*, 81–95.
Gero, J.S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine 11(4)*, 26–36.
Girard, P., & Doumeingts, G. (2004). Modelling the engineering design system to improve performance. *Computers and Industrial Engineering 46(1)*, 43–67.
Goel, A.K., & Craw, S. (2006). Design, innovation and case-based reasoning. *Knowledge Engineering Review 20(3)*, 271–276.
Gomez De Silva Garza, A., & Maher, M. (1996). Design by interactive exploration using memory-based techniques. *Knowledge-Based Systems 9(3)*, 151–161.
Gu, D.-X., Liang, C.-Y., Bichindaritz, I., Zuo, C.-R., & Wang, J. (2012). A case-based knowledge system for safety evaluation decision making of thermal power plants. *Knowledge-Based Systems 26*, 185–195.
Guo, Y., Hu, J., & Hong Peng, Y. (2012). A CBR system for injection mould design based on ontology: a case study. *Computer-Aided Design 44(6)*, 496–508.
Haskins, C. (2011). *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*. San Diego, CA: INCOSE.
Huang, C.-C., & Kusiak, A. (1998). Modularity in design of products and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A 28(1)*, 66–77.
Huysentruyt, J., & Chen, D. (2010). Contribution to the development of a general theory of design. *8th Int. Conf. Modeling and Simulation, MOSIM 2010*, Hammamet, Tunisia.
ISO. (2008). *ISO/IEC 15288:2008. Systems and Software Engineering System Life Cycle Processes*. Geneva: Author.
Jabrouni, H., Foguem, B.K., Geneste, L., & Vaysse, C. (2011). Continuous improvement through knowledge-guided analysis in experience feedback. *Engineering Applications of Artificial Intelligence 24(8)*, 1419–1431.
Jabrouni, H., Kamsu-Foguem, B., & Geneste, L. (2009). Exploitation of knowledge extracted from industrial feedback processes. *Proc. Software, Knowledge and Information Management and Applications, SKIMA 2009*, Fes, Morocco.
Janthong, N., Brissaud, D., & Butdee, S. (2010). Combining axiomatic design and case-based reasoning in an innovative design methodology of mechatronics products. *CIRP Journal of Manufacturing Science and Technology 2(4)*, 226–239.
Junker, U., & Mailharro, D. (2003). Preference programming: advanced problem solving for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17(1)*, 13–29.
Kam, C., & Fischer, M. (2004). Capitalizing on early project decision-making opportunities to improve facility design, construction, and life-cycle performance-POP, PM4D, and decision dashboard approaches. *Automation in Construction 13(1)*, 53–65.
Kim, K.-Y., Manley, D.G., & Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design 38(12)*, 1233–1250.
Kolb, D.A. (1984). Experiential learning: experience as the source of learning and development. *Journal of Organizational Behavior 8*, 359–360.
Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.
Lau, A.S.M., Tsui, E., & Lee, W.B. (2009). An ontology-based similarity measurement for problem-based case reasoning. *Expert Systems With Applications 36(3)*, 6574–6579.
Leake, D., & McSherry, D. (2005). Introduction to the special issue on explanation in case-based reasoning. *Artificial Intelligence Review 24(2)*, 103–108.
Lee, K., & Luo, C. (2002). Application of case-based reasoning in die-casting die design. *International Journal of Advanced Manufacturing Technology 20*, 284–295.
Liu, D.-R., & Ke, C.-K. (2007). Knowledge support for problem-solving in a production process: a hybrid of knowledge discovery and case-based reasoning. *Expert Systems With Applications 33(1)*, 147–161.
Liu, H.-W. (2005). New similarity measures between intuitionistic fuzzy sets and between elements. *Mathematical and Computer Modelling 42(12)*, 61–70.
Macedo, L., & Cardoso, A. (1998). Nested graph-structured representations for cases. *Proc. 4th European Workshop on Advances in Case-Based Reasoning (EWCBR-98)* (Smyth, B., & Cunningham, P. Eds.), LNAI, Vol. 1488, pp. 1–12. Berlin: Springer.
Maher, M.-L., & Gomez de Silva Garza, A. (1997). Case-based reasoning in design. *IEEE Expert 12(2)*, 34–41.
Martin, J.N. (2000). Processes for engineering a system: an overview of the ansi/eia 632 standard and its heritage. *Systems Engineering 3(1)*, 1–26.
Mileman, T., Knight, B., Petridis, M., Cowell, D., & Ewer, J. (2002). Case-based retrieval of 3-dimensional shapes for the design of metal castings. *Journal of Intelligent Manufacturing 13*, 39–45.
Mok, C., Hua, M., & Wong, S. (2008). A hybrid case-based reasoning CAD system for injection mould design. *International Journal of Production Research 46(14)*, 3783–3800.
Mondragon, C.C., Mondragon, A.C., Miller, R., & Mondragon, E C. (2009). Managing technology for highly complex critical modular systems: the case of automotive by-wire systems. *International Journal of Production Economics 118(2)*, 473–485.
Montanari, U. (1974). Networks of constraints: fundamental properties and application to picture processing. *Information Science 7*, 95–132.
Nanda, J., Thevenot, H.J., Simpson, T.W., Stone, R.B., Bohm, M., & Shooter, S.B. (2007). Product family design knowledge representation, aggregation, reuse, and analysis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21(2)*, 173–192.
Negny, S., & Le-Lann, J. (2008). Case-based reasoning for chemical engineering design. *Chemical Engineering Research and Design 86(6)*, 648–658.
Negny, S., Riesco, H., & Lann, J.-M.L. (2010). Effective retrieval and new indexing method for case based reasoning: application in chemical process design. *Engineering Applications of Artificial Intelligence 23(6)*, 880–894.
Pahl, G., & Beitz, W. (1984). *Engineering Design: A Systematic Approach*. Berlin: Springer.
Policastro, C.A., de Carvalho, A.C.P.L.F., & Delbem, A.C.B. (2006). Automatic knowledge learning and case adaptation with a hybrid committee approach. *Journal of Applied Logic 4(1)*, 26–38.
Policastro, C.A., de Carvalho, A.C.P.L.F., Delbem, A.C.B. (2008). A hybrid case adaptation approach for case-based reasoning. *Applied Intelligence 28(2)*, 101–119.
Qin, X., & Regli, W. (2003). A study in applying case-based reasoning to engineering design: mechanical bearing design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17(3)*, 235–252.
Rakoto, H., Hermosillo-Worley, J., & Ruet, M. (2002). Integration of experience based decision support in industrial processes. *IEEE Int. Conf. Systems, Man and Cybernetics, SMC'02*. Hammamet, Tunisia.

Richards, D., & Simoff, S.J. (2001). Design ontology in context—a situated cognition approach to conceptual modelling. *Artificial Intelligence in Engineering 15(2)*, 121–136.
Ruet, M., & Geneste, L. (2002). Search and adaptation in a fuzzy object oriented case base. *Proc. 6th European Conf. Case Based Reasoning*, LNAI, Vol. 2416, pp. 350–364. Berlin: Springer.
Saridakis, K., & Dentsoras, A. (2007). Case-desc: a system for case-based design with soft computing techniques. *Expert Systems With Applications 32(2)*, 641–657.
Settouti, L.S., Prié, Y., Marty, J.-C., & Mille, A. (2009). A trace-based system for technology-enhanced learning systems personalisation. *Proc. 9th IEEE Int. Conf. Advance Learning Technologies*, pp. 93–97.
Simon, H. (1969). *The Sciences of the Artificial*. Cambridge, MA: MIT Press.
Stahl, A., & Bergmann, R. (2000). Applying recursive CBR for the customization of structured products in an electronic shop. *Advances in Case-Based Reasoning* (Blanzieri, E., & Portinale, L. Eds.), LNCS, Vol. 1898, pp. 297–308. Berlin: Springer.
Studer, R., Benjamins, V.R., & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & Knowledge Engineering 25(1–2)*, 161–197.
Suh, N.P. (1990). *The Principles of Design*. New York: Oxford University Press.
Sun, Z., Han, J., & Dong, D. (2008). Five perspectives on case based reasoning. *Advanced Intelligent Computing Theories and Applications: With Aspects of Artificial Intelligence* (Huang, D.-S., Wunsch, D.C., Levine, D., & Jo, K.-H., Eds.), LNSC, Vol. 5227, pp. 410–419. Berlin: Springer.
Tang, M. (1997). A knowledge-based architecture for intelligent design support. *International Journal of Knowledge Engineering Review 12(4)*, 387–460.
Thornton, A.C. (1996). The use of constraint-based design knowledge to improve the search for feasible designs. *Engineering Applications of Artificial Intelligence 9(4)*, 393–402.
Ullman, D. (2003). *The Mechanical Design Process*. New York: McGraw–Hill Higher Education.
Uschold, M., & Gruninger, M. (1996). Ontologies: principles, methods and applications. *Knowledge Sharing and Review 11(2)*, 93–155.
Vareilles, E., Aldanondo, M., de Boisse, A.C., Coudert, T., Gaborit, P., & Geneste, L. (2012). How to take into account general and contextual knowledge for interactive aiding design: towards the coupling of csp and cbr approaches. *Engineering Applications of Artificial Intelligence 25(1)*, 31–47.
Wang, J., Tang, M., & Gabrys, B. (2006). An agent-based system supporting collaborative product design. *Knowledge-Based Intelligent Information and Engineering Systems* (Heidelberg, S.-V.B., Ed.), LNAI, Vol. 4252, Part II, pp. 670–677. Berlin: Springer.
Wang, W.-J. (1997). New similarity measures on fuzzy sets and on elements. *Fuzzy Sets and Systems 85(3)*, 305–309.
Weber, R., Aha, D.W., & Becerra-Fernandez, I. (2001). Intelligent lessons learned systems. *Expert System Applications 20(1)*, 17–34.
Woon, F.L., Knight, B., Petridis, M., & Patel, M.K. (2005). CBE-conveyor: a case-based reasoning system to assist engineers in designing conveyor systems. *Case-Based Reasoning Research and Development* (Muñoz-Avila, H., & Ricci, F., Eds.), LNCS, Vol. 3620, pp. 640–651. Berlin: Springer.
Wu, M.-C., Lo, Y.-F., & Hsu, S.-H. (2008). A fuzzy cbr technique for generating product ideas. *Expert Systems With Applications 34(1)*, 530–540.
Wu, Z., & Palmer, M. (1994). Verb semantics and lexical selection. *Proc. 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–138, New Mexico State University, Las Cruces.
Xuanyuan, S., Jiang, Z., Li, Y., & Li, Z. (2011). Case reuse based product fuzzy configuration. *Advanced Engineering Informatics 25(2)*, 193–197.
Yang, C., & Chen, J. (2011). Accelerating preliminary eco-innovation design for products that integrates case-based reasoning and TRIZ method. *Journal of Cleaner Production 19*, 998–1006.
Zarandi, M.F., Razaee, Z.S., & Karbasian, M. (2011). A fuzzy case based reasoning approach to value engineering. *Expert Systems With Applications 38(8)*, 9334–9339.

Juan Camilo Romero Bejarano is a Supply Chain Consultant and Trainer in the aeronautical industry. He is also a PhD student in industrial systems at the University of Toulouse. He obtained his MS from the University of Toulouse and his BS in industrial engineering from the National University of Colombia. His research interests are focused on problem solving and knowledge management within the frame of collaborative supply chains.

Thierry Coudert is an Assistant Professor in the Ecole Nationale D'Ingenieurs de Tarbes, National Polytechnic Institute of Toulouse, Laboratoire Génie de Production, University of Toulouse. His research is carried out at the Laboratoire Génie de Production. His work mainly concerns system engineering, metaheuristics for system engineering, and knowledge acquisition and exploitation by experience feedback approaches.

Elise Vareilles is an Assistant Professor at the University of Toulouse. She received a PhD from the National Polytechnic Institute of Toulouse in 2006. Dr. Vareilles' research interests are the development of interactive knowledge based aiding design tools.

Laurent Geneste is with a Professor in the Ecole Nationale D'Ingenieurs de Tarbes, National Polytechnic Institute of Toulouse, University of Toulouse. He received his PhD from University Paul Sabatier (Toulouse) in 1995 and an accreditation to supervise research in 2002. Dr. Geneste is currently Head of Cognitive and Decisional Systems in the Production Management Laboratory in Tarbes. His current research interest relates to knowledge engineering and more specifically to experience feedback and lessons learned for problem solving in industrial organizations.

Michel Aldanondo is a Professor and Director of the Industrial Engineering Laboratory, Mines-Albi, University of Toulouse. Professor Aldanondo teaches design and operation management courses, mainly at the graduate level. His research is concentrated on the development of interactive knowledge based design tools. He has directed 11 PhD students and more than 50 master's students. Dr. Aldanondo has published more than 150 articles in journals and conference proceedings.

Joël Abeille received his PhD degree from the National Polytechnic Institute of Toulouse (Toulouse) in 2008. He is currently an engineer in an R&D company.