ARDECO: an assistant for experience reuse in CAD

Pierre-Antoine Champin

ARDECO: an assistant for experience reuse in Computer Aided Design

Pierre-Antoine Champin

Lyon Research Center for Images and Intelligent Information Systems
LIRIS FRE 2672 CNRS — Lyon 1 University, France
champin@liris.univ-lyon1.fr
http://liris.univ-lyon1.fr/

Abstract. In this paper, we present the ARDECO prototype, an assistant to experience reuse for CAD systems, and more specifically aimed at Dassault Systèmes' CATIA. This work has taken part in an interdisciplinary project, aiming at studying both human and technical aspects of experience reuse in design. We propose the notion of *design episode* to capture and represent design experience, and a general architecture for retrieval and adaptation of episodes.

1 Introduction

Design is an inherently complex activity. Designers of complex systems thus often have to rely on their experience rather than on predefined rules or methods to solve design problems. Therefore, reusing previous design solutions and adapting them to new design problems is often considered easier and faster than solving those problems from scratch. It is not surprising then that Case Based Reasoning (CBR) has already been used to enhance Computer Aided Design (CAD) applications.

ARDECO[1] has been an interdisciplinary project involving researchers from cognitive ergonomics and artificial intelligence. It aimed at studying the processes of experience reusing in design, in order to provide an efficient computer assistant for it. It has been funded by the french National Center for Scientific Research, and by the industrial partner Dassault Systèmes. The work [2] has therefore been focused on Dassault's CAD application CATIA.

In the next section (2), we will present existing CBR systems used in CAD applications, and discuss their limitations with regard to the complexity of design. We will then propose the notion of design episode to represent reusable experience "units", while coping with that complexity. Section 3 discusses how episodes are represented. Section 4 describes the retrieval and adaptation of episodes to provide assistance to designers. Finally, we conclude and discuss some further work.

[1] http://www710.univ-lyon1.fr/~champin/ardeco

2 Capturing design experience

2.1 Design as an opportunistic activity

As we stressed it in introduction, complexity is what characterizes design problems. The specifics of the design activity have been widely studied. Models of this activity have been proposed [7], if not to reduce its complexity, at least to harness it. From these models, it appears that design problems are *ill defined* problems. It follows that solving them is often performed in an *opportunistic* way [9]: the solving task can not be structured in advance into a hierarchical decomposition of subtasks. Designers rather sets goals and subgoals all along the activity, and may achieve, change or abandon them depending on the evolution of their work.

However, most CBR systems proposed to assist designers do not take into account the ill defined structure of design problems, nor the opportunistic aspect of solving them. A system like RESYN/CBR [5] tackles a very well defined problem class (build a plan to produce a given molecule). Similarly, the FAMING system explicitly focuses an a very specific class of mechanical problems, where a lot of case features can be automatically reconstructed from a limited amount of available data. Other systems like DÉJÀ VU [8] or CADsyn [6] use hierarchically decomposed cases (in software design and architectural design, respectively).

All those systems have proved useful in their specific domains. However, they make strong assumptions about the structure of cases. The elaboration of cases thus becomes a critical step. In limited, well known domains, as the ones addressed by the cited systems, this step can be automated. But in systems with a larger scope, it falls to designers to elicit their goals and motivations, in order to make the cases actually usable. Since it is not always a familiar practice, and since it can often be performed only *a posteriori* because of the opportunistic organization of the design activity, this elicitation effort may become an obstacle to using the system at all.

This problem with case elaboration in design has already been noticed by [4], stressing the difference between deep (*i.e.* explained) cases and shallow cases, and insisting on the fact that only the latter kind is available in many CAD applications. In ARDECO as well, we tried not to put the burden of case elaboration on the designer, nor to limit the scope of the help system to a particular design task.

2.2 Design episodes

To provide designers with an assistant for reusing experience, we first had to identify reusable "units", which could be manage with CBR techniques. We called these units *design episodes*, after the distinction in cognitive psychology between episodic and semantic memory systems. The latter is supposed to store semantic knowledge (*e.g.*, concepts, general rules) while the former is supposed to store contextualized knowledge (dated and located memories), hence experience. Though the semantic memory has long been considered preeminent in problem

solving, a growing number of cognitive psychologists now consider that the role of episodic memory has been underestimated — hence, partly, the emergence of CBR.

We define a design episode as the part of the activity between the moment when a new goal is identified, and the moment when this goal has been achieved (or possibly abandoned). Thus we take into account the opportunistic organization of the design activity [9]. While hierarchical decomposition of tasks allows other systems to manage complexity and keep cases reusable, we do not make such an assumption. Instead, we let our episodes focus on a single goal in order to keep them reusable.

Our colleagues in cognitive ergonomics have studied CATIA users in order to locate the boundaries of design episodes [1]. They have found that designers have a recurrent behavior when setting a new goal and finishing it. They named it "location phase" and "checking phase", respectively. The first one consists in exploring the artifact being designed until determining the next goal to pursue. The second one consists in checking the artifact for expected features and behaviors, in order to determine the actual results of what has been performed. Both phases are characterized by intensive changes of the viewpoint in the 3D environment of the CAD application. Those *behavioral cues* can hence be used to automatically detect episodes boundaries.

It is also worth noting that episodes are highly dependent on the degree of expertise of designers. Indeed, while novice designers set a great number of intermediate goals to achieve a given task, the same task will be solved with a small number of intermediate goals, possibly only one, by an expert designer. This intuitive fact has been confirmed by the observations of our colleagues in ergonomics, and raises a limitation to the reusability of design episodes ("expert" episodes being not reusable for novice users, and *vice-versa*). We consider this limitation to be the drawback of a generalist help system — such problems obviously do not arise in systems helping to solve one single class of problems.

3 Episode representation

Now that we are able to detect design episodes, we have to represent them in a form which will be adapted for CBR mechanisms. CATIA not only provides designers with a geometrical view of the artifact being designed. A second view is available: a hierarchy of objects, some having a direct geometrical counterpart (*e.g.*, extruded profile, hole), but some others more abstract (*e.g.*, numerical parameter, geometrical constraint). Let us insist on the fact that this second view is as familiar to CATIA users as the geometrical one. Indeed, many operations are more easily performed in the hierarchical view than in the geometrical view. Finally, the more abstract character of the hierarchical view makes it relevant for episode representation.

A design episode is represented simply by the hierarchical view of its *initial state* (the moment when the goal is set, detected at the location phase) and the hierarchical view of its *final state* (the moment when the goal is achieved,

detected at the checking phase). The *transition* between both states is also computed, as a set of deletions and additions to obtain the final state from the initial state. The whole design activity can be described as a *trace*, a chain of successive episodes, the final state of one being the initial state of the other.

Differential vs. dynamic representation of episodes

One could be concerned about the fact that we do not deal with the actual operations performed by the designers between both states of an episode. We did consider representing episodes *dynamically* (*i.e.* as a set of operations) rather than *differentially* (as a difference between two states). Theoretically, states could have been reconstructed given all the intermediate operations. However, CATIA users are already familiar with the static representation we use for states, while a reified representation of their operations might have been less meaningful for them.

Furthermore, we already mentioned the difference of abstraction between expert design episodes and novice design episodes. Operations, on the other hand, always have the same level of granularity. Managing abstract episodes with fine-grained operations only would have required a lot of domain specific knowledge, while the hierarchical structure of states allow us to easily abstract out useless details.

4 Episode retrieval and reuse

Figure 1 presents the global architecture of the ARDECO assistant. The hollow arrows (1–3) represent normal design activity, while black arrows (a–e) represent the assisting process. The CAD application, CATIA, is instrumented in order to produce design episodes (2) according to the activity of the designer (1). Those episodes are read by the Assistant, and stored by the Episode Manager (3) as a trace. When required by the designer (a), the Assistant asks the Episode Manager (b) for previous episodes, reusable in the current context. The Episode Manager retrieves the most reusable episodes (c), which are proposed to the designer by the Assistant with their adaptation to the given context (d). It is then up to the designer to apply the appropriate episode in the CAD application (e), following or possibly revising the adaptation proposed by the Assistant. In the following, we will focus on the retrieval and adaptation parts.

4.1 Retrieving episodes

States and transition constituting the design episodes are represented as labeled graphs — the hierarchical view used to represent states involves primarily tree-like composition relationships, hence its name, but other relations also exists, making it a graph rather than a tree. We developed a similarity measure [3] in order to assess the reusability of an episode in a given context. This measure

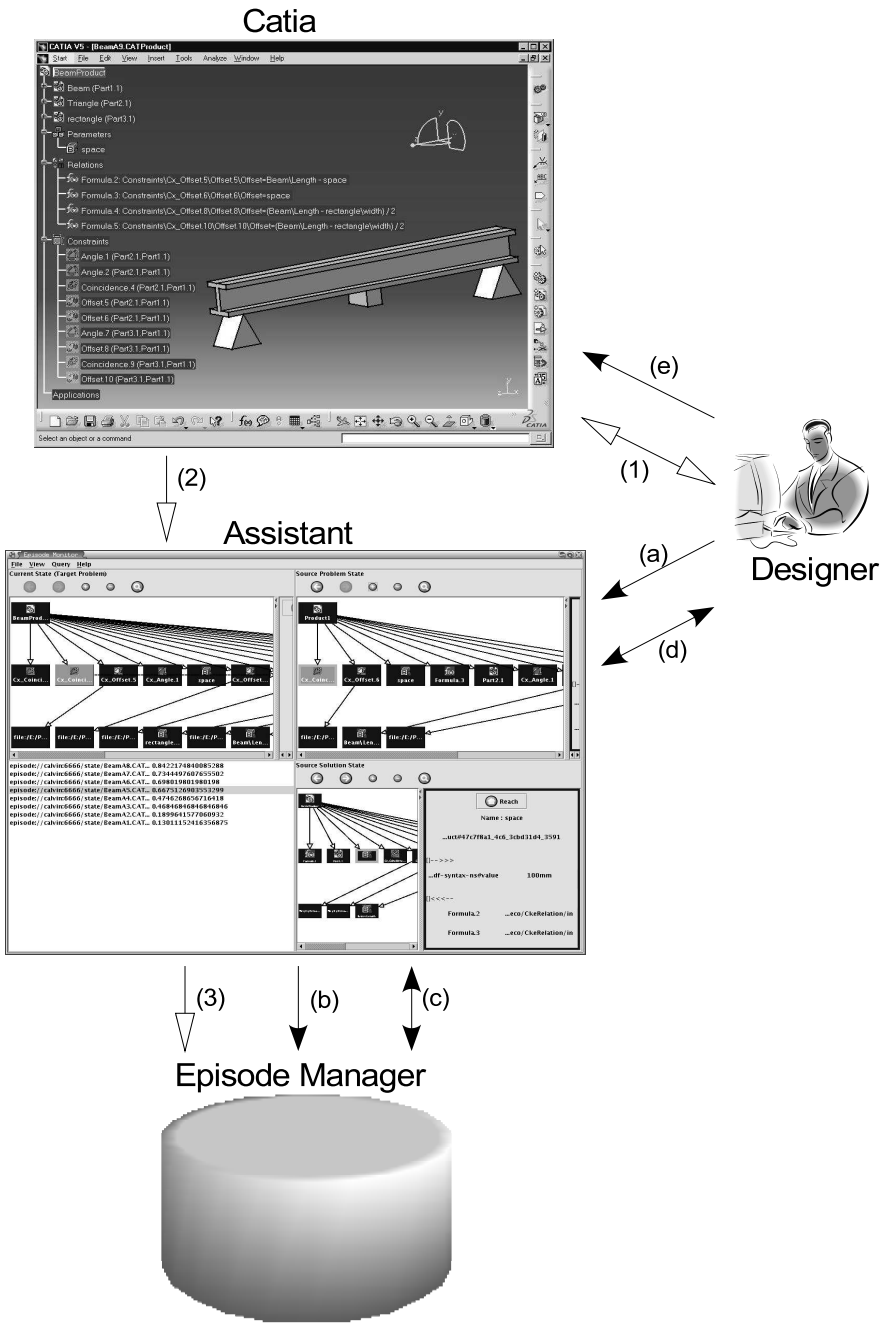

Fig. 1. The ARDECO architecture

does not only provide an opaque similarity value, but rather looks for a correspondence between elements of the graphs, such as to minimize the differences between the graphs: the smaller the set of differences, the higher the similarity value. This difference-based approach has two advantages in CBR. First, categories of differences as well as individual ones can be valuated differently to compute the minimal set of differences. It follows that general domain knowledge, as well as specific contextual knowledge, can easily be taken into account by our measure. Second, the difference-minimizing correspondence can be supplied to the reuse phase in order to guide adaptation (see section 4.2). In this respect, they can be compared to similarity paths introduced by [5].

To retrieve a reusable episode, the Episode Manager searches the traces for a state similar to the last encountered one, considered as the current context. The episodes whose initial state is similar enough will be proposed for adaptation to the designer. This notion of *context* (the state of the application at the moment the user is querying for help) is the shallowest possible one. Indeed, we could as well consider that the relevant context is made up of the two or three last encountered states. It is hard to decide *a priori*, though, which depth is relevant in a particular domain. Only experimentations can answer that question.

4.2 Adapting episodes

As we already mentioned, one advantage of our similarity measure is that it provides a relevant correspondence between the reusable episode's initial state and the current context. The adaptation process is then simply to "replay" the transition of the reusable episode from the current context, in order to produce a new state. Domain knowledge can be used to avoid naive mistakes while replaying. However, the designer still has to check, and potentially revise, the result of the adaptation.

For the sake of simplicity, the example we give is a block world example rather than a full CATIA episode. The upper part of figure 2 represents an episode whose initial state is similar to the current state. The similarity measure provides the correspondence between the element of both states: A with 1 and 2, B with 3 and C with 4 (meaning that 1 and 2 play the same *role* in the context as A plays in the initial state, 3 plays the same role as B, etc.). That allows the assistant to replay the original transition on the appropriate objects: modifications of relations between elements of the initial state are applied, whenever possible, on the relations between the corresponding objects. If needed, objects are also created or deleted accordingly to the reused transition: in the example, block 5 is created accordingly to the creation of block D. Domain knowledge is used to prevent the replay when it is not possible: in the example, since nothing can be put on a pyramid (block 4), block 3 is no longer in relation with the other blocks. The designer will have to decide what becomes of it.

The adapted state is then proposed to the designer, with possibly inconsistent features, should they be detected (e.g., the unknown position of 3) or undetected. In our example, the designer deletes block 3. Furthermore, he has an implicit goal to prevent any white block to be on top of a gray block. So he decides to

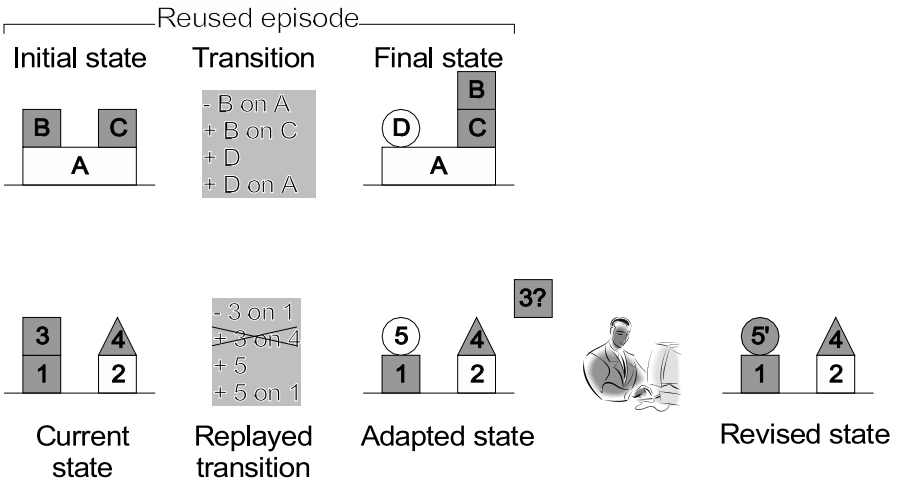

Fig. 2. Adapting an episode: an example

modify the color of the newly created block 5 to satisfy this goal. This block world example, although simple, illustrates how the adaptation process works, uses available design knowledge and relies on the designer when it fails to solve local problems.

5 Conclusion and further work

In this paper, we presented the ARDECO prototype, a assistant in CAD for helping designers to reuse their past experience. A way to capture experience as *design episodes* has been discussed, which is founded on the results of work in cognitive ergonomics. A model for representing episodes has then been proposed, as well as retrieval and adaptation processes. This work has shown that CBR mechanism can be used in a CAD context without any strong hypothesis about the structure of the design activity nor elicitation of high level information by designers.

There are a number of perspectives to this work. First, the duration of the project has not allowed us yet to test the prototype in "real" design situations. Only such tests could help us fine-tune a number of parameters in order to provide an efficient assistance to designers. Those parameters include the relative importance given to the elements of states for the similarity measure, as well as the depth of the context to be compared when retrieving reusable episodes (as mentioned in section 4.1).

It seems also desirable to better integrate the assistant with the original application. This would allow reused and adapted states to be seamlessly presented to designers. However, this is not a trivial work: this would imply to change the application in order to enable the representation of similarity correspondence

(which has been a real challenge when developing the separate assistant). It would also need to represent inconsistent states, as shown in the example for the adaptation process (section 4.2).

Finally, the proposed architecture is not limited to CAD in mechanics. We intend to apply it to the domain of software design. Indeed, software designers often use version management tools, in which they *explicitly* declare relevant states of their work — and even annotate it with textual explanations. Such information could be valuable to build design traces *a posteriori*, and then provide an experience reuse assistance in that field.

References

1. Patrick Bougé, Françoise Détienne, and Laurent Di Cesare. Épisodes de conception : une étude ergonomique pour le recueil de "cas". In Jean Charlet, editor, *Journées française d'Ingénierie des Connaissances*, pages 79–94, Grenoble (FR), June 2001. Presses Universitaires de Grenoble, Grenoble (FR).
2. Pierre-Antoine Champin. *Modéliser l'expérience pour en assister la réutilisation : de la Conception Assistée par Ordinateur au Web Sémantique.* Thèse de doctorat en informatique, Université Claude Bernard, Lyon (FR), 2002.
3. Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In Kevin Ashley and Derek Bridge, editors, *Fifth International Conference on Case-Based Reasoning*, Trondheim, Norway, June 2003.
4. Kefeng Hua, Boi Faltings, and Ian Smith. CADRE: Case-Based Geometric Design. *Journal of Artificial Intelligence in Engineering*, 10:171–183, 1996.
5. Jean Lieber and Amedeo Napoli. Using Classification in Case-Based Planning. In Wolfgang Wahlster, editor, *European Conference on Artificial Intelligence*, pages 132–136, Budapest (HU), August 1996.
6. Mary-Lou Maher and Andrés Gómez. Developing Case-Based Reasoning for Structural Design. *IEEE Expert*, 11(3):42–52, 1996.
7. Lena Qian and John S. Gero. Function-behaviour-structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10:289–312, 1996.
8. Barry Smyth and Mark T. Keane. Retrieval and Adaptation in Déjà Vu, a Case-Based Reasoning System for Software Design. In David W. Aha and Ashwin Ram, editors, *AAAI Fall Symposium on Adaptation of Knowledge for Reuse*, MIT Campus, Cambridge, MA (US), November 1995. American Association for Artificial Intelligence, Menlo Park, CA (US).
9. Willemien Visser. A Tribute to Simon, and Some —Too Late— Questions, by a Cognitive Ergonomist. In Jacques Perrin, editor, *Internationnal Conference on the Sciences of Design*, INSA Lyon, Villeurbanne (FR), 2002.