Retrieving Reusable Design Cases

Exploiting Adaptation Knowledge in Design Reuse

Barry Smyth[1] and Mark T. Keane[2]

[1] Hitachi Dublin Laboratory, 16 Westland Row, Dublin 2, Ireland
[2] Trinity College Dublin, Dublin 2, Ireland

Abstract. Case-based reasoning has been identified as a potentially fruitful candidate technology with which to investigate the development of automated design systems. Two critical stages in case-based design are design retrieval and design adaptation. In the former, designs that can be reused for a new design problem must be located. In the latter, retrieved designs must be modified to meet the specific demands of the new target situation.
In this paper we will address both of these stages in the context of a case-based software design system called Déjà Vu. In particular, it will be argued that the notions of design reusability and adaptability are intricately linked and an approach will be described which allows the adaptation requirements of design cases to be accurately predicted during retrieval and subsequently exploited during adaptation. We argue that this approach benefits from improved retrieval accuracy, flexibility, and greater overall problem solving efficacy.

1 Introduction

Case-Based Reasoning (CBR) is a reasoning method that exploits experiential knowledge, in the form of past cases, to solve new problems [1]. When faced with a new problem, a CBR system will retrieve a case that is similar, and if necessary, adapt it to provide the desired solution. To date there has been some preliminary success in applying the CBR paradigm to complex design tasks ([2], [3], [4], and [5]). Obviously, the success of case-based design (CBD) is critically dependent on the retrieval of a suitable design case; that is, one that can be reused in the target situation. Moreover, the efficiency of case-based methods depends critically on the retrieval of a design that is the easiest, of those available, to reuse. In this paper we view the concepts of reusability and adaptability as closely coupled, and we claim that estimating the adaptation requirements and complexity of cases during retrieval is an critical step in locating reusable designs.

The majority of CBR (and CBD) systems have proven successful in judging the general suitability of cases to new problem situations. However, accurately determining the adaptability or "ease of adaptation" of a design case has proven more difficult because of inherent efficiency problems; how can adaptation be accurately predicted without actually performing the adaptation itself? This has led most researchers to abandon *deep algorithmic* methods of computing case adaptability, in favour of more efficient, albeit less accurate, *shallow heuristic* methods; the hope being that heuristic manipulation of good predictive indices will result in the retrieval of the appropriate case. Typically, these heuristics are designed to measure the semantic similarity between the target and a candidate case, giving preference to those candidates with features that have been observed to yield desirable retrieval results. Unfortunately, such approaches seldom anticipate all adaptation problems and consequently sub-optimal cases are often retrieved.

In this paper we advance a case retrieval mechanism that *can* accurately determine the ease of adaptation of a design case whilst, at the same time, overcoming the efficiency problems that led to the adoption of heuristic methods [6]. The technique uses adaptation knowledge during retrieval to *look ahead* to the adaptation stage, allowing its complexity to be assessed, but without incurring the full cost of adaptation. In addition, we describe a framework for design adaptation that can exploit fully the results of the retrieval stage. Our methods are implemented in Déjà Vu[3], a case-based

[3] Not to be confused with DEJAVU an CBR shell for assisting in mechanical design [7].

reasoning system for real world software design, and we demonstrate our approach using examples from this system.

The next section briefly introduces Déjà Vu and its application domain. Section 3 highlights the essence of retrieval in CBD and demonstrates where traditional approaches may fail to select optimal cases. Then, in section 4, we concentrate on Déjà Vu's retrieval mechanism (termed *adaptation guided retrieval*), which avoids such retrieval failures by using adaptation knowledge to efficiently compute the adaptability of cases; in describing this retrieval mechanism, the nature of adaptation in Déjà Vu is also discussed. Finally, in section 5, we conclude by arguing that our methods benefit from improved retrieval accuracy and flexibility, as well as greater overall problem solving performance.

2 Déjà Vu

Déjà Vu is a CBD system for software design operating in the domain of Plant-Control software [8]. Plant-Control software is concerned with controlling robotic vehicles within a factory or plant environment. In contrast to conventional software domains, the plant-control domain can not be completely formalised; for example, certain timing considerations must be taken into account which can only be estimated on the basis of expert experience and lack any strong causal theory. This weakening of the domain theory suggests case-based methods as a potentially fruitful approach.

One particular set of problems that Déjà Vu has been designed to deal with are those concerned with loading and unloading tasks. For example, Figure 1 illustrates the loading or unloading of, in this case, spools or coils of steel within a steel-mill environment. Using a hierarchical approach

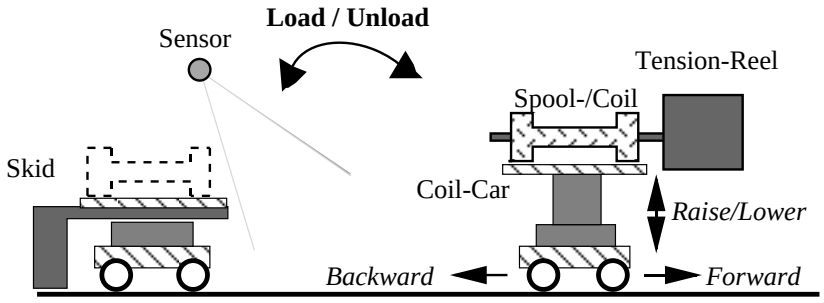

Fig. 1. Load/Unload Plant-Control Tasks

to design, Déjà Vu retrieves a number of design cases at different levels of abstraction. These are adapted to provide solutions to the various sub-tasks of the target problem, the resulting solution segments being integrated into the overall solution on the fly.

Indeed, Déjà Vu's hierarchical problem solving method can be viewed as an integration of decompositional and case-based design techniques; software specifications are decomposed into simpler sub-specifications which may then be solved by the retrieval and adaptation of an appropriate design case. In fact the decomposition of specifications is also addressed by using case-based methods. Déjà Vu's design cases represent not only specific design episodes, but also more abstract, high-level design plans that act as decomposition strategies. Problem solving activity is efficiently co-ordinated using a blackboard architecture with dedicated knowledge sources handling the various problem solving stages of analysis, decomposition, retrieval, adaptation, and re-composition.

3 Retrieving a Reusable Design Case

The primary concern of the retrieval stage[4] is to select a case that is a suitable starting point from which to develop a design solution to the target problem. But what exactly determines whether a given case is suitable or not? In design tasks, this question of suitability is strongly linked to the notion of reusability. In turn, reusability is related to such criteria as adaptability, maintainability and serviceability. In particular we view adaptability as a crucial factor in determining reusability; certainly, to apply a design case to a new situation it must be possible to adapt it. Furthermore, the chosen case should not only be adaptable but should also be the easiest of those available to adapt.

Many approaches to retrieval tend to avoid measuring adaptability, selecting cases on the basis of semantic similarity instead. It will be demonstrated here that such approaches can lead to cases that are not even adaptable, nevermind reusable. We will argue that a more perspicacious approach is to consider directly the adaptability of cases. This is precisely the motivation behind Déjà Vu's retrieval method, which uses adaptation knowledge to make efficient and accurate predictions about the adaptation needs of candidate design cases.

3.1 Conventional Approaches

Traditionally, researchers have side-stepped the notion of adaptability, preferring to exploit semantic similarity as a more tractable answer to the question of reusability. Exactly how a candidate design case will be adapted is very much ignored, retrieval efficiency being chosen in favour of accuracy. The rationale implicit in these methods is that the case whose specification is most semantically similar to the targets will also be the "most useful" case and will require the least adaptation ([11] and [12]).

While such traditional retrieval techniques can produce efficient retrieval results this rationale on which they are based may not be fully justified, and this may ultimately lead to a sub-optimal adaptation stage. That is, the most similar case to the target problem may not be the most useful, or indeed, the easiest to adapt. Semantic similarity does not guarantee the best results. Two cases could be equally similar to a target problem on this measure but one could be adapted with ease while the other may be considerably harder or even impossible to adapt.

To compensate for these problems, many researchers have argued that other factors as well as semantic similarity need to be used in retrieval ([13], [3], and [14]), the spirit of these approaches being that *all mappings are not equal*.

For example, Kolodner [13] has argued that some mappings found between a target problem and a candidate case should be preferred over others if they exhibit certain characteristics; for instance, if a match is more *specific* or *goal-directed* it should be preferred. In particular, Kolodner also argues that the *ease-of-adaptation* of a match should result in it being preferred over other matches which are indicative of more difficult adaptations. Similarly, in KRITIK Goel [3] also favours candidate design cases which are easier to adapt by preferring matches which satisfy the functional specifications of the desired, target design. Birnbaum et al. ([14]) propose a system that learns to index cases on the basis of their adaptability, overriding semantic similarity where appropriate. During problem solving certain feature combinations are identified as particularly problematic and cases with such features can be avoided in future problem solving episodes.

[4] Retrieval can be viewed as a two stage process. First, the filtering stage identifies a small number of candidate cases that are deemed to be contextually relevant to the target. This eliminates many irrelevant cases from the more computationally expensive selection stage. The selection stage performs a detailed analysis between the target and each of these candidates. During this analysis, a set of correspondences or mappings is established between the target and the candidates (see [9] and [10]). In general these mappings are used to determine a measure of similarity between the cases and form the basis of the subsequent adaptation process.

In all of these approaches the quality of a candidate case is based on the presence or absence of certain features which are pre-classified or weighted as important with respect to retrieval. The relationship between specification features and the subsequent adaptation phase is very much ignored. Consequently, cases are *still* selected on the basis of an "educated guess" rather than through any real insight into their adaptation requirements.

3.2 Semantic Similarity *vs.* Adaptability

To demonstrate the type of retrieval inaccuracies that emerge from traditional similarity-based methods consider the following very simple example from Déjà Vu's plant-control domain. It will be shown that for a given target specification, the case selected for retrieval differs depending on whether semantic similarity or adaptability is used as a measure of suitability. In the example we will explain how semantic similarity can mask the adaptation requirements of a candidate, forcing the retrieval of a case that is difficult to adapt. Measuring adaptability however results in the selection of a case that is much easier to adapt.

A piece of software is needed to move a buggy (a vehicle) forward to a tension-reel in two speed mode[5]. Two candidate cases are located as relevant. The first moves the buggy forward to the tension-reel but in one speed mode[6], and the second raises a lifter platform in two speed mode. Figure 2 shows the mappings that are generated between the target and these candidates during the mapping phase. The same mappings are established irrespective of whether semantic similarity or adaptability is being used as the retrieval criterion, but we will se that the quality of these mappings varies considerably. Measuring the semantic similarity between the target and candidates, the first

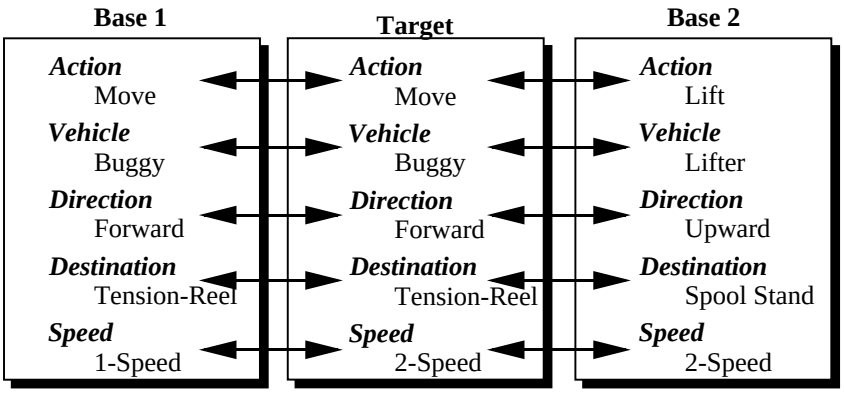

Fig. 2. An example set of mappings.

case (Base 1) is retrieved. This case matches exactly on all features except speed and so scores very highly. The second case (Base 2) however differs on all features but speed and so does not score nearly as high. There is a problem however. The retrieved case is more difficult to adapt than the one rejected. This is because modifying the speed of a case is far more difficult than modifying attributes such as direction. The former requires that new design components be added or deleted whereas the latter requires only parametric changes to existing design elements.

If instead of using semantic similarity adaptability is used as the retrieval criterion then a very different result is obtained. Now the second case is retrieved. The discrepancies between the direction,

[5] Two speed mode indicates that first the vehicle moves at its higher speed setting, at a certain point it then slows down, and finally on reaching its destination it stops.

[6] In one speed mode the vehicle will move at its slower speed until reaching its destination, where it stops.

vehicle, and destination of the target and second candidate are recognised as relatively easy to adapt when compared to the speed changes necessary if the first candidate was retrieved. The crucial point here is that to measure adaptability some form of adaptation knowledge is needed to construct and grade the mappings, knowledge about the capabilities of the adaptation stage. Using this knowledge the retrieval stage can then determine that, for example, changing the speed of a case is difficult whereas altering the direction of motion or destination or vehicle is considerably easier.

4 Adaptation Guided Retrieval

In order to guarantee the retrieval of a case that is the easiest to adapt, the retrieval mechanism must give explicit consideration to *how* design cases will be adapted. To achieve this without actually performing full adaptation, Déjà Vu uses adaptation knowledge during retrieval to predict the adaptation requirements of a candidate design case. During retrieval, as mappings are formed between the target and candidate features, it is necessary to predict their adaptation requirements. In fact, a target feature X should only be matched to a candidate feature Y if there is evidence (in the form of adaptation knowledge, see section 4.1) that Y can be adapted to give X.

We can think of the processes of retrieval and adaptation as searching of two distinct search spaces, the retrieval space (or design specification space) and the adaptation space (or design transformation space). Elements of the retrieval space are matches between target and candidate features. Elements of the adaptation space are the adaptation operators needed to transform the candidate design into the desired target design. To determine the adaptability of a candidate design case, a measure of the closeness of the target and candidate in the adaptation space is needed.

Conceptually, Déjà Vu links the retrieval space and the adaptation space using adaptation knowledge (see Figure 3). In this way it is possible to determine how elements of the retrieval space relate to elements of the adaptation space, and so the matches formed during retrieval can be associated with adaptation operators that will be needed during the subsequent adaptation stage. Thus, complex adaptation requirements can be predicted by comparing the features of the target and candidate cases.

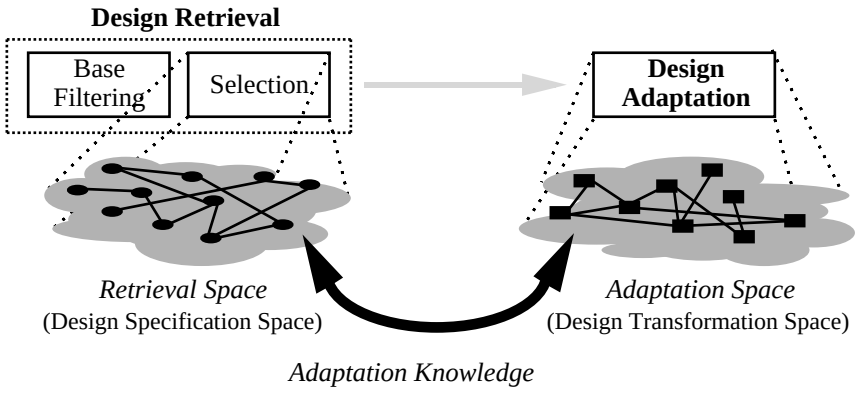

Fig. 3. Adaptation knowledge links the retrieval and adaptation spaces.

4.1 Déjà Vu's Adaptation Knowledge

Déjà Vu uses an adaptation scheme that facilitates both specific local modifications, through the action of *adaptation specialists*, as well as global conflict resolution, via *adaptation strategies*. As such adaptation knowledge is captured as a set of specialists and a set of general strategies.

Adaptation specialists correspond to packages of design transformation knowledge each concerned with a specific adaptation task. Each specialist can thus make a specific local modification to a retrieved case. During adaptation many specialists will act on the retrieved design to transform it into the desired target design. Thus, through specialist activity, the differences between the retrieved case and the target are reduced in a fragmentary fashion. As well as procedural knowledge each specialist also has declarative knowledge describing its particular adaptation capability. In this way specialists are organised in terms of the modifications they are designed to carry out.

For example, in the plant-control domain, one common difference between a retrieved case and a target problem is that the speed of the target specification may differ from that of a retrieved case. To cater for this situation Déjà Vu uses a dedicated *speed specialist* (see Figure 4) which can satisfy the speed requirements of the target by modifying those of the retrieved case. In the course of adapting

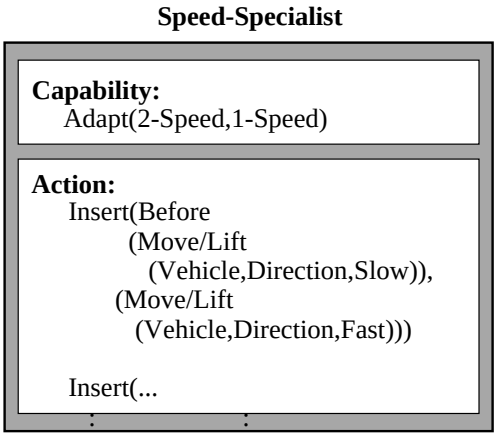

Fig. 4. An example adaptation specialist.

a retrieved design case it is possible that interactions will arise within the modified design solution. This is because specialists are not designed to consider the modifications made by other specialists and so interactions that occur between specialists go unchecked and may ultimately lead to design error and failure. In the past, the resolution of such interactions has been one of the stumbling blocks of many planning and automated design systems [15]. Déjà Vu attempts to overcome this problem by using an efficient scheme of interaction representation and resolution. Using a set of adaptation strategies, Déjà Vu can detect and repair many conflicts that might arise. Strategies are organised in terms of the interactions they resolve and each is indexed by a description of the type of failure it can repair. Of course each strategy also has an associated method of repair for resolving the conflict in question. As an example, one very common type of interaction occurs when the effect of some event prevents the occurrence of some later event. Figure 5 depicts this situation; some goal event (1) is prevented by the disablement of one of its preconditions (2), the precondition having been blocked by some earlier event (3) causing a conflicting state (4). This *blocked pre-condition* interaction can be repaired in a number of ways. For instance, an event could be added before the blocking event (3) which prevents its blocking effect. The blocked pre-condition adaptation strategy contains a description of this situation as well as the appropriate repair methods.

4.2 Specialist Associations

Déjà Vu constructs mappings between target and candidate features if and only if there is evidence that the differences that they entail can be catered for during adaptation. Déjà Vu's approach is

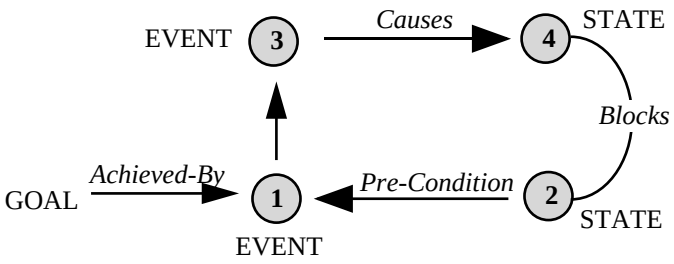

Fig. 5. The "blocked pre-condition" configuration.

based on the fact that the mappings established between the candidate and target are predictive of the differences that exist between the retrieved design and desired target design specification. Identical mappings suggest candidate design elements which can be transferred intact to the target. On the other hand, non-identical mappings are indicative of candidate design elements that will need to be adapted.

In the example of section 3.2 a non-identical mapping was formed between the single speed feature of the first candidate case and the two speed feature of the target. This mapping served to point out that the candidate design solution required a speed modification. To form such a mapping, Déjà Vu needs evidence that the speed modification is possible. We have said that this evidence is provided by the adaptation knowledge. More precisely, this evidence exists in the form of specialist capability information. During case retrieval, sets of mappings are matched against the capability descriptions of specialists.

Returning to our perspective on the coupling of the retrieval and adaptation spaces we can see that in this situation the mapping between the speed features, an element of the retrieval space, is indeed linked to an appropriate element of the adaptation space, the speed specialist. And, this linkage is provided by adaptation knowledge in the form of the speed specialist's capability information (see Figure 6). To facilitate the efficient location of the appropriate specialists, the

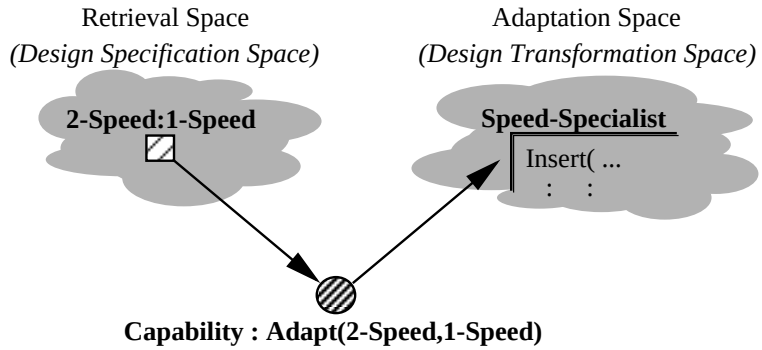

Fig. 6. Linking the retrieval and adaptation spaces.

capability descriptions themselves are in the form of generalised groups of mappings.

4.3 Strategy Associations

Unfortunately, to accurately predict the adaptability of a case it is not sufficient to examine the differences in isolation. As was mentioned in section 4.1 interactions can occur which will lead

to adaptation failure. These interactions arise due to the context sensitive nature of adaptation specialists and dependencies between design elements or features. For example, the existence of a dependency between the speed of a case and the power availability of a vehicle may lead to an adaptation failure if the speed is modified during adaptation. Increasing the speed of a case means increasing the power consumption of the vehicle. If the vehicle has only limited power available then obviously the adaptation may produce a design which will fail due to a lack of power. In fact this is an example of the block-precondition failure described in section 4.1; the effect of increasing the speed of the design has blocked the pre-condition of motion, the availability of power.

Adaptation strategies are used during retrieval to predict such interactions by considering the consequences of adaptations. This is achieved by modelling the dependencies that exist between design elements, using structures very similar to *influence graphs* ([16]), as well as the capabilities of adaptation operators. Very briefly, an *influence relation* is a qualitative causal relationship between two domain elements. It specifies that one element (the *influencer*) effects another (the *influenced*) in some way. The mode of influence can be either positive (+) or negative (-). A positive influence means that a change in the influencer entails a corresponding change in the influenced. For example, speed and power consumption are connected by a positive influence relation from speed to power consumption; an increase in speed leads to an increase in power consumption. A negative influence means that a change in the influencer leads to a qualitatively opposite change in the influenced. For example, power consumption exerts a negative influence on power availability; an increase in power consumption causes less power to be available. Using graphs of these influence relations a qualitative model of the dependencies between domain elements can be built up. With these graphs it is possible to describe both the desired effects and side-effects of specialists. For example, the speed specialist changes the speed of a case. According to the influences above it also effects the power consumption and power availability of the case.

Strategies are indexed into the domain knowledge-base by sets of influence relations. During retrieval the formation of specialist associations activates a set of influences that capture their intended effect. In turn these influences activate relevant strategy descriptions, indicating possible interaction problems. The retrieval context is used to instantiate these strategies which are then associated with the problematic specialists and mappings. In this way, during retrieval, interactions can be predicted and repairs scheduled.

4.4 An Example Retrieval Scenario

As an example, let us return to the problem of section 4.1 which was to design a two speed movement case. Let us consider what happens during retrieval as the target is compared to the first candidate (the one speed case).

The mappings between the speed features of the candidate and target predict the need for a speed modifying specialist. Once a specialist has been found the mapping can be established. In addition, a measure of the quality of the mapping is based on the computational complexity of the specialist. But, what about predicting interactions? In particular, how can the power availability problem be foreseen and an appropriate strategy identified to effect its repair ? Figure 7 illustrates the important states and events during this retrieval. The target problem is concerned with moving a two speed buggy to a tension-reel (1). A pre-condition of movement is that power be available (2). The speed specialist will cause the speed of the case to be upgraded from one speed mode to two speed mode. The influence that this increase in speed (3) exerts on power consumption (4) leads to the disablement or blocking of the power availability pre-condition. This configuration (enclosed portion of Figure 7) is the description for the blocked pre-condition strategy described in section 4.1. After instantiating the strategy in the current base and target contexts it is associated with the speed specialist. During adaptation the action of the speed specialist is augmented with the repair action of this adaptation strategy; in this case adaptation consists of changing the speed of the case after upgrading the power capacity of the buggy.

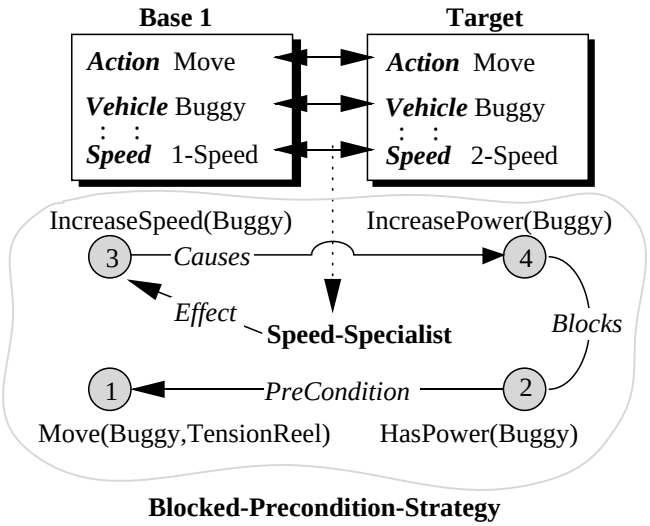

Fig. 7. An example failure configuration.

Although simple, the example above does highlight the key features of our approach; the relevant local and global adaptation knowledge (in the form of specialists and adaptation strategies) is efficiently assembled during retrieval facilitating an accurate estimate of the adaptation requirements of a candidate design case[7]. The result is the selection of not just a suitable design (together with some numerical quality value), but also the formation of a structured semantic representation, in the form of adaptation specialists and strategies, that captures the precise nature of the similarities and dissimilarities between the target and candidate design specifications.

5 Conclusions

The main thrust of this paper has centred on the description of an important issue in CBD, that of case retrieval, and has also touched on the area case adaptation. In particular, it concentrated on a critical case selection criterion, that of adaptability, which we claim is an important step in selecting appropriate reusable designs. Overall, as well as benifiting from improved retrieval accuracy, we can expect to witness both improved problem solving efficiency and competence together with greater flexibility.

Our approach favours the retrieval of a case that requires minimal adaptation. This is in contrast to other CBD systems that do not directly couple retrieval and adaptation and, as such, can only estimate the reusability of a given design in terms of its semantic similarity to the new target demands – which is often not a very accurate measure of reusability. By directly predicting the adaptation requirements of a candidate design case much of the "guess work" can be taken out of retrieval; if a case exists that can be adapted, we can expect that our retrieval mechanism will find it.

The retrieval of the most adaptable case should result in an optimal adaptation stage. In addition, retrieval now carries out the preliminary adaptation work by identifying the specialists and strategies that will be necessary during the adaptation stage. Therefore, further efficiency gains can

[7] A full description of how this complexity estimate is computed is beyond the scope of this paper. Very briefly, it is a function of individual complexity estimates for each specialist and strategy used during retrieval. The important point here is that the relevant adaptation knowledge *can* be assembled.

be expected during adaptation by exploiting this additional retrieval information. In general, retrieval complexity is reduced by structuring design adaptation knowledge in a manner that permits the efficient identification of the appropriate specialists and strategies, and any additional retrieval expense is offset by improved adaptation efficiency .

In addition, greater retrieval flexibility is also achieved. With conventional approaches, changes to the adaptation capabilities of a system will not be immediately reflected in the retrieval preferences of the system. Instead changes must be made to the retrieval heuristics to capture the new adaptation possibilities. In contrast, because the retrieval and adaptation stages are directly coupled in Déjà Vu, any changes to its adaptation capabilities *will* be immediately available to the retrieval system; the altered adaptation knowledge itself is used explicitly during retrieval.

Moreover, the representational requirements of the approach are domain independent and thus facilitate the adoption of the technique across a range of CBD application domains. Already Déjà Vu has been used to investigate other design tasks. As well as plant-control software, Motif graphical user interface design has been investigated. Initial results are very encouraging with the same retrieval and adaptation techniques being successfully transferred to this, quite different, software design domain. Finally, our current work is concerned with providing a concrete empirical demonstration of the effectiveness of Déjà Vu's retrieval and adaptation mechanisms.

References

1. J. Kolodner: Case-Based Reasoning Morgan Kaufmann (1993)
2. E. Domeshek and J. Kolodner: A Case-Based Design Aid for Architecture. Artificial Intelligence in Design '92. Kluwer Academic (1992) 479–516
3. A.K. Goel: Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph.D. Dissertation. Ohio State University (1989)
4. K. Hua, I. Smith, B. Faltings: Integrated Case-Based Building Design. Preprints of the First European Workshop on Case-Based Reasoning (1993) 246–251
5. M. L. Maher, D. M. Zhang: CADSYN: A Case-Based Design Process Model. AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 7(3). (1993) 97–110
6. B. Smyth, and M. T. Keane: Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval. Proceedings of the First European Workshop on Case-Based Reasoning. Springer-Verlag (Lecture Notes in Artificial Intelligence) (1994)
7. T. Bardasz and I. Zeid: DEJAVU: A Case-Based System to Aid Novice Designers. AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 7(3). (1993) 111–124
8. B. Smyth, and P. Cunningham: Deja Vu: A Hierarchical Case-Based Reasoning System for Software Design. Proceedings of the 10th European Conference on Artificial Intelligence. Wiley (1992) 587–589
9. M.T. Keane: Analogical problem solving. Chichester: Ellis Horwood (1988)
10. D. Gentner: Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7. (1983) 155–170
11. R. Bareiss, J.A. King: Similarity Assessment in Case-based Reasoning. Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann. (1989) 67–71.
12. P. Thagard, K.J. Holyoak, G. Nelson, D. Gochfeld: Analog Retrieval by Constraint Satisfaction. Artificial Intelligence, 46. (1990) 259–310
13. J. Kolodner: Judging Which is the "Best" Case for a Case-Based Reasoner. Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann. (1989) 77–84
14. L. Birnbaum, G. Collins, M. Brand, M. Freed, B. Krulwich, and L. Pryor: A Model-Based Approach to the Construction of Adaptive Case-Based Planning Systems. Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann (1988) 191–198
15. J. Hendler, A. Tate, M. Drummond: AI Planning: Systems and Techniques. AI Magazine, 11. (1990) 61–77
16. K.P. Sycara, D. Navinchandra: Influences: A Thematic Abstraction for Creative Use of Multiple Cases. Proceedings of the Case-Based Reasoning Workshop. Morgan Kaufmann (1991) 133–144

This article was processed using the LaTeX macro package with LLNCS style