Case Adaptation & Reuse in Déjà Vu

Barry Smyth[1]

Abstract. It is generally agreed that one of the most challenging issues facing the case-based reasoning community is that of adaptation. To date the lion's share of CBR research has concentrated on the retrieval of similar cases, and the result is a wide range of quality retrieval techniques. However, retrieval is just the first part of the CBR equation, because once a similar case has been retrieved it must be adapted. Adaptation research is still in its earliest stages, and researchers are still trying to properly understand and formulate the important issues. In this paper I describe a treatment of adaptation in the context of a case-based reasoning system for software design, called Déjà Vu. Déjà Vu is particularly interesting, not only because it performs automatic adaptation of retrieved cases, but also because it uses a variety of techniques to try and reduce and predict the degree of adaptation necessary.

1 INTRODUCTION

Déjà Vu is a case-based system for designing industrial device control software. Its main application domain is the control of robotic vehicles and machinery in steel-mills. These mills contain various types of vehicles (called Coil-Cars), loading bays (called Skids), and pressing devices (called Tension-Reels) which have a wide range of device configurations and operational settings. Important tasks include the loading and unloading of coils and spools of steel and the transport of coils by vehicles (see Smyth & Cunningham, 1992; Smyth & Keane, 1994).

The main objective of this paper is to outline the adaptation process in Déjà Vu. However, before this, the next section briefly explains how Déjà Vu tries to minimise the need for adaptation by supporting the reuse of multiple cases and by using a sophisticated adaptation-guided retrieval technique. The structure of adaptation knowledge is described in section 1.3, and an example of how this knowledge is used during adaptation is the topic of section 1.4. Section 1.5 looks at some related work, focusing not just on adaptation but also on retrieval, and the reuse of multiple cases. Section 1.6 outlines the strengths and limitations of the approach taken, and section 1.7 concludes with a list of further issues and questions.

2 AVOIDING ADAPTATION IN DÉJÀ VU

In may seem rather strange to begin a paper on adaptation by describing how a system avoids it, but this idea is fundamental to Déjà Vu's philosophy. The standard, "single-shot" model of CBR (i.e., reusing only a single case) has been extended in two ways: (1) Déjà Vu uses a technique called *hierarchical case-based reasoning* (HCBR) to support the reuse of multiple cases; (2) An retrieval approach called *adaptation-guided retrieval* (AGR) which ensures that adaptable cases are always retrieved. Both of these extensions minimise the amount of adaptation that is necessary in a typical problem solving session.

2.1 Hierarchical CBR

New target problems are solved by retrieving, adapting and combining solutions from a number of relevant cases, at varying levels of abstraction. In effect, Déjà Vu integrates decompositional and case-based design processes to solve complex problems in a top-down fashion. Complex problems are stored as hierarchical collections of cases, and individual cases describe part of a more complex solution at some given level of abstraction.

There are two principal case types, abstract or detailed. Abstract cases contain high-level design solutions. In fact, an abstract solution can be viewed as a set of sub-problems so that the retrieval and adaptation of an abstract case actually results in the definition of new target sub-problem specifications which must be solved by further CBR cycles. In this way abstract cases are used to decompose complex problems into simpler problems. In contrast, detailed design cases contain actual plant-control software code, and the retrieval and adaptation of one of these cases completes part of the overall target problem solution. (see also Smyth & Cunningham, 1992; and for related work, Maher & Zhang, 1993; Branting & Aha, 1995)

HCBR reduces the adaptation load by reusing the best parts of many larger solutions rather than insisting on the reuse of a single monolithic solution. Indeed, without HCBR it would not be possible to solve complex plant-control problems except by using extremely large case-bases or very sophisticated adaptation facilities.

2.2 Adaptation-Guided Retrieval

The AGR philosophy states that at all times we should try to retrieve a case which is not only adaptable, but which is the easiest of those available to adapt. This is made possible in Déjà Vu by using adaptation knowledge during retrieval to determine the adaptation requirements of cases. In short, a case is only retrieved if there is evidence, in the form of adaptation knowledge, that it can be adapted to fit the target problem, and if its predicted adaptation cost is the best available. Furthermore, AGR works without overburdening the retrieval stage (see Smyth & Keane, 1994).

Experiments demonstrate that AGR does indeed significantly reduce the adaptation load when compared to more standard similarity-based retrieval methods. Overall costs are further reduced because one of the side-effects of

[1] Department of Computer Science,
University College Dublin, Belfield, Dublin 4, IRELAND.
Email: bsmyth@cslan.ucd.ie

AGR is the identification of relevant adaptation knowledge, thereby setting up the adaptation stage (see Smyth & Keane, 1995, and for related work Purvis & Pu, 1995)

3 ADAPTATION KNOWLEDGE

In case-based reasoning systems adaptation knowledge is needed to make changes to the solutions of retrieved cases. Plant-control software solutions are represented as graph structures. Individual nodes correspond to solution commands and the connections between nodes encode sequential and parallel control flow.

Adaptation knowledge must specify how, and under what conditions, these solution graphs are to be modified. Déjà Vu uses 3 basic adaptation operators to perform node substitution, node insertion, and node deletion. Adaptation knowledge contains sequences of these operators. There are two basic forms of adaptation knowledge. The first is an *adaptation specialist*, these are designed to perform specific adaptation tasks. The second form is an *adaptation strategy*, and these encode more general modifications, and can be used to co-ordinate specialists as well as resolve adaptation conflicts. Both types of adaptation knowledge are used during retrieval as well as during adaptation, and both can be used to modify detailed design cases as well as abstract cases.

3.1 Adaptation Specialists

The most common reason for adaptation is that there are differences between the target problem and base case. For example, different entities may be used, or different tasks may be performed, or different operational constraints may be valid. Typically, to compensate for such, the base solution will have to be modified by making various substitutions or structural changes, and it is these type of adaptations that specialists are designed to handle.

For example, one common plant-control task is the MOVE task, in which a vehicle (COIL-CAR) is moved from one location to another using either 1-SPEED or 2-SPEED motion. In 1-SPEED motion the vehicle moves at its slow speed until reaching a destination, whereupon the activation of a sensor stops the vehicle. In 2-SPEED motion the vehicle initially moves at its fast speed, at a certain distance (vehicle dependent) before the destination it slows down, and finally on reaching its destination it stops. When solving MOVE problems, cases will often be retrieved that differ from the target in terms of their destination locations, or their vehicles, or their speed of motion. Each of these differences will require certain types of adaptation. For instance, if the target problem specifies a different destination, then the command in the base solution which checks if the vehicle has reached its destination must be adapted. In addition, the destination change may alter the direction of travel, and hence direction adaptations may also be required. If there is a speed difference, then this can mean adding or removing nodes from the solution depending on the type of speed difference.

3.2 Adaptation Strategies

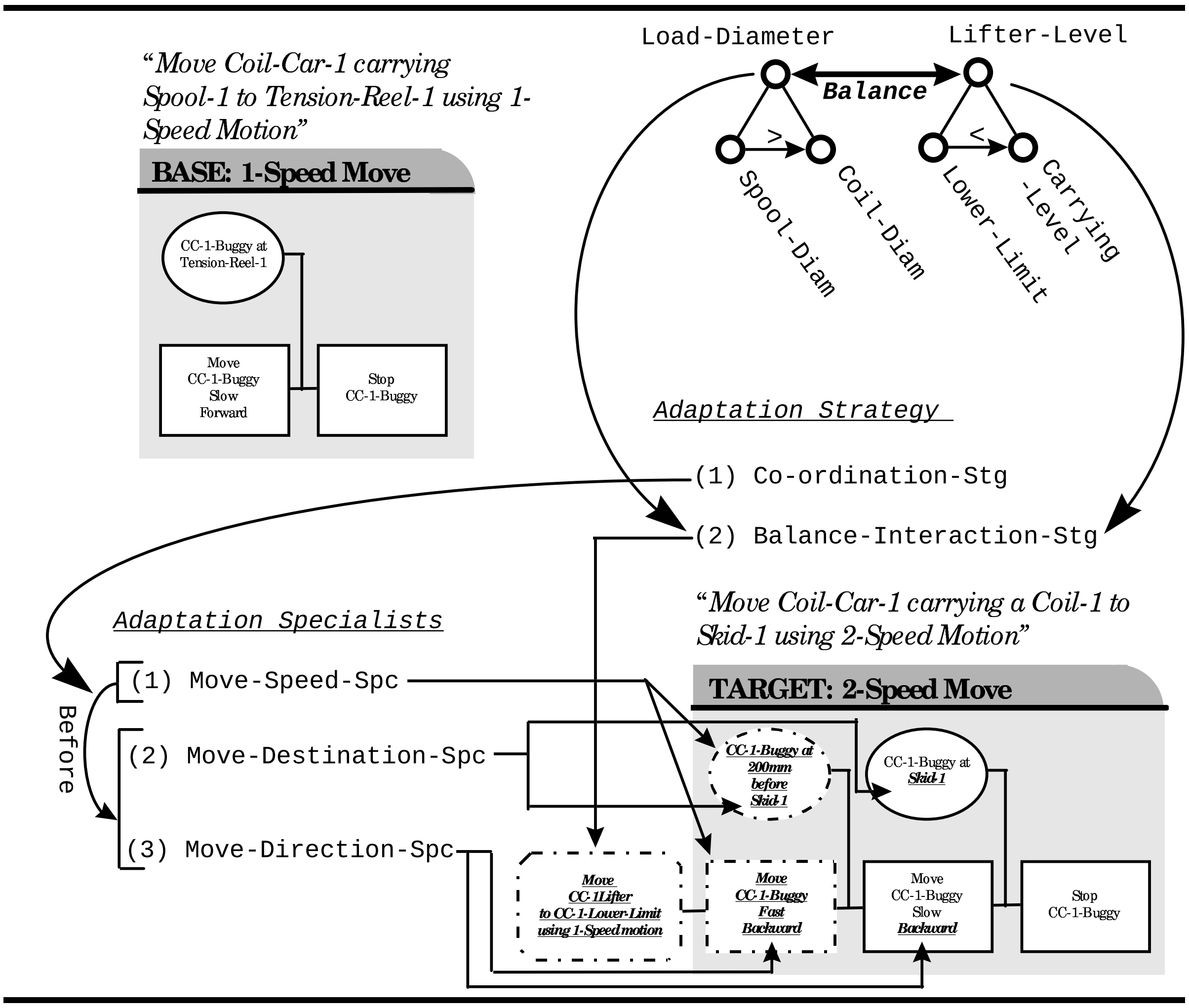
Specialists make localised changes to solutions, and they are blind to the changes made by other specialists. Problems arise when the actions of specialists conflict to invalidate their combined actions; similar interaction and conflict problems have plagued the planning community for decades. Some way of co-ordinating specialists and resolving their implicit conflicts is needed. This is the role of adaptation strategies.

Co-ordination Problems: Often a retrieved case will need to be adapted by more than one specialist, each working on different parts of the solution. This picture is complicated by the fact that sometimes the operation of one specialist depends on an adaptation made by another. A co-ordination mechanism is needed to ensure that this condition is satisfied during the adaptation stage. This is the role of a CO-ORDINATION strategy. Its job is to recognise ordering constraints between relevant specialists and to use these constraints to compute an appropriate activation schedule.

Interaction Problems: Sometimes conflicts between specialists may be so serious that they cannot be resolved by simply co-ordinating and scheduling the action of specialists. For instance, even when there are no immediate conflicts between specialists, it can happen that through the action of one specialist, a totally new conflict is introduced, which of course must then be resolved. For instance, a *balance-interaction* conflict is said to occur when the value of one feature is proportionally dependent on another, because adapting one feature may have an adverse effect on the validity of the other. For example, when moving a COIL-CAR across the factory floor the height of its lifter platform must be adjusted to accommodate the load being transported. There is a balance condition between the height of the lifting platform and the diameter of the load. In general, empty coils (spools) can be carried at the COIL-CAR'S carrying-level height, while the larger coils must be carried at the COIL-CAR'S lower-limit height setting. If this balance is not properly maintained then a failure may occur (the COIL-CAR may collide with an overhead obstacle). The repair action for this conflict is to make sure that a compensating adaptation is performed to restore the balance condition. The BALANCE-INTERACTION strategy uses this approach.

4 AN ADAPTATION EXAMPLE

It has been mentioned in section 2 that the result of adaptation-guided retrieval is, not only the selection of a suitable base case, but also a list of the adaptation specialists and strategies relevant to the adaptation of this case. Previous work has described in detail how retrieval operates and how this relevant knowledge is located (Smyth & Keane, 1993, 1995). Here we will explain, through a worked example, how specialists and strategies are used during adaptation to modify the base solution to fit the target problem.



"Move Coil-Car-1 carrying Spool-1 to Tension-Reel-1 using 1-Speed Motion"
BASE: 1-Speed Move
CC-1-Buggy at Tension-Reel-1
Move CC-1-Buggy Slow Forward
Stop CC-1-Buggy
Load-Diameter
Lifter-Level
Balance
Spool-Diam
Coil-Diam
Lower-Limit
Carrying -Level
Adaptation Strategy
(1) Co-ordination-Stg
(2) Balance-Interaction-Stg
"Move Coil-Car-1 carrying a Coil-1 to Skid-1 using 2-Speed Motion"
Adaptation Specialists
(1) Move-Speed-Spc
(2) Move-Destination-Spc
(3) Move-Direction-Spc
Before
TARGET: 2-Speed Move
CC-1-Buggy at 200mm before Skid-1
CC-1-Buggy at Skid-1
Move CC-1Lifter to CC-1-Lower-Limit using 1-Speed motion
Move CC-1-Buggy Fast Backward
Move CC-1-Buggy Slow Backward
Stop CC-1-Buggy

Figure 1. An example adaptation scenario.

Figure 1 illustrates the adaptation of a 1-SPEED MOVE case to solve a 2-SPEED target problem. Both problems share certain features, such as the type of task and the vehicle used, but they differ in terms of speed requirements, the destination location, and the content being carried. These differences signal the need for a number of specialists: (1) The speed specialist is need to transform the single speed solution into a two speed solution by adding extra nodes as shown (the two new noes have a dashed outline); (2) the destination specialist is needed to substitute the target destination (SKID-1) for the base destination (TENSION-REEL-1) in the sensor-check nodes that determine the distance of the vehicle from its destination; (3) the direction specialist is needed because, since the vehicle is travelling to a different destination, it is also travelling in a different direction, and so the direction parameter of each move command must be adapted. Note that the substitutional changes are shown in Figure 1 by underlining the new terms and using italics in the target solution graph.

On their own these specialists do not fully adapt the base solution. First of all co-ordination is needed between the speed specialist and the destination and direction specialists. The speed specialist adds new solution nodes, and these nodes will be later modified by the destination and direction specialists. So we must ensure that the speed specialist performs its actions first. The need for this type of co-ordination is recognised and performed by the CO-ORDINATION strategy because there is a BEFORE relation between the dependant specialists.

Another interaction problem exists because of the difference in content between the target and base. As explained above there is a balance condition between the content being transported and the vehicle's lifting platform height. The base case carries a small diameter empty spool but the new target solution will carry a large diameter coil. Therefore, we must ensure to lower the lifting platform height to restore the balance condition. This means that a new node has to be added to the target solution to handle this lowering task before the main MOVE solution proceeds an abstract node is added as shown that specifies a new LIFT sub-problem (this new node is drawn with a dashed outline).

5 DISCUSSION & COMPARISONS

The main goal of the workshop is to:

"… develop a framework of design options in adaptation. This will be done in a bottom up way, by

asking system developers to identify and rationalize common and different features, shortcomings and strengths of their systems"

To promote an organised exchange of ideas the workshop papers have been divided into a number of clusters dealing with specific adaptation-related themes. In this section I will address two of these themes and compare Déjà Vu to similar systems in the appropriate clusters.

5.1 Abstraction

In section 2 of this paper I described how an important feature of Déjà Vu is its hierarchical CBR method which makes use of case hierarchies during problem solving. In particular, each problem is represented in the case-base as a hierarchy of cases at various levels of abstraction. Currently the system makes use of explicit levels of abstraction by using an abstraction vocabulary. This vocabulary extends the primitive solution operator set to include abstract operators, which are then used in the abstract cases. In this way Déjà Vu's case hierarchies share much in common with the type of plan hierarchies that are automatically built by the PARIS system (Bergmann & Wilke, 1995, 1996). In particular, both systems make use of abstract knowledge that is represented as actual high-level solution code.

The EADOCS system (Netten & Vingerhoeds) also uses abstraction knowledge, however, unlike Déjà Vu and PARIS this knowledge is not stored with the structure of abstract cases. Instead a separate knowledge source is used to specialise design problems to formulate more tractable sub-problems. Thus, abstraction knowledge is explicitly encoded within a set of specialisation rules. Moreover, in EADOCS a static set of specialisation knowledge seems to be encoded within fixed decomposition strategies thus limited the flexibility of specialisation.

Of course the main reason that abstraction is used at all is to help reduce the adaptation overhead. In the case of Déjà Vu and PARIS complex problems are more easily solved at high-levels of abstraction, with the resulting high-level solutions leading to the formulation of lower-level sub-problems during decomposition.

The main condition that must be satisfied in order to use Déjà Vu hierarchical CBR method is that the domain must be decomposable. A similar condition must be satisfied if the PARIS approach is to be used. However, an advantage that PARIS has over Déjà Vu is that it has the ability to automatically produce abstract cases. Déjà Vu does not have this facility and thus the case hierarchies must be carefully hand-coded, adding to the overall knowledge acquisition cost.

5.2 Decomposition & Incremental Adaptation

In earlier sections I have described how Déjà Vu uses a combination of decomposition and adaptation to reuse cases to solve new target problems. One of the advantages of Déjà Vu's decomposition technique is that it does not relay on static decomposition knowledge. Instead, the case hierarchies drive the decomposition process, and decomposition is a direct result of the retrieval and adaptation of abstract cases. In this context Déjà Vu is similar to PARIS (Bergmann & Wilke, 1995, 1996) because it permits the explicit reuse of previous problem decompositions (the abstract case solutions) and it is these adapted decompositions that guide the decomposition process. However, an important difference between Déjà Vu and PARIS is that Déjà Vu refines its abstract solutions by the further retrieval and adaptation of more concrete cases, while PARIS uses search-based refinement methods. In other words, when Déjà Vu retrieves and adapts an abstract case, thereby introducing addition sub-problems that must solved, it goes on to retrieve and adapt new cases with which to solve these sub-problems, rather than using a search-based solution approach.

Purvis and Pu (1995, 1996) describe a system called COMPOSER which also reuses multiple cases, where decomposition is a direct consequence of the way that different cases match different parts of the target problem; these partially matching cases are all reused. Like Déjà Vu, decomposition is dynamic, and depends predominantly on the current case-base organisation and target problem structure, however the explicit reuse of abstract cases (or previous problem decompositions) does not directly occur.

Netten & Vingerhoeds (1996) also advocate the use of problem decomposition during design in the EADOCS system. However, the decompositional component of EADOCS is limited by the use of static decomposition methods.

5 STRENGTHS & LIMITATIONS

Hierarchical CBR and AGR both reduce the adaptation load by reusing and combining optimal cases. Because of this it is possible to encode the required adaptation knowledge as a collection of about 30 - 60 domain specific specialists and much smaller number of general repair strategies (about 4 strategies are usually used).

Currently, Déjà Vu has been validated on a diversity of plant-control tasks covering a wide range of steel-mill configurations and layouts. In addition, preliminary studies have been carried out in alternative domains which suggest that the idea of characterising adaptation knowledge as specialists and strategies is one that will successfully transfer to many other application areas; to date we have also looked at graphical-user interface design. Moreover, it should be noted that while adaptation specialists will tend to change from one domain to another, the adaptation strategies should be reusable because they encode very general types of repair knowledge.

6 FURTHER ISSUES & QUESTIONS

- Obviously there is trade-off between the case-base and the adaptation knowledge, in the sense that adaptation allows us to bridge gaps in the case-base and vice versa. Should adaptation have limited scope in CBR? If so, how can we discuss & characterise these limits?
- Is there a correlation between domain characteristics and the type of adaptation that is supported by a system, or the type of adaptation knowledge that is used? What sort of predictions can we make about the type of adaptation that is most useful in a given domain or for a specific task?

- How can we measure the coverage of adaptation knowledge? For example, in Déjà Vu part of the scaling-up problem is simply recognising that sufficient adaptation knowledge has been encoded, or conversely determining that additional knowledge is needed. Is there any way that we can monitor and guide the acquisition of adaptation knowledge?
- Is it always possible to predict the cost of particular types of adaptation so that we can avoid expensive adaptation during retrieval?

REFERENCES

Bergmann, R., & Wilke, W. (1996) PARIS: Flexible Plan Adaptation by Abstraction and Refinement. *Proceedings of the Workshop on Adaptation in Case-Based Reasoning, 12th European Conference on Artificial Intelligence*, Budapest, Hungary.

Bergmann, R., & Wilke, W. (1995) Building and Refining Abstract Planning Cases by Change of Representation Language. *Journal of Artificial Intelligence Research,* **3**, pp. 53 - 118.

Branting, L. K. & Aha. D. W. (1995) Stratified Case-Based Reasoning: Reusing Hierarchical Problem Solving Episodes. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Montreal, Canada.

Hammond, K. (1989) *Case-Based Planning*. Academic Press.

Kolodner, J. (1989) Judging Which Is the "Best" Case for a Case-Based Reasoner. *Proceedings of the Case-Based Reasoning Workshop*. Florida, USA.

Maher, M. L. & Zhang, D. M. (1993) CADSYN: A Case-Based Design Process Model. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing.,* ***7****(2), pp. 97-110.*

Netten, B.D., & Vingerhoeds, R. A. (1996) Adaptation for Conceptual Design in EADOCS. *Proceedings of the Workshop on Adaptation in Case-Based Reasoning, 12th European Conference on Artificial Intelligence,* Budapest, Hungary.

Purvis, L., & Pu, P. (1996) An Approach to Case Combination. *Proceedings of the Workshop on Adaptation in Case-Based Reasoning, 12th European Conference on Artificial Intelligence*, Budapest, Hungary.

Purvis, L. & Pu, P. (1995) Adapting Using Constraint Satisfaction Techniques. (Ed.s M. Veloso & A. Aamodt) *Case-Based Reasoning: Research & Development - Proceedings of the 1st International Conference on Case-Based Reasoning*. Springer-Verlag

Smyth, B. & Cunningham, P. (1992) A Hierarchical Case-Based Reasoning System for Software Design. Proceedings of the *10th European Conference on Artificial Intelligence*, Vienna, Austria.

Smyth, B. & Keane, M. T. (1995) Experiments on Adaptation-Guided Retrieval in a Case-Based Design System. (Ed.s M. Veloso & A. Aamodt) *Case-Based Reasoning: Research & Development - Proceedings of the 1st International Conference on Case-Based Reasoning*. Springer-Verlag

Smyth, B. & Keane, M. (1994) Retrieving Adaptable Cases. (Eds. M. Richter, S. Wess, and K-D Dieter) *Topics on Case-Based Reasoning. Lecture Notes on AI*. Springer-Verlag.