Knowledge Engineering in a Real World Case-Based Reasoning Application[1]

Pádraig Cunningham, Andrea Bonzano
Trinity College Dublin
padraig.cunningham@tcd.ie, bonzano@ms.com

Abstract

Case-Based Reasoning (CBR) has emerged from research in cognitive psychology as a model of human memory and remembering. It has been embraced by researchers of AI applications as a methodology that avoids some of the knowledge acquisition and reasoning problems that occur with other methods for developing knowledge-based systems. In this paper we propose that, in developing knowledge based systems, knowledge engineering addresses two tasks. There is a problem analysis task that produces the problem representation and there is the task of developing the inference mechanism. CBR has an impact on the second of these tasks but helps less with the first. We argue that in some domains this problem analysis process can be significant and propose an iterative methodology for addressing it. To evaluate this, we describe the application of case-based reasoning to the problem of aircraft conflict resolution in a system called ISAC. We describe the application of this iterative methodology and assess the knowledge engineering impact of CBR.

1 Introduction

Having access to relevant case history in problem solving reduces the need for problem analysis because solution chunks from old problems can be reused and less in-depth analysis of the new problem is required. The conventional wisdom is that case-based reasoning (CBR) is particularly appropriate in *weak theory* domains where the important influences and interactions are not well understood. The hope is that the problem of modelling these interactions can be finessed by reusing previously solved problems stored as cases. This suggests that developing CBR systems may require less knowledge engineering than, say, rule-based or model-based approaches. It is generally accepted among CBR researchers that this is only true to a limited extent. A CBR system that is not built on the type of domain analysis that knowledge engineering involves will probably not work very effectively (Cunningham, 1998).

The development of a knowledge-based system (KBS) involves: identifying a real world problem solving task that is to be tackled, representing the key components of this task in the KBS, and implementing the inference process that produces solutions. Thus there are two key components involved in the knowledge engineering process. There is the task of producing a representation of the problem that captures the key features and the task of developing an inference mechanism that describes the causal interactions involved in deriving solutions, as shown in Figure 1.

The inference mechanism is implemented using a case-base of solved problems and a mechanism for retrieving and adapting these cases. Many implemented CBR systems involve little or no adaptation and the reasoning mechanism is simply a retrieval system with solutions being used intact or with adaptation performed by the user.

[1] This research has been funded by Eurocontrol Experimental Centre, Paris, the European Centre for Air Traffic Control.

The knowledge is encoded in the system in:

- the knowledge representation used,
- the cases themselves,
- the similarity metric used in identifying cases to be reused,
- the mechanism for adapting solutions, if any.

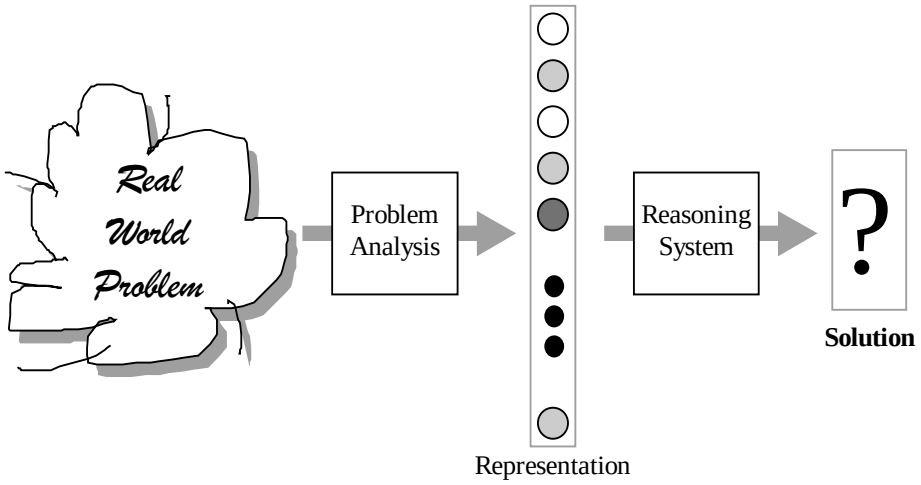

Figure 1: *Development of a KBS.*

In Richter's knowledge container terms these four components correspond to the vocabulary, case, retrieval and adaptation containers (Richter, 1995, Wilke & Bergmann, 1998). The development of the similarity metric and the adaptation mechanism is probably simpler than alternative techniques provided the adaptation mechanism does not prove too complicated (Cunningham, Finn and Slattery, 1994).

If retrieval and adaptation mechanisms are easy to implement, then CBR has clear knowledge engineering advantages over "from first principles" techniques. However, this advantage will be reduced if the problem analysis task that produces the problem representation should dominate in the knowledge engineering effort. So in CBR systems with no automatic adaptation the big knowledge engineering issue is the knowledge representation. If an appropriate representation is easy to identify then CBR has strong benefits; if not then there may be considerable knowledge engineering effort in determining a good representation.

In the later sections we analyse an iterative process of improving the representation driven by an analysis of the faulty solutions produced by a CBR system. First, we begin with a brief description of the air traffic control (ATC) problem addressed in the case study described here. Then, in section 3, we present the background for this iterative knowledge acquisition process. The paper concludes with analysis of effectiveness of this process and some reflections on when there is a big knowledge engineering *win* associated with CBR and when there is not.

2 ISAC

The acronym ISAC stands for Intelligent System for Aircraft Conflict Resolution. It is a CBR system that helps air traffic controllers to solve conflicts between sets of aircraft

(Bonzano, Cunningham & Meckiff, 1996; Bonzano, 1998). A conflict occurs in ATC when two (or more) aircraft are on course to pass too close to each other resulting in what is called a 'loss of separation'. Minimum horizontal separations are typically 8 nautical miles (1 Nm=1852 m) in radar controlled regions and either 1000ft or 2000ft vertically*, depending on altitude. The three stages of the decision making process for conflict resolution are:

- selecting the aircraft to manoeuvre,
- deciding on the type of manoeuvre and
- specifying the details of the manoeuvre.

The choices made depend on several factors: the geometry of the conflict, the capabilities of the aircraft, their position relative to their destination, etc. ISAC is an intelligent agent that assists the controllers in the first two stages of this decision process. ISAC is designed to interact with another system called HIPS (Highly Interactive Problem Solver) which detects conflicts and offers vertical, horizontal and speed perspectives on the conflicts in different windows (see Figure 2). ISAC acquires a representation of the next conflict for processing from HIPS (see Figure 3). It then selects an aircraft and a type of manoeuvre and the appropriate windows for visualising this manoeuvre are presented to the controller. If the controller does not like the proposed manoeuvre then the case-base is updated accordingly; i.e. a case incorporating the solution used by the controller is added to the case-base. Much of the development effort in ISAC involved refining the case representation to produce a set of features that characterised and discriminated cases in all circumstances.

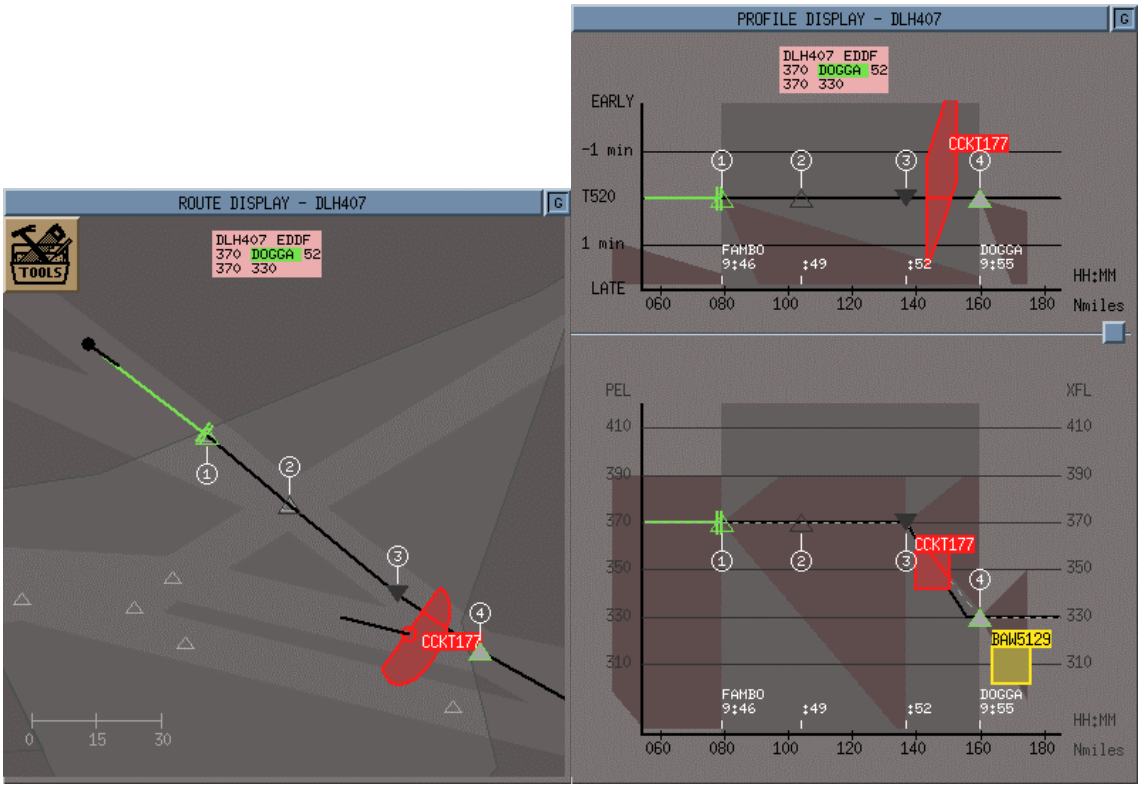

Figure 2: *The horizontal, vertical and speed views in HIPS*

* ATC in Europe and the US does in fact use imperial units.

Because of the nature of the conflict resolution problem ISAC is effectively a retrieval-only CBR system. Solution adaptation is performed only to the extent that in some circumstances solutions are aggregated to produce a single solution. Case retrieval is effectively a *k*-Nearest Neighbour (*k*-NN) process using feature weights that vary from case to case to capture the context sensitivity of the problem. (Bonzano, Cunningham & Smyth, 1998a&b).

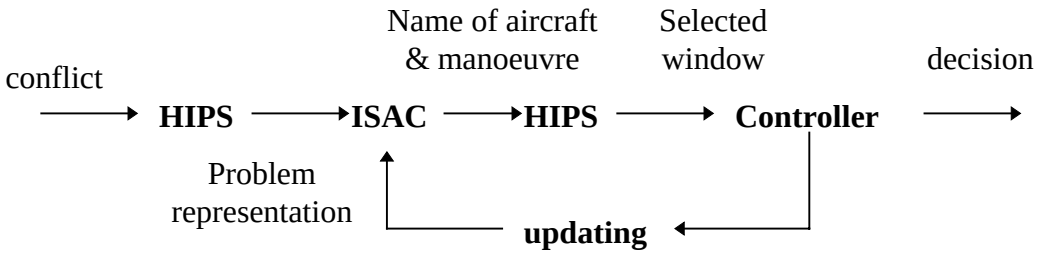

Figure 3: *The interaction of ISAC and HIPS.*

3 Knowledge Acquisition

The dominant methodology for developing knowledge based systems is CommonKADS (Schreiber et al. 1994). CommonKADS is inspired by Boehm's Spiral Model for software development (Boehm, 1988) and we will use that as our starting point here.

Bohem proposed his Spiral Model of software development in the late eighties in response to the inflexibility of the standard Waterfall Model. The Waterfall Model requires that complete design documents are prepared before proceeding to software development. When this is possible it is the right way to go, but often requirements are poorly understood and poorly specified and it is simply not possible to provide completely elaborated designs early in the process. What Boehm proposed is that there should be a few iterations of the development process with more sophisticated prototypes being produced at each stage. This is illustrated in the diagram of the Spiral Model shown in Figure 4. A typical cycle involves elaborating a portion of a product (e.g. performance, functionality, etc.). Each cycle has distinct; requirements analysis, design, develop and test phases.

When we move to the design and development of knowledge based systems this problem of poorly understood requirements is compounded by an incomplete understanding of the interactions in the problem domain. The CommonKADS process is cyclic in the same way as the Spiral Model (see Figure 5). However, given the particular requirements of KBS development, much of the emphasis is on refining the model at each iteration.

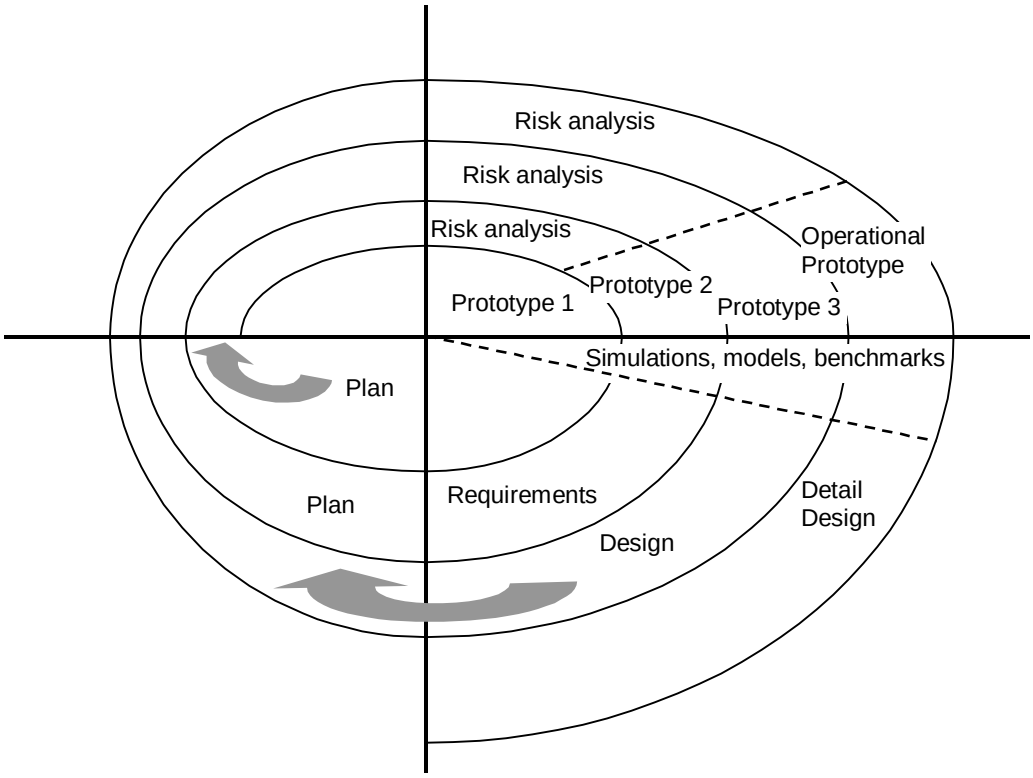

Figure 4. Boehm's Spiral Model of the Software Process (Boehm, 1988).

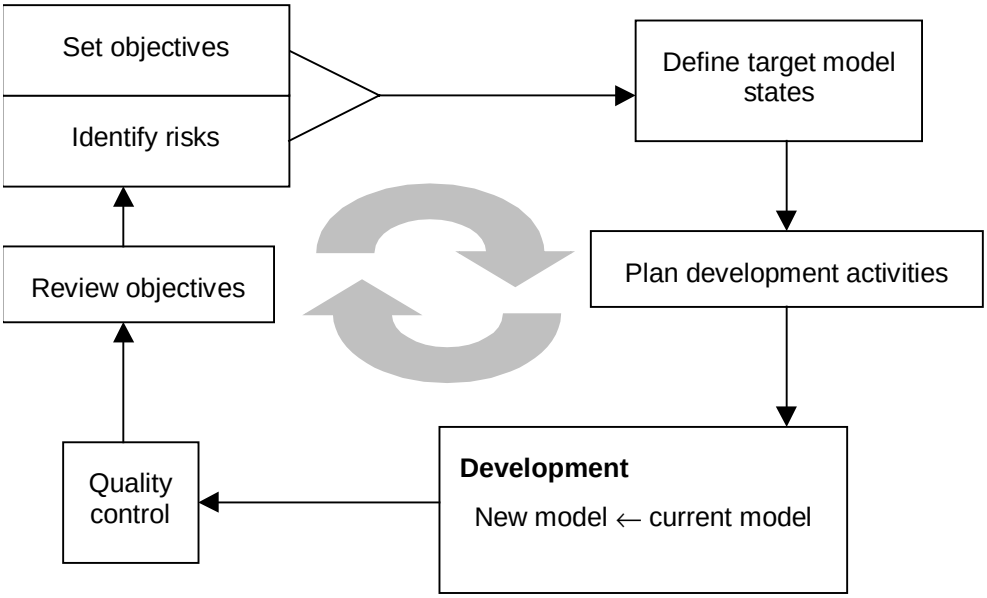

Figure 5. The cyclic development process in CommonKADS (Schreiber et al. 1994).

As stated in the Introduction, CBR may finesse some of this modelling requirement – particularly if there is to be no automated adaptation involved. However, there is still the need to come up with the appropriate case representation. In terms of our ATC application we don't know in detail what the appropriate features are and we don't know how they influence the decision making process. With CBR the answer to this second question is compiled into the cases and we do not need to worry about it. We are still left with the representation problem to solve however. In turn, there are two parts to this

task, there is the *abductive* process of proposing relevant features (**1**) and there is the analytic process of assessing the relevance of these features(**2**) (see Figure 6). In our methodology the process of proposing new features is driven by analysis by domain experts of errors produced by the different prototype systems. An error occurs when two cases appear similar in the current case representation but should not. The solution is to propose new features or revise existing features so that they can discriminate.

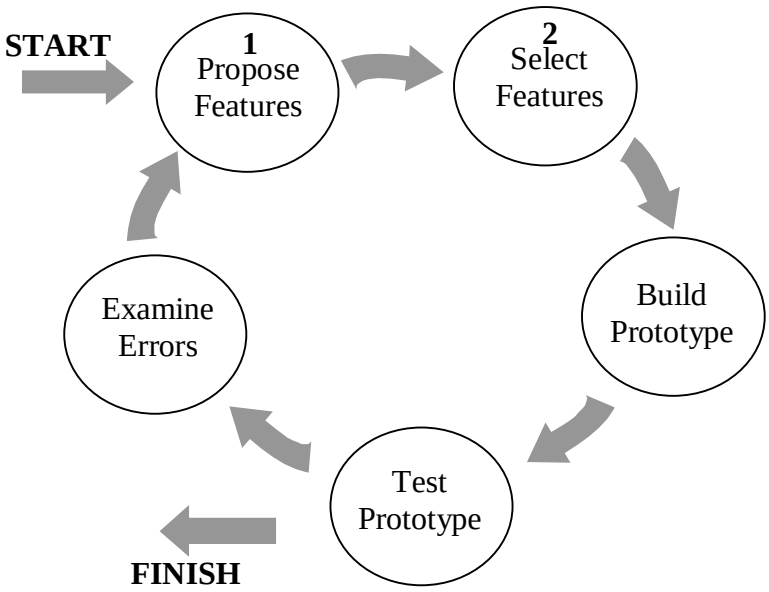

Figure 6. The cyclic process for determining a good case representation.

This process generates a set of possible features and these need to be assessed for their usefulness. Because ISAC uses *k*-NN retrieval it is important that less predictive features should be removed from the case representation. It is well known that irrelevant features cause problems with *k*-NN because, in effect, they are noise. In order to rate features for their predictivness the type of information theoretic analysis used in ID3 was adopted (Quinlan '86). This allows features to be scored according to their predictivness of particular outcomes (e.g aircraft to manoeuvre or type of manoeuvre). The features that scored well were selected for inclusion in the case representation. There are some problems with this approach (e.g. correlated features and numeric features) and these are discussed in the Conclusions.

4 Case Study

4.1 Getting Started

The project started with an initial two month phase spent getting familiar with the ATC domain. The available literature on ATC and on expert systems in ATC was investigated. Talking to controllers and taking part in real time simulations was also important in developing an understanding of the domain. The participation in ATC simulations and discussions with the controllers suggested some initial ideas on what the features for the description of the conflicts might be, the problems that could be encountered and the assumptions to be made.

4.2 Initial Prototype

During this phase, that took 10 months, the first version of HIPS in which ISAC was embedded was a very simple visualisation tool representing the radar screen and the flight plan strips used by the controllers. In this version a lot of significant features were not available. The first case-base was built by showing the controllers some conflicts coming from very simple traffic samples and by generalising from what was understood from the literature. This first case description is reported in Table 1 with the name of the features and their possible values. This first description was heavily influenced by two systems previously developed in Eurocontrol.

Table 1: *Initial case description.*

Feature type	Name of feature	Possible values
General features	horizontal-conflict-configuration	similar track, opposing track, crossing track
	medical-emergency	yes, no
Flight characteristics for each aircraft	size of the aircraft	light, small, medium, heavy
	absolute-speed	numeric value
	phase-of-flight-before conflict	climb, cruise, pre-descent, descent
	phase-of-flight-during conflict	climb, cruise, pre-descent, descent
	phase-of-flight-after conflict	climb, cruise, pre-descent, descent
	altitudes before, during and after the conflict	numeric value
	is the aircraft close to the sector boundaries?	yes, no
Aircraft capabilities	can the aircraft go faster?	yes, no
	can the aircraft go slower?	yes, no
	can the aircraft climb?	yes, no
	can the aircraft descend?	yes, no
	can the aircraft turn left?	yes, no
	can the aircraft turn right?	yes, no
Environment features	is an altitude manoeuvre possible?	above, below
	is an horizontal manoeuvre possible?	left, right
	is a speed manoeuvre possible?	acceleration, deceleration

In the case-base, each case has a name, is described using the above features and has a solution which consists of the aircraft that has to be manoeuvred and the type of manoeuvre. Two different case representations were adopted and tested: a representation with the description of both the aircraft in the same case, referred to as “TwoInOne” and a representation referred as “OneInOne” with the description of only one aircraft in each case. This rather odd OneInOne representation was evaluated because it was felt that it would facilitate the handling of multiple aircraft conflicts in the future.

Discussion with the controllers suggested that some features were critical to the matching process and these were declared to be constraints, which meant that cases needed to match exactly on these features to be retrieved. This had the effect of speeding

up retrieval without any loss in competence (Bonzano, Cunningham and Meckiff, 1996) but these constraints were to prove problematic in later versions of the system.

The main problem encountered while preparing this first conflict representation was the excessive simplicity, and sometimes superficiality, of many of the components: the conflicts were badly specified, the environment was too unrealistic and the acquisition of the features was not accurate.

Because the initial version of HIPS emphasised the geometrical aspect of the no-go zones (the zones where loss of separation would occur), the initial case description was mainly geometric. There was no reference to other important features like the aircraft performance, the departure and arrival airports and the type of aircraft. Reviewing the performance of this initial prototype with controllers showed that this other information needed to be taken into account.

The acquisition of the feature values from HIPS was also a considerable problem. Two conditions had to be satisfied. It had to be possible to extract a value for the feature from HIPS and the extracted value must correspond to what the controller sees on the screen.

The first issue involved the manipulation of a lot of data and the use of a lot of geometry, e.g. to find the angle to exit from a no-go zone. The second issue implied asking the controllers many questions to see if the acquired features were expressing exactly what he/she intended.

For the evaluation of this version of the retrieval mechanism and of the case representation, a case-base composed of a small set of carefully selected cases was built. Because the features used were so simple, it seemed possible to cover almost all the case space with hand crafted cases that were quite general. This system of 50 cases was tested using a LeaveOneOut strategy; i.e. each case was removed from the case-base in turn and the remaining case was tested to see if it produced the correct answer for that case. The results were quite good: the system found the same solution as the one given by the controller in 95% of the conflicts.

The most important and helpful feedback from this evaluation came from verbal comments made by controllers during the testing sessions. Moreover, this evaluation was useful for the verification of the speed, efficiency and robustness of the tool in the hands of controllers.

Summarising, the first step in the construction of the case-base was necessary to set all the structures up and to prepare the retrieval engine, but the resulting system was a toy system that did not have the robustness needed to work in the real world.

4.3 Interim Refinements Description

Having established the soundness of the basic idea the next step was to build a system with real cases and a more realistic case description. It was first decided to build a completely new case-base by using traffic samples coming from real time simulations. Secondly, an updated version of HIPS was introduced which included an actual radar screen for the visualisation of the sector. With these improvements the controllers felt more comfortable and the solutions given became more precise. The most important change was in the case representation where more complex features were used and the number of possible values for each feature extended. These changes were driven by an analysis with controllers of the shortcomings of the first prototype. The possibility of having more than one acceptable solution was also introduced.

A controller examined the entire set of new and more realistic conflicts and gave some "non-artificial" solutions. It seemed that two problems had been solved: the conflict description presented through HIPS was more realistic compared to the previous version and the solution had been given by an expert. Most of the features imply the existence of a "first" and a "second" aircraft. A set of rules decides which is the "first" aircraft and the canonical case description depends on those rules.

This phase took 3 months and the most evident change to the previous description was that there were less features dealing with the geometric description of the conflict and more features dealing with the performance of the aircraft. The features implying a direct route ("InFrontDirect" and "InFrontMoreSpace") had not been used because horizontal manoeuvres are not very common. Features like "Horizontal Intention" that might seem essential to a non-expert for understanding the geometry of the conflict were present in the initial description but were discarded afterwards. The controller is not interested in knowing whether the aircraft is turning right or left, but is only interested in knowing whether the aircraft is turning or not.

The overall complexity affected the performance of the system. The case-base, constituting 67 "realistic" conflicts extracted from the available traffic samples, was tested with the "LeaveOneOut" method. The results, presented in (Bonzano, Cunningham and Meckiff, 1996), were worse than the results of the toy system: only 70% of the solutions suggested by ISAC matched the solutions given by the controller. The main reason for this was the lack of coverage of the space of all the possible cases; the more realistic case representation greatly increased the size and complexity of the problem space.

All the tests were performed using the Horizontal Conflict Configuration (HorConflConf) as the only constraint. This caused particular problems with case coverage when particular geometries of conflict were encountered. For example, in the case-base there were not enough "head-on" conflicts, so the solutions of most of the cases where the constraint value was "head-on" were solved incorrectly.

4.4 Third System Description

After the toy system and the first attempt to work with a real world system, the need for a bigger case-base was evident. Because of the lack of time, it was not possible to continue acquiring conflicts from the traffic samples to build a bigger case-base. The 67 conflicts coming from the real world simulation that constituted the case-base in the previous step were kept as a test set. A new case-base was built from scratch by giving the controllers the description of a general conflict and asking for its solution, then changing the features one by one and recording how the solution would change. During this phase, that took 3 months, a new case representation was introduced, characterised by a drastic reduction in the features represented with numerical values. As described in section 3 these new features were proposed based on analysis of failures produced by the previous prototype. This process over-produced candidate features and actual features were chosen by ranking predictivness using information theoretic analysis as mentioned in section 3.

During this analysis, it was discovered that the controller's workload heavily influenced the solution of the conflict even if it was not directly connected to the conflict description. Depending on the workload, a controller could alter his behaviour. If the workload is low the controller has time to choose a complex solution that will be

economical for the aircraft (in time or fuel). On the other hand if the workload is high there is time only for a very simple but sometimes expensive solution that does not need any monitoring. One way of calculating the workload would be to count the number of aircraft that are visible on the radar screen and if more than a certain percentage of the aircraft are climbing or descending than the workload is considered high. This percentage threshold varies depending on the controller. The workload, as a feature, has not yet been used in ISAC, we mention it here because it highlights the problems of knowledge elicitation that remain even with CBR.

The last change introduced during this stage was the evaluation strategy. The "LeaveOneOUT" strategy was replaced by a more realistic test on the case-base of 150 cases using as a test set the 67 conflicts coming from the real traffic samples as mentioned above.

In testing the system each solution given by a controller was recorded and compared with the solution given by ISAC and with the solution given by the other controllers. The controller could either accept or discard the solution suggested by ISAC.

Four different situations were tested. The system performance is reported in Table 2. The conflicts solved by only one controller have been identified with "One" whereas the tests done on the set of conflicts that have been solved by more than one controller are indicated by "AtLeastOne". For the "One" situation, a suggestion was considered correct if the solution of the controller and the solution given by ISAC were the same. In the "AtLeastOne" situation, ISAC's solution was considered correct if at least one of the controllers gave the same solution.

Table 2: *Performance of the third ISAC's prototype.*

Case Representation	Controller	% of correct solutions
OneInOne	One	49%
TwoInOne	One	71%
OneInOne	AtLeastOne	83%
TwoInOne	AtLeastOne	94%

It can be seen that the performance of the system with the "OneInOne" case representation is in general worse than the performance with the "TwoInOne" case representation. This is different to what was observed in the previous evaluation but is confirmed in the final evaluation. This is supported by the intuitive consideration that the "OneInOne" case representation is less effective because less information about the global conflict and the other aircraft is stored in the case. For this reason, the "OneInOne" case representation was not used in the final evaluation of the system.

A more detailed analysis of the results revealed that the majority of the errors made by ISAC were due to a wrong choice of the aircraft to manoeuvre but not to the incorrect type of manoeuvre. This was encouraging because the case-base used for the evaluation contained very little knowledge about the choice of the aircraft.

From the performance, it was clear that a lot of work still had to be done on extending the case-base because 150 conflicts were not enough to characterise all the possible ATC conflicts. It was also clear that the case representation still did not capture what the controller sees on the radar screen.

4.5 Fourth System Description

The final version of the case representation is shown in Table 3 and an example TwoInOne case is shown in Table 4. Two key features in this representation are manoeuvrability and close-to-TOD. It was found in analysis of failures of the previous system that controllers were using information about the relative manoeuvrability of different aircraft types and this information is now made available to ISAC by a database lookup. The close-to-TOD feature reflects how close an aircraft is to the point when it starts to descend and captures the important 'phase of flight' information correctly.

Table 3: *Final case description.*

Name of feature	**Possible values**
horizontal-conflict-configuration	crossing, converging, head-on, diverging
priority	higher, lower, same
altitude-now	different, same
speed	faster, slower, same
altitude-configuration	climbing, descending, stable
close-to-TOD	numeric value
close-to-boundaries	numeric value
manoeuvrability	numeric value
easy-to-exit-horizontally	veryEasy, easy, possible, difficult
levels-available	yes, none, above, below, withSpaces
faster	easy, possible, difficult
slower	easy, possible, difficult

After realising that the most used manoeuvre is the altitude manoeuvre and that the horizontal and speed manoeuvres are not often used, there was no longer a need to discriminate between the two "easy-to-exit-right" and "easy-to-exit-left" features: the more general "easy-to-exit-horizontally" feature was introduced. For the same reason, the "altitude" solution given by the system was changed into a more precise "climbing" or "descending" solution.

Table 5: *A conflict expressed in the final "TwoInOne" case representation.*

Case690			
HorConflConf	crossing	Faster(A)	difficult
Priority	same	Slower(A)	difficult
AltitudeNow	same	AltConfiguration(B)	stable
Speed	faster	CloseToTOD(B)	352
AltConfiguration(A)	stable	CloseToBoundaries(B)	8.3
CloseToTOD(A)	155	Manoeuvrability(B)	0.78
CloseToBoundaries(A)	4.8	EasyToExitHoriz(B)	possible
Manoeuvrability(A)	0.78	LevelsAvailable(B)	yes
EasyToExitHoriz(A)	easy	Faster(B)	difficult
LevelsAvailable(A)	yes	Slower(B)	difficult
Solution	**dow1**		

Given the fact that solutions are not clearly right or wrong in this domain, a new policy for the evaluation of the system was introduced with the aim of gaining a better understanding of how the controller can interact with ISAC and how the system performs.

The tests consist of three steps: first, a conflict is shown to the controller. Second, the solution for the conflict is requested from the controller. In the final step ISAC gives, on purpose, either a good or a bad solution to the conflict. The controller has to rank the given solution from 0 (very bad) to 7 (very good). The wrong solution is a random solution chosen from the solutions that were not selected. Moreover, the controller is asked why it is a good/bad solution and what changes could be made to the solution to improve it.

The second step is necessary because controllers sometimes accept sub-optimal solutions, as the controllers themselves confirm. By previously asking the controller for his solution, the risk of the controller passively accepting the solution suggested by ISAC is avoided. Moreover, with this step, it is possible to evaluate whether or not the controller is biased toward particular solution types.

The case-base used for the final round of tests has approximately 700 conflicts, i.e., 1400 cases in the "TwoInOne.nonCanonical" case representation, which has the best performance of all three representations. This final case-base also contains some conflicts stored with the purpose of solving some multiple aircraft conflicts.

The results of the evaluation are shown in Figure 7. The mark that all the controllers gave to the wrong solutions suggested by ISAC was, on average, 2.03/7, i.e. 29%, whereas the mark given to the good solutions was 5.49/7, i.e. 78%. The mark given by the controller that generated the solutions for the cases in the case-base are respectively 2/7, i.e. 28%, for the bad solutions and 7/7, i.e. 100%, for the good solutions. This discrepancy in marks is due to the different preferences of each controller.

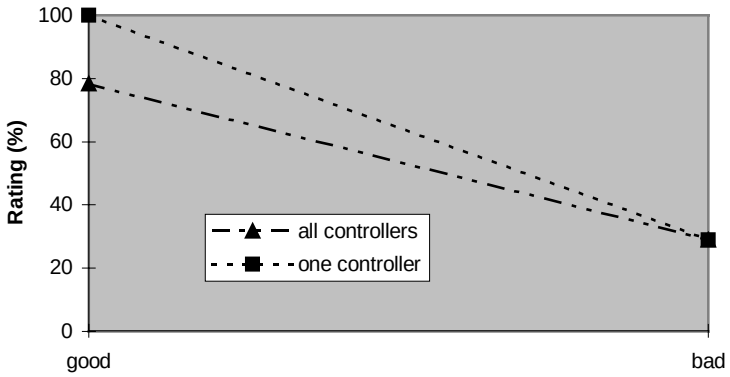

Figure 7: *How the controllers evaluate ISAC.*

This final result is very encouraging because it indicates that a controller will be completely satisfied with suggestions coming from a case-base made up of solutions originating with himself. This indicates that the final case representation captures the important aspects of the problem domain.

5 Conclusions

The first stage of the knowledge engineering process involved an analysis of the problem that produced a representation that can be manipulated by the reasoning system. The second stage involved developing the reasoning mechanism that manipulates the problem representation to produce a solution.

In developing IASC the second step was the easier to accomplish. The coding of the retrieval algorithm and adaptation, when present, was done without any major problems and the system is able to work with any case-base containing both numeric and symbolic features and no speed problems have been encountered. Instead it was found that the knowledge engineering effort in determining a good case representation was substantial and involved four iterations of the methodology described here.

The model of KE requirements described in Figure 1 can result in various specific scenarios. The characteristic of specific applications will dictate the balance of effort between the tasks of determining the salient features and developing the inference mechanism (Figure 8). CBR is very effective in a situation like in Figure 8 (a), where the acquisition of the case-base and the determination of the features is straightforward compared with the task of developing the reasoning mechanism. When the acquisition of the case-base and the decision of the features becomes more dominant, like in Figures 8 (b) and (c), the advantages of CBR over RBS are less evident. However with CBR the analysis of retrieval failures may help drive the refinement of the representation.

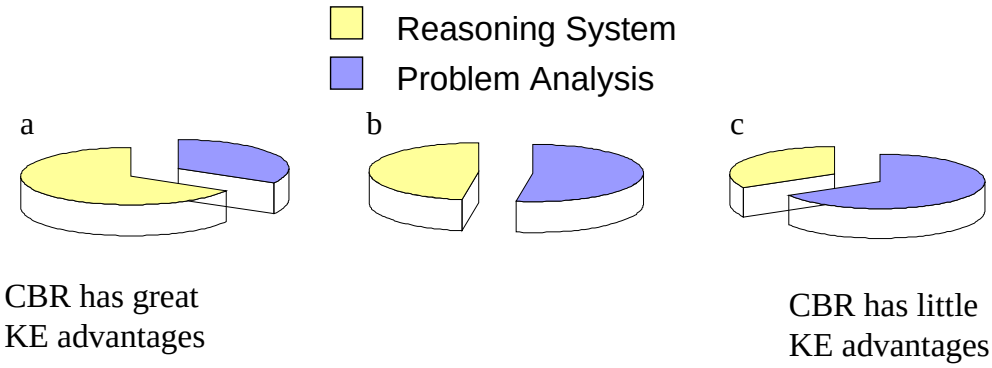

Figure 8: *Different divisions of effort in the knowledge engineeing process.*

If a case representation is not obvious we propose that the iterative methodology presented here is a good means of proceeding. This methodology can be summarised as follows:

At each iteration:

1. Analyse retrieval failures
2. Propose new features and new feature values to address retrieval failures
3. Select a discriminating set of features for the new case representation
4. Evaluate the competence of this new representation

The feature selection process involved using information theoretic analysis to score features by their predictivness of the outcomes – with the most predictive features being selected.

In the future we would like to improve on this feature selection process to address three perceived shortcomings. The first is that it is known that information theoretic analysis may not work well with numeric features. The second is that two highly correlated features may score well but one may be redundant in the presence of the other. Finally, all machine learning techniques have biases and in this scenario the bias of the feature selection process may be different to the bias of the learning system, i.e. feature selection should not be considered in isolation form the type of learning system used (Fukunaga, 1990)

6 References

Boehm, B. W., (1988) A Spiral Model of Software Development and Enhancement, *IEEE Computer*, Vol. 21, No. 5, pp61-72.

Bonzano A., Cunningham P., Meckiff C. (1996). ISAC: A CBR System for Decision support in Air Traffic Control, Advances in Case-Based Reasoning, Proceedings of the 1996 European Workshop on Case-Based Reasoning, I. Smith and B. Faltings Eds., Springer Verlag Lecture Notes in Artificial Intelligence, pp.44-57.

Bonzano A., Cunningham P., Smyth B. (1997a). Using introspective learning to improve retrieval in CBR: A case study in air traffic control, Case-Based Reasoning Research and Development, Proceedings of the 1997 International Conference on Case-Based Reasoning, D.B. Leake and E. Plaza Eds., Springer Verlag, Lecture Notes in Artificial Intelligence, pp.291-302.

Bonzano A., Cunningham P., Smyth B. (1997b). Learning feature weights for CBR: Global versus Local, AIIA 97: Advances in Artificial Intelligence, Proceedings of AIIA'97, M. Lenzerini Ed., Lecture Notes in Artificial Intelligence, Springer Verlag, pp.417-426.

Bonzano A. (1998). ISAC: a Case-Based Reasoning System for Aircraft Conflict Resolution, Ph.D. Thesis, Trinity College Dublin.

Cunningham, P., (1998) CBR: Strengths and Weaknesses, in Proceedings of *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, eds A. P. del Pobil, J. Mira & M. Ali, Lecture Notes in Artificial Intelligence 1416, Vo. 2, pp517-523.

Cunningham P., Finn D., Slattery S. (1994). Knowledge Engineering Requirements in Derivational Analogy, Topics in Case-based Reasoning, Lecture Notes in Artificial Intelligence, S. Wess, K.-D. Althoff, M.M. Richter Eds., Springer Verlag, pp.234-245.

Fukunaga, K., (1990) *Introduction to Statistical Pattern Recognition, 2nd Ed.,* Academic Press, San Diego.

Quinlan J.R. (1986) Induction of Decision Trees, *Machine Learning, 1*, 81-106.

Richter, M. M., (1995) The Knowledge Contained in Similarity Measures, *Invited Talk at the 1st International Conference on Case-Based Reasoning (ICCBR95),* (http://wwwagr.informatik.uni-kl.de/~lsa/CBR/Richtericcbr95remarks.html).

Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H., Van de Velde, W., (1994) CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert*, Vol. 9, No. 6, pp28-37, 1994.

Wilke, W., Bergmann R., (1998) Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving, in Proceedings of *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, eds A. P. del Pobil, J. Mira & M. Ali, Lecture Notes in Artificial Intelligence 1416, Vo. 2, pp497-506.