Case-Based Reasoning for Matching SmartHouse Technology to People's Needs

Nirmalie Wiratunga [a], Susan Craw [a,*], Bruce Taylor [b]

[a] *School of Computing, The Robert Gordon University, Aberdeen, AB25 1HG, Scotland, UK*

[b] *Scott Sutherland School, The Robert Gordon University, Aberdeen, AB10 7QB, Scotland, UK*

Abstract

SmartHouse technology offers devices that help the elderly and people with disabilities to live independently in their homes. This paper presents our experiences from a pilot project applying case-based reasoning techniques to match the needs of the elderly and those with disabilities to SmartHouse technology. The SmartHouse problem is decomposed into sub-tasks, and generalised concepts added for each sub-task. This decomposition and generalisation enables multiple case reuse employing a standard decision tree index based iterative retrieval strategy. Documented real situations are used to create a small case base. An initial prototype is evaluated empirically using leave-one-out testing, and separately with the domain expert on newly created test cases. Results show that system generated solutions to be comparable to those of a domain expert. Importantly, the iterative retrieval strategy employing multiple indices generated solutions that were significantly better compared to a one-shot retrieval without indices.

Key words: case-based reasoning, SmartHouse technology, multiple indexed retrieval, concept generalisation

* Corresponding author.
Email addresses: nw@comp.rgu.ac.uk (Nirmalie Wiratunga), smc@comp.rgu.ac.uk (Susan Craw), b.taylor@rgu.ac.uk (Bruce Taylor).

1 Introduction

Smart House Technology has been around for about twenty years and aims to encourage independent living for the elderly and disabled. This technology may consist of something as simple as a telephone amplifying unit for those with hearing impairments, to a field bus architecture system, where large numbers of sensors and actuators each with their own microprocessors are connected together, for people with severe mobility problems.

Occupational Therapists usually match assistive technology to people's needs by observing people engaged in daily tasks. If patients have difficulty doing these tasks then a technological solution may be suggested, often based on similar past patients. In recent years the range of technology has increased considerably. This additionally makes it harder for occupational therapists to keep abreast with SmartHouse devices, thereby creating a real need for automated tools to help match SmartHouse technology to people's needs (1).

A case-based reasoning (CBR) system solves new problems by reusing solutions from previously correctly solved similar problems (2). These past experiences consisting of problem descriptions and associated solutions are called cases and are stored in a case base. For the SmartHouse domain a case consists of a person's disability needs together with the SmartHouse devices that can enable this person to live independently. Case retrieval is the first stage of the CBR cycle in Figure 1. Here given a description of the new problem (a person's disability needs) a similar case or a subset of similar cases most useful for solving the new problem are retrieved from the case-base. Depending on the differences between the current problem and the retrieved cases some adaptation of the retrieved cases might be necessary before the retrieved solution can be reused. Subsequent stages include verification of the proposed solution and if necessary retention of the new problem and the modified solution with the aim of reusing it in the future.

Retrieval of relevant cases is crucial for the successful operation of any CBR system. This is particularly true when adaptation knowledge is difficult to acquire (3). This paper tackles two important issues for a retrieval only system in the context of the SmartHouse domain:

- Case representation enabling storage of cases in the case base according to a predefined structure; and
- An iterative retrieval strategy primarily comprising an indexing scheme to restrict the similarity matching to just a selection of cases, but also facilitating multiple case reuse.

Typically case representation has a flat structure consisting of feature-value pairs. Additionally the SmartHouse case representation captures the inher-

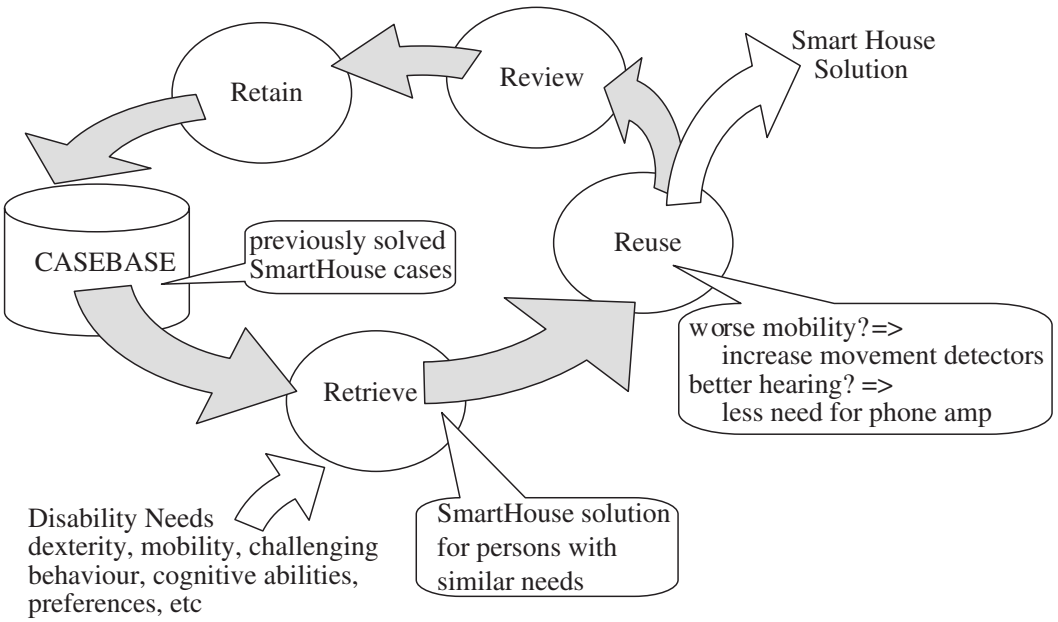

Fig. 1. SmartHouse task and the CBR cycle (2).

ent task decomposition within this domain. The similarity measure used for case retrieval can be a straightforward distance function, but it may also involve some knowledge intensive matching of cases (4). Retrieval knowledge typically comprises weights reflecting importance of features and relevance of features for index creation. Given the relatively small size of the SmartHouse case base, here we were able to acquire this knowledge from the domain expert.

In the rest of this paper we will describe how a prototype was built to match people's needs to SmartHouse technology. Section 2 discusses case representation issues, and an iterative retrieval strategy is presented in Section 3. Evaluation results are presented in Section 4 followed by related work in Section 5, and conclusions in Section 6.

2 Case Representation

The EU funded Custodian project created a SmartHouse technology simulation tool to illustrate how different SmartHouse devices function in the house (5). More importantly for the work presented in this paper eleven real scenarios consisting of people needs and matching technology solutions were acquired during this project.

The first task for developing our prototype CBR system was to *manually* translate the documented textual cases into a more structured representation of feature-value pairs. This involved identifying relevant features that belonged to the problem space and the solution space; and establishing the data labels

with which to instantiate these features.

Consider the SmartHouse textual case example in Figure 2, from this we can extract the following to describe the problem: wheel-chair-indoor Yes, wheel-chair-outdoor unknown, house-type ground-floor-flat, care-staff Yes, able-into-bath with-help; and the solution described by: electric-locks-external-door Yes, intercom-front-door video-hands-free. Converting the textual descriptions into problem and solution features require establishing quantifiable values with which to instantiate these nominal features. Some features are binary valued (e.g. Yes/No) but many are multi-valued (e.g. the feature able-into-bath can have values Yes, No or with-help). Missing values was another common problem because these cases were documented for an entirely different purpose from that of developing a CBR tool. Although our domain expert could have made informed guesses in some situations, often it was left as unknown.

... Ms M is an **indoor wheelchair** user with **cerebral palsy**. She is a tenant in her **own ground-floor flat**, and whilst living moderately independently, did have support from **care workers** to assist her in **getting dressed and bathed**... She required an **intercom** that was both **hands free** and with **video** so that she could operate it from her wheel chair ... also **electrically operated locks** were fitted on her **external door** ...

Fig. 2. Textual SmartHouse case.

A total of 108 features were identified of which 64 describe the problem in terms of the person, their home, abilities and needs; and a further 44 features describe the solution; i.e. SmartHouse devices. With increased numbers of features and values it made sense to group these features particularly the solution features, according to their context, forming 10 problem space groups and 14 solution space groups consisting of SmartHouse devices (see Figure 3). The number within brackets denotes the number of features in each group and with solution space groups these features directly correspond to number of devices, e.g. 5 `MovementRelated` devices. We will refer to solution space groups as device groups and for consistency use the same terminology for the 3 singleton groups (`RemoteAlarm`, `FBArchitecture` and `PoweredWindows`). Grouping devices is generally a straightforward task for the domain expert and in difficult instances a pragmatic decision was made, e.g. would it make a difference if devices associated with `Doors` is split into exterior and interior doors. Importantly this grouping decomposes the task thereby enabling retrieval to concentrate on each sub-task separately.

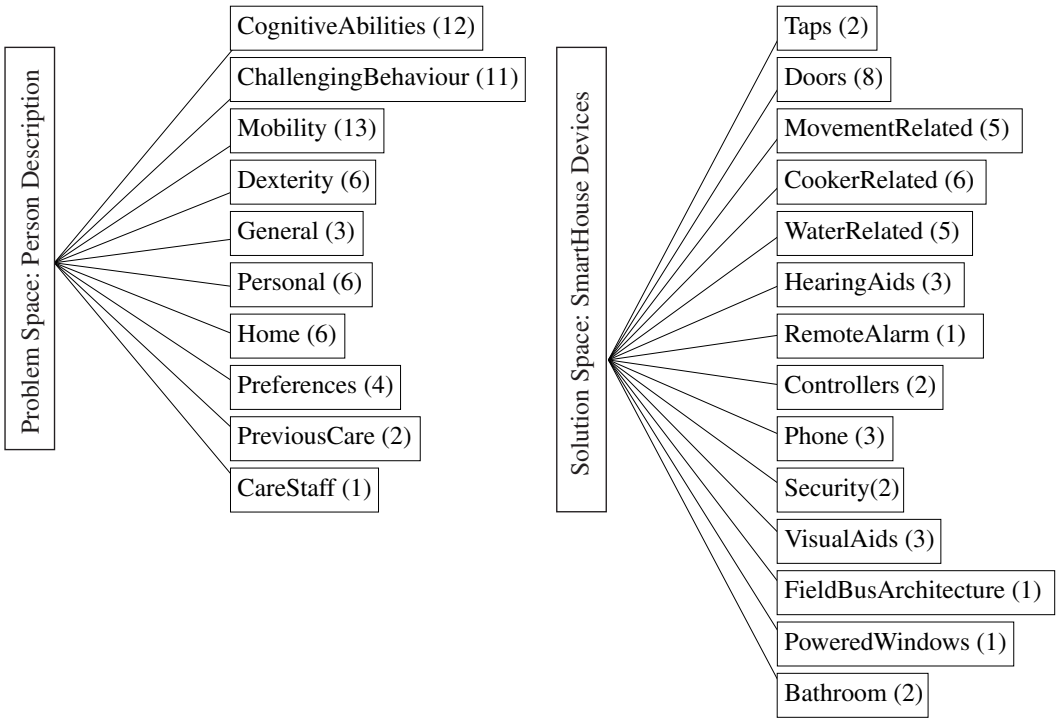

Fig. 3. SmartHouse case representation.

3 Retrieval Strategy

Several commercial CBR tools (e.g. ReCall, ReMind and Kate) use decision trees (DTs) to index the case base in order to improve retrieval efficiency (6). Although efficiency is a primary reason for using an index, a DT index formed using C4.5's information gain ratio heuristic (7) additionally provides a useful means to explain the underlying reasoning behind the retrieval; a desirable feature for CBR applied to the SmartHouse domain. Essentially the index is created for a selected concept (or class) and acts as a filter enabling the CBR system to locate the relevant subset of cases for a given problem (see Figure 4). The most similar k cases to the current problem are then identified from this subset by applying the k Nearest Neighbour algorithm (k-NN).

Let us consider a retrieval sub-task within the SmartHouse domain. Figure 5 illustrates a case base index created by inducing a C4.5 DT for the sub-task `PoweredWindows` (i.e. powered windows required or not). Given a new problem the tree is traversed and depending on the person's ability to open internal doors the relevant leaf node and its cases are retrieved. These cases form the neighbourhood and the majority solution obtained using k-NN instantiates the `PoweredWindows` part of the solution for the new problem. Notice that

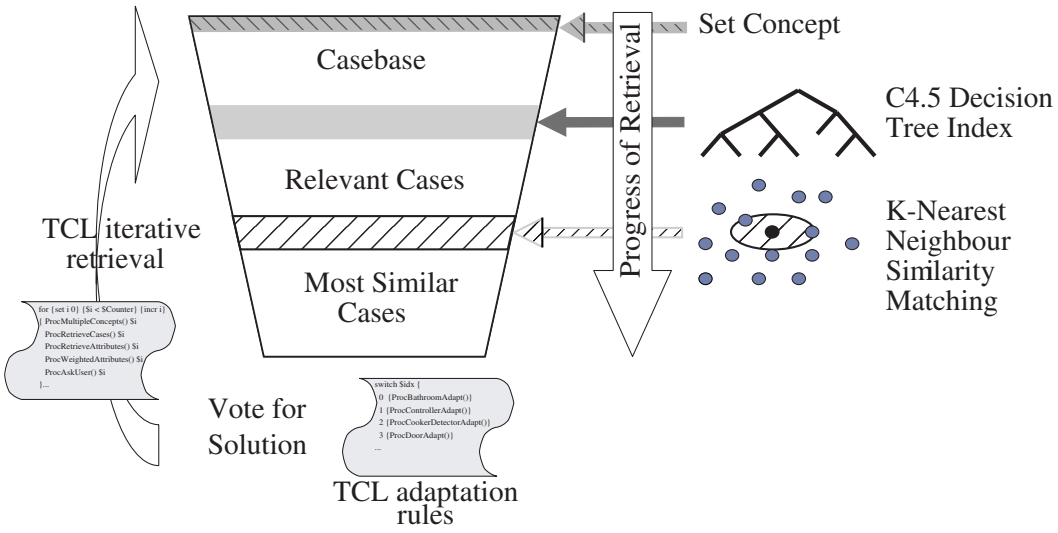

Fig. 4. Multiple case retrieval.

the DT is built with the aim of partitioning cases according to a specified concept; here it is `PoweredWindows` with possible instantiations values: Yes or No. Generally the partitioning tends to result in leaf-nodes containing cases with similar instantiations for `PoweredWindows`. However, with the second leaf node in Figure 5 we have 2 cases that relate to persons who are unable to open doors, yet one has powered windows, while the other does not, because existing sash windows cannot be powered. Therefore if cases were retrieved from this leaf node and the new problem also consisted of a house with sash windows then applying k-NN would result in the `PoweredWindows` part of the solution to be instantiated with value, No.

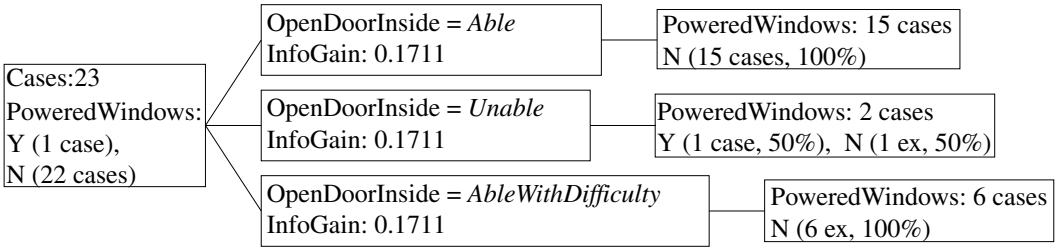

Fig. 5. Index aided retrieval for the PoweredWindows sub-task.

3.1 Using Multiple Indices for Iterative Retrieval

In previous work (8), we have shown how multiple DTs can be employed to solve design problems, where each tree concentrates on retrieving cases pertinent to solve a different part of a design task. Essentially this involves an iterative retrieval strategy whereby at a given iteration a different sub-task becomes the focus for case retrieval. The main advantage of this approach is that it enables the CBR system to use different parts of different cases in the

case base to solve different sub-tasks for the new problem.

A similar approach can be adopted for the SmartHouse domain, where identifying a suitable SmartHouse device is a sub-task and is solved in conjunction with a separate DT index given the new person's description. Traversal of each tree identifies a different subset of cases to solve each sub-task. This is the approach we use for `RemoteAlarms`, `PoweredWindows` and `FBArchitecture` requiring 3 DT indices. However if we were to insist on separate indices for the remaining devices we would need a further 41 DTs, which from an efficiency point of view does not make sense. Instead we exploit the fact that these 41 devices are already grouped in 11 device groups. We create a generalised concept feature, for each group. This means we induce 11 DTs (instead of 41), in addition to the 3 DTs for `RemoteAlarm`, `PoweredWindows` and `FBArchitecture`.

3.2 Creating Generalised Concepts

The new generalised concept feature that we wish to create for a given group should be an abstraction or summarisation of the devices that are members of that device group. We will now look at how a value can be assigned to such a feature created for a device group with the aid of an example. Consider two cases from the case base and their solution instantiations for just the `CookerRelated` device group (see Table 1). With *CaseX* we have a solution involving a timer-based automatic cooker shut-off, together with a detector that can detect gas, heat or smoke, and three alarms; while *CaseY* does not require detectors or alarms. So how do we establish a value for the new generalised concept feature, F_{cooker}? The simplest approach is to instantiate it with value Needed, if at least one of the cooker devices are needed, otherwise with value Not-Needed. Accordingly, in our example, F_{cooker}, is instantiated with Needed for *CaseX* and Not-Needed for *CaseY*.

In this manner all cases in the case base are modified by adding a generalised concept feature, per device group [1]. We can then induce a DT for each device group with the new feature selected as the concept. For example, to establish suitable devices related with cookers for a given new problem, the `CookerRelated` DT (induced with F_{cooker} as the concept) is traversed to identify the leaf node cases, and then k-NN is applied to access the k nearest neighbours. The majority solutions for cooker detectors and alarms from these k cases are reused and instantiate the cooker related solution part of the new problem.

[1] Note this generalisation is not required for the 3 singleton device groups.

Table 1
Case values for the CookerRelated device group.

CookerRelated	*Instantiations For*	
Group	*CaseX*	*CaseY*
auto-cooker-shut-off	{gas, heat, timer}	No
detectors-over-cooker	{gas, heat, smoke}	unknown
alarm-for-carers	No	No
cooker-isolation-alarm	No	No
fire-alarm	Yes	No
gas-alarm	Yes	No
F_{cooker}	*Needed*	*Not-Needed*

4 Evaluation

In our experiments we implemented the SmartHouse prototype using the ReCall CBR shell together with Tcl script to drive the retrieval. Multiple is the iterative retrieval strategy described in Section 3, where multiple indices enable reuse of multiple cases to solve different sub-tasks. Traversal of each index identifies the relevant subset of cases to which 1-NN is applied, and the solution part from the nearest neighbour corresponding to the sub-task is reused. Multiple is compared with Single, where the entire solution of the best match case is reused. Unlike Multiple, Single does not use indices, instead applies 1-NN to the entire case base, once.

Eleven real-cases were supplemented with another twelve cases invented by our domain expert, forming a case base of 23 cases. Evaluation was carried out in two ways:

Leave-One-Out Testing: 10 cases from the real cases were each used in turn as the probe to the CBR system containing a case base of the remaining 22 cases. This provides results from 10 different leave-one-out test runs; and
User Testing: Comparison of the system's and expert's solutions for 3 new problem scenarios supplied by the expert.

Ascertaining the quality of a generated system solution involves establishing its similarity to the expert's solution. When the solution contains nominal values it is typical to employ the standard overlap similarity metric for this purpose (9). However this metric can be misleading when the distribution of nominal values is highly skewed. Although there are 44 devices, a SmartHouse solution would very rarely require an instantiation consisting of all devices. Instead it is more likely for a solution to require fewer than half of

these devices.

Consider the example in Table 2 listing the system's and expert's solution for the `Doors` device group. Of the 8 listed devices we have an exact match in 5, of which 4 are True Negatives (TN) and only one is a True Positive (TP). A False Negative (FN) is a system fault caused when it fails to propose a device, as with electric-locks-external-doors. In contrast with contacts-on-external-doors we have the system incorrectly proposing a device, referred to as a False Positive (FP). A similar situation arises with intercom-to-front-door, but here the difference is that, although the system has proposed a device, it is not the right one (Audio instead of Video-HF). We will consider such a system failure also as a FP although it is actually referred to as a failing positive (10).

Table 2
Comparing system and expert solutions.

Doors Group	*Solution*		*Categories*			
	System	*Expert*	*TP*	*TN*	*FP*	*FN*
alarm-when-open	No	No	0	1	0	0
contacts-on-external-doors	Yes	No	0	0	1	0
contacts-on-bedroom-door	No	No	0	1	0	0
contacts-on-bathroom-door	No	No	0	1	0	0
contacts-on-kitchen-door	No	No	0	1	0	0
intercom-to-front-door	Audio	Video-HF	0	0	1	0
powered-external-doors	Yes	Yes	1	0	0	0
electric-locks-external-doors	No	Yes	0	0	0	1

The overlap similarity metric (Sim), is the number of matching devices divided by all the devices; $(TP + TN)/(TP + TN + FP + FN)$. In the Table 2 example, this overlap is 0.625. Here the influence of the lesser important TNs can suggest an overly optimistic similarity between solutions. Therefore, we also employ traditional Information Retrieval measures (11) in the context of SmartHouse devices:

- Precision (P) is the proportion of proposed devices that are actually correct $\mathrm{P} = TP/(TP + FP)$; and
- Recall (R) is the proportion of required devices that are correctly proposed $\mathrm{R} = TP/(TP + FN)$.

Generally high recall is desirable because lower FN and higher TP is important for the SmartHouse domain. The combined effect of these two measures is commonly measured by the F-measure (F) calculated as their harmonic mean; $2PR/(P+R)$. By applying this to the `Doors` example in Table 2 we have a far

more realistic similarity based comparison of system and expert solutions; P is 0.33, R is 0.5 and F is 0.4. Significance is reported from a one-sided paired Wilcoxon signed rank test.

4.1 Leave-One-Out Testing

The average results from 10 leave-one-out test runs appear in Table 3. The first column lists the 14 device groups and the adjacent column indicates the number of devices in each group. For comparison purposes both the Sim and F values are provided for SINGLE and MULTIPLE.

Table 3
Averaged results from 10 leave-one-out test runs.

Device group	*No Of Devices*	*Single*		*Multiple*	
		Sim	*F*	*Sim*	*F*
Bathroom	2	0.85	0	0.85	0
Controllers	2	1	1	0.9	0.67
Cooker Related	6	0.83	0	0.83	0
Doors	8	0.85	0.57	0.89	**0.71**
Hearing Aids	3	0.93	0.8	1	**1**
Movement Related	5	0.86	0.46	0.86	0.46
Phone Related	3	0.73	0.42	0.8	**0.7**
Remote Alarms	1	0.9	0.8	0.9	0.67
Security	2	0.85	0	1	**1**
Taps	2	0.9	0	1	**1**
FB Architecture	1	0.9	0	1	**1**
Visual Aids	3	0.9	0	0.9	0
Water Related	5	0.98	0	1	**1**
Windows	1	0.6	0.33	0.8	**0.67**
Overall Average		*0.86*	*0.31*	*0.91*	*0.63*

MULTIPLE shows a significant improvement in 8 of the 14 groups over SINGLE; with both F (p=0.016) and Sim (p=0.029) measures. The domain expert indicated that some interesting valid alternative solutions were proposed by MULTIPLE. For instance in connection with `WaterRelated` for a person with cognitive impairments, learning problems and challenging behaviour, MULTIPLE suggested the need for flood detectors in the bathroom while the expert

solution contained special push to operate taps. With another problem, MULTIPLE suggested the use of a pendant activated community alarm plinth as one of the 3 devices grouped under `Phone`. Although the expert solution also had the community alarm plinth the need for a pendant was documented as a future need.

With `Controllers`, MULTIPLE incorrectly suggests the use of environment and lighting controllers for a person that only has some mobility problems requiring the use of a wheelchair indoors, but otherwise having good manual dexterity and therefore able to operate switches. However MULTIPLE's suggestion of an environmental controller is not a totally wrong suggestion. Yet if this overly expensive solution had been implemented it would have gradually reduced the person's range of movements which would be both physically and psychologically detrimental to the person. This highlights the fact that system generated solutions should not be taken on board as the all encompassing definitive solution, and instead should be treated as a guide that localises search to potential solution areas. The over estimation of similarity with the overlap measure (Sim) is clearly seen with 7 of the device groups; SINGLE has a Sim measure > 0.85 (because there are many TNs) but the F measure is 0 (because TP is 0).

4.2 User Testing

Three new problem scenarios authored by the expert were used as probes to the CBR system. Test cases A and B were created by making minor variations based on existing cases while test case C was created from scratch, hence a more challenging problem. Importantly these test cases were generated to portray realistic problem scenarios with different levels of similarity to existing cases in the case base, and also with different levels of problem difficulty.

A comparison of the expert's and MULTIPLE's solutions for test cases A and B, appear in table 4. It is clear that these 2 test cases have a relatively low level of difficulty; they each require only 3 and 5 devices in the solution. Therefore it is not surprising that MULTIPLE's solution can be obtained by SINGLE's one-shot retrieval. We further investigated solution quality by increasing the nearest neighbours of MULTIPLE from 1 to 3. However due to the small case base size this increase actually had a detrimental affect on the solution.

Generally the domain expert felt that there was sufficient overlap between the actual and system solutions for test cases A and B. With `Controllers` for case A the expert solution suggests a lighting controller while the system solution fails to suggest this device. The reason for this is that the DT for `Controllers` selected, ambulant mobility, as a discriminatory feature to partition the case

Table 4
Comparison of MULTIPLE and expert solutions.

Device	*Test Case A*				*Test Case B*			
group	*TP*	*TN*	*FP*	*FN*	*TP*	*TN*	*FP*	*FN*
Bathroom	0	2	0	0	0	2	0	0
Controllers	0	1	0	1	0	2	0	0
Cooker Related	0	6	0	0	0	6	0	0
Doors	0	8	0	0	**1**	5	2	0
Hearing Aids	0	3	0	0	0	3	0	0
Movement Related	0	5	0	0	0	5	0	0
Phone Related	**1**	2	0	0	**1**	2	0	0
Remote Alarms	0	1	0	0	0	1	0	0
Security	**1**	1	0	0	0	2	0	0
Taps	0	2	0	0	0	2	0	0
FB Architecture	0	1	0	0	0	1	0	0
Visual Aids	0	3	0	0	0	3	0	0
Water Related	0	5	0	0	**3**	2	0	0
Windows	0	1	0	0	0	1	0	0
Total	**2**	41	0	1	**5**	37	2	0

base. Since the only case with a lighting controller device in the case base differed from test case A, in ambulant mobility, it did not fall under the same leaf node as test case A. This is an obvious problem with the limited number of cases in the case base. Similar explanations hold for test case B with for instance the `Doors` device group.

The use of DTs was well received by the expert because the decision nodes provides a useful explanation mechanism of the system's retrieval stage. For instance the selection of ambulant mobility, as the decision node for the `Controllers` device group, enabled the expert to conclude that although test case A involved a person with good ambulant mobility, a lighting controller device was necessary because of the inability to operate light switches.

Test case C was a harder case to solve because the expert's solution consisted of 11 devices compared to 3 and 5 devices for A and B. Therefore it was encouraging to see that the iterative retrieval strategy reflects this problem solving difficulty in that it reused 9 different cases to solve different sub-tasks of C. In Table 5, although MULTIPLE and SINGLE have equal Sim values, the Recall measure drops from 0.64 to 0.18 with SINGLE. Since the quality of the

proposed solution is best ascertained by the system's ability to predict (true) devices here the increase in recall with MULTIPLE is most encouraging.

Table 5
Results for test case C with SINGLE and MULTIPLE.

Device	SINGLE				MULTIPLE			
group	*TP*	*TN*	*FP*	*FN*	*TP*	*TN*	em FP	*FN*
Bathroom	0	2	0	0	0	2	0	0
Controllers	0	2	0	0	0	2	0	0
Cooker Related	0	4	0	2	0	4	0	2
Doors	0	5	0	3	**1**	5	0	2
Hearing Aids	0	2	0	1	**1**	0	2	0
Movement Related	0	3	0	2	**2**	0	3	0
Phone Related	**1**	1	1	0	**1**	1	1	0
Remote Alarms	**1**	0	0	0	**1**	0	0	0
Security	0	1	0	1	**1**	1	0	0
Taps	0	2	0	0	0	2	0	0
FB Architecture	0	1	0	0	0	1	0	0
Visual Aids	0	3	0	0	0	3	0	0
Water Related	0	5	0	0	0	5	0	0
Windows	0	1	0	0	0	1	0	0
Total	*2*	32	1	*9*	*7*	27	6	4
Recall	*0.18*				*0.64*			
Precision	*0.67*				*0.54*			
Sim	*0.77*				*0.77*			

Overall the poor performance with device groups containing a greater number of devices; such as `Doors`, `CookerRelated` and `MovementRelated`, may suggest that the binary valued approach adopted with generalised concept creation (F_i), needs to be extended to a multi-valued concept.

5 Related Work

Task decomposition is commonly employed for complex problem solving, and is particularly successful when sub-tasks have few interactions (12). One of the main advantages of task decomposition is multiple case reuse, and a good

demonstration of this is seen with a hierarchical CBR system developed for automating plant-control software design (13). The design case base is organised as a hierarchy and interestingly uses abstract and concrete cases; top level nodes are abstract cases decomposing the problem at subsequent levels in the hierarchy, while leaf nodes contain concrete sub-solutions. Since concrete cases can be attached to more than one parent the same case can be retrieved for different sub-tasks, thereby facilitating multiple case reuse. With the SmartHouse prototype, task decomposition was achieved by tackling device groups separately. Retrieval employed an automatically created DT structure as the index for each of these groups. Although abstract cases can be viewed as an index structure in a hierarchical case base, the knowledge engineering effort involved with identifying abstract cases and manually discovering connections between these would have been far more demanding for our domain expert. Instead DT based indexing greatly relieves this burden by automatically inducing a tree like structure which can then be examined by the expert.

The use of DTs as indices for sub-task retrieval has been applied with the tablet formulation domain (8), where instantiation of each component of a tablet is treated as a sub-task and a DT-based retrieval is triggered for each sub-task. The main difference with the SmartHouse domain is that the sub-tasks are too numerous to be dealt with separately and hence a grouping of sub-tasks into device groups and the addition of generalised concepts is required before iterative retrieval is manageable.

Iterative retrieval applied to decomposable tasks stands to gain much from a well designed case structure. In fact it is the discovery of device groups that enabled multiple case reuse with the SmartHouse prototype. The Auguste Project, operating in a similar application domain, aids decision making with regards to prescribing a neuroleptic drug for Alzheimer's patients (14). The case representation consists of a grouping of features where the top level view contains personal information, physical health, behavioural aspects, etc. We have also adopted this general approach of grouping areas of the problem and solution space, although with SmartHouse devices the solution space being more complex benefited from concept generalisation. An example of a hierarchical case representation can be seen with the NIRMANI system (15). This hierarchic structure is employed as a means to enforce context guided retrieval, where parts of cases are retrieved in stages. Presently the SmartHouse case representation involves a single level hierarchy however it is possible that a multi-level hierarchy might be appropriate particularly to cope with the increasing range of devices in this domain.

6 Conclusion

We have presented our experiences with developing a prototype CBR system for the SmartHouse problem domain, where the task involved matching SmartHouse technology to the needs of the elderly and people with disabilities. The main users of a SmartHouse CBR system will be suppliers of assistive technology, enabling them to match appropriate devices from supplier catalogues. The number of companies supplying SmartHouse devices to support independent living at home is growing as they recognise the market opportunities in the increasing aging populations of developed countries.

Organising the solution space into device groups decomposed the SmartHouse task into manageable sub-tasks. Task decomposition with concept generalisation enabled the use of a standard DT-based iterative retrieval strategy. Importantly the use of multiple indices enabled re-use of different cases to solve different parts of a given test case, thereby encouraging best use of the relatively small case base. An empirical evaluation has clearly shown significant improvement in solution quality with Multiple's retrieval strategy over Single's best match retrieval without indexing. Subjective evaluation with the expert has been promising, but we are conscious of the fact that these results are partly based on cases invented by the domain expert. Therefore a more realistic evaluation is needed before operational feasibility of this approach can be fully justified.

The ability to explain the reasoning behind the proposed system's solution is a desirable facility and with the SmartHouse prototype the domain expert found that the display of both index trees and similarity of cases at leaf nodes to be a useful step in this direction. Finally it is worth mentioning that it was the prototype's ability to use multiple cases to solve different parts of a given SmartHouse problem that most impressed our domain expert.

References

[1] J. Kolodner, Improving human decision making through case-based decision aiding, AI Magazine 12 (2) (1991) 52–68.

[2] J. Kolodner, Case-Based Reasoning, Morgan Kauffman Publishers, Inc., San Mateo, CA., 1993.

[3] K. Hanney, M. T. Keane, The adaptation knowledge bottleneck: How to ease it by learning from cases, in: D. B. Leake, E. Plaza (Eds.), Proceedings of the 2nd International Conference on CBR, ICCBR-97, Springer-Verlag, Providence, RI, USA, 1997, pp. 359–370.

[4] J. Jarmulak, S. Craw, R. Rowe, Genetic algorithms to optimise CBR retrieval, in: E. Blanzieri, L. Portinale (Eds.), Proceedings of the 5th Eu-

ropean Workshop on CBR, EWCBR-2K, Springer-Verlag, Berlin, Trento, Italy, 2000, pp. 137–149.

[5] G. Dewsbury, S. Bonner, B. Taylor, M. Edge, Final evaluation of tools developed, Tech. Rep. CUSTODIAN/RGU/WP7.3/re/004vb, CUSTODIAN Project, EU Telematics Initiative for Disabled and Elderly People (2001).

[6] D. W. Aha, L. A. Breslow, Comparing simplification procedures for decision trees on an economics classification task, Technical Report AIC-98-009, Navy Center for Applied Research in AI, Washington DC (1998).

[7] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, 1993.

[8] S. Craw, N. Wiratunga, R. Rowe, Case-based design for tablet formulation, in: B. Smyth, P. Cunningham (Eds.), Proceedings of the 4th European Workshop on Case-Based Reasoning, Springer, Dublin, Eire, 1998, pp. 358–369.

[9] T. Payne, P. Edwards, Implicit feature selection with the value difference metric, in: Proceedings of the European Conference on Artificial Intelligence, ECAI-98, John Wiley and Sons Ltd., Brighton, UK, 1998, pp. 450–454.

[10] D. Ourston, R. Mooney, Theory refinement combining analytical and empirical methods, Artificial Intelligence 66 (1994) 273–309.

[11] D. McSherry, Precision and recall in interactive case-based reasoning, in: D. W. Aha, I. Watson (Eds.), Proceedings of the 4th International Conference on CBR, ICCBR-01, Springer Verlag, Vancouver, BC, Canada, 2001, pp. 392–406.

[12] A. Stahl, R. Bergmann, Applying recursive CBR for the customisation of structured products in an electronic shop, in: E. Blanzieri, L. Portinale (Eds.), Proceedings of the 5th European Workshop on CBR, EWCBR-2K, 2000, pp. 297–308.

[13] B. Smyth, P. Cunningham, A hierarchical case-based reasoning - integrating case-based and decompositional problem-solving techniques for plant-control software design, IEEE Transactions on Knowledge and Data Engineering 13 (2001) 793–812.

[14] C. Marling, P. Whitehouse, Case-based reasoning in the care of Alzheimer's disease patients, in: D. W. Aha, I. Watson (Eds.), Proceedings of the 4th International Conference on CBR, ICCBR-01, Springer Verlag, Vancouver, BC, Canada, 2001, pp. 702–715.

[15] I. Watson, S. Perera, A hierarchical case representation using context guided retrieval, Knowledge-Based Systems 11 (1998) 285–292.