ソフトウェア工学 121－9
(1998. 11. 5)

CBRによる創造的なソフトウエア設計

サニプール シャダン　B.H. ファー　河野 善彌

〒338-8570 埼玉県浦和市下大久保255
埼玉大学情報システム工学科
Tel. 048-858-3489,　Fax. 048-858-3716
E-mail. shadan@cit.ics.saitama-u.ac.jp
http://www.cit.ics.saitama-u.ac.jp/~shadan/

あらまし:　ソフトウエア設計特にユーザーインターフェス設計は構造化が難しい作業である。この領域では設計対象が広範囲にわたるために、一定の戦略がいつも成功するとは限らない。創造的設計が必要に応じてなされるためである。小型知識ベースを備えた創造的なシステムによってソフトウエア設計における様々な問題に対応することが可能になる。本論文においては、創造的設計のためのモデルの実装方法を紹介し、それがユーザインターフェス設計に対し有効なアプローチであることを示す。ここでは創造的設計として蓄えられた知識から直接導くことのできない設計を想定している。そのためシステムは解の発見を可能にするような隠れた知識を表現する潜在的関係を考慮している。

キーワード　*CBR*、設計、創造的設計、動的な類似、ユーザインターフェス (*UI*)

Creative Software Design with CBR

Shadan Saniepour　Behrouz H. Far　Zenya Koono

Department of Information and Computer Science,
Faculty of Engineering, Saitama University
255 Shimo-okubo, Urawa 338-8570, Saitama, Japan
Tel. +81-48-858-3489,　Fax. +81-48-858-3716
E-mail. shadan@cit.ics.saitama-u.ac.jp
http://www.cit.ics.saitama-u.ac.jp/~shadan/

Abstract:　Software design or more particularly UI design is naturally an ill-structured task. The wide scope of design in this domain makes the routine strategies not always successful. The concept of creativity is among the preliminary necessities that may arise. Having a creative system allows us with the small knowledge base deal with a considerable variety of problems. In this article we discuss how we interpret and implement a model for creative design, and how it is a successful approach for UI design. In this regard as the basic definition for creative design we have assumed a kind of design which can not directly be derived from the knowledge stored in knowledge resources, instead the system has to take into account some implicit relations to figure out some hidden knowledge which enable the system to find the solution.

Key words　*CBR, Design, Creativity, Dynamic similarity, User Interface (UI)*

1 Introduction

The goal of this work is increasing creativity in design, when using CBR. Since creativity is basically in the nature of design, design without creativity is an incomplete work. Although it is not necessary to always be creative, adding creativity allows considering more alternatives to find a better solution and sometimes the only solution.

Design and specially creative design basically has an ill-structured nature, since most of the time there is no predefined framework to cover the whole task. This makes CBR an appropriate candidate to deal with it. To formulate creative design we have selected the model which is explained in Section 2.1. We discuss how we interpret and implement the concepts of this model.

Our major motivation for using creativity in design is that the (optimal) solution can not always be derived relying on the knowledge stored in knowledge resources, instead the system should extract some implicit relation to find out some hidden knowledge which sums to a novel solution.

Using this idea we have depicted the whole complementary knowledge necessary for reasoning with cases in a network called *Interpretation Network (IN)*. IN interacts with the case base. Whenever IN receives a problem it tries to find any possible interpretation considering both implicit and explicit relations in the network.

As the application we introduce software design or more particularly UI design. In designing such a program usually the human experts rely extensively on their experiences. Besides creativity in using these experiences is a vital issue. In the following we discuss how to build a system which produce creative design, and then how we use this system in UI design.

2 Design

Design is the process of specifying a description of an artifact that satisfies a collection of constraints. Although constraint usually means something that is either satisfied or not, here we mean softer restrictions, such as preferences [1].

The variation of design classes although have been described in many ways, clustering them into three main classes: routine, innovative, creative has been well accepted in this field. In routine design both the knowledge sources and the problem solving strategies are known in advance. In innovative design, only the knowledge sources are known in advance, while in creative design neither is known [1, 11].

2.1 Creative Design

The need for creative design normally arises when the system can not directly derive the solution using the knowledge base. It needs to use some implicit relations which sum to a novel solution.

As the existent models for the creativity process we can point to Wallas's model as the earliest work in this field. Wallas [1] introduced the model in four steps: preparation, incubation, illumination, and verification. Preparation is the process of formulating the problem, incubation corresponds to the generation and formulation of possible solutions, while it is subconscious or partially conscious. Illumination is the stage in which the solution is consciously proposed and the important insight about the problem occur to the creator unexpectedly. Finally verification is the process of evaluating the creative properties.

As the other model we can point to Bhatta and Goel's [2] definition of creative design. They define creative and innovative design against routine design. While in routine design both the variables and the ranges of values they can take are fixed, in innovative design the variables of design problem are fixed but the ranges of values may change, and in creative design even the variables may change. Besides innovative and creative design involves the reminding of analogs from domains different and distant from that of given problem. In their model innovative and creative design involves non-local adaptation of design analogs, and also evaluating candidate design plays a key role in innovative and creative design.

In the model introduced by Gomes et. al [3] the main idea is design space exploration as a way of generating creative design solutions. Other focused issues in this model are problem elaboration phase and adaptation process, such as composition of different parts of design and thematic abstraction.

Wills and Kolodner [4] define their model as: problem re-description, evaluation, assimilation and strategic control. Problem re-description is generating multiple ways of describing a problem which provides several different contexts for specifying what would be relevant, if remembered. Evaluation is the process of not only checking for satisfaction of constraints, but also allowing the evaluative issues to emerge in the course of evaluating. Assimilation is the process of comparing and contrasting alternatives with one another along the dimensions relevant to the problem context. Strategic control refer to process of moving fluidly between various problem pieces and design process instead of following a rigid methodical plan detailing what to do next.

Although we think that all the introduced models for creativity have common points (e.g., reformulating the problem and evaluating the solution), we believe that in Kolodner and Wills's model the steps are more explicit and this model can cover our definition of creative design explained earlier. However we also think that all the other models somehow imply the process explained in this model in other terms.

3 Case-Based Creative Design

Case based reasoning is an appropriate approach for the ill-structured tasks. Design and specially creative design are naturally ill-structured tasks. Also one of the characteristics of design is that designers rely extensively on past experiences in order to create a new design [16]. Therefore CBR prepares a good framework to deal with creative design.

Creative design in CBR perspective is in fact the process of using the well-known pieces of design in unusual way or modify well-known design in unusual way [5]. In the following we discuss how we interpret and implement the concepts described in creativity model.

3.1 Knowledge Resources in CBR

The need for creative design raises because the solution can not directly be derived from the knowledge stored in knowledge resources. In this section we describe the knowledge resources.

Richter [6] has described four containers in which a CBR system can store knowledge. This knowledge may include domain knowledge as well as problem solving knowledge which describes the "method of application" of the domain knowledge inside the container. The four containers mentioned are:

1. The "vocabulary" used to describe the domain
2. The "case base"
3. The "similarity measure" used for retrieval
4. The "solution transformation" used during adaptation

A CBR system developer has to decide how the knowledge should be distributed among the containers to have more efficient system.

3.2 Interaction Between the Knowledge Resources

If there is not enough knowledge available to fill one container as requested, there is a need for knowledge transformation from some containers to others [12]. However in our work we are using the containers interaction instead of transformation, which means in any step of reasoning we are not only considering the knowledge needed to reason in that step but also the knowledge needed for future steps. This will let us be flexible enough to use any knowledge resource in a wider domain.

Adaptation is a compensatory part of CBR [3]. It allows us to consider the knowledge missed from the other parts and derive a better solution. This provides a good framework for adaptation knowledge to interact with other knowledge resources in order to improve the performance. Interaction between adaptation knowledge and similarity measure has been previously used in DéjàVu [8] and RCR [8]. However we consider interaction between all the knowledge resources (see Figure 1). This enables us to have a better view of the whole procedure in each step of reasoning.

4 Reformulating the Concepts of Creativity

As we explained there are four main concepts in the selected model (Re-description, assimilation,

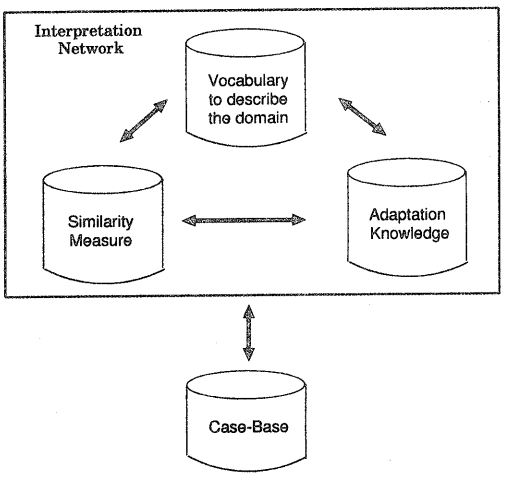

Figure 1: Interaction between Knowledge Resources

evaluation, strategy control). In the following we explain how we implement each of these concepts.

4.1 Re-description

In the Section 3.2 we pointed out how to use adaptation knowledge in other steps of reasoning. Base on that idea, here we want to use adaptation knowledge in situation assessment step which may help us to interpret the problem in any possible adaptable way, and lead us to re-formulate the problem in the first step.

4.2 Assimilation

Assimilation is the process of comparing and contrasting alternatives with one another along the dimensions relevant to the problem context. To formulate this concept we describe a term *dynamic similarity*. By dynamic similarity we mean that two constraint may be highly similar in a situation while they may differ under the other situation.

Aspect [7] defines some aspects for the cases and depict them in a network in which any node is a polyhedron and represents a case. Each face defines an aspect. The link between each face of a case to the face of another case has a weight which defines its similarity. The combination of these weights will give us the total similarity between any two cases.

However we define it in the other way. We define a function for dynamic similarity which takes into account the issues important for finding similarity between two constraint. We are not considering a global case here, rather we find the similarity between two constraint considering the situation dynamically.

4.3 Evaluation

To formulate this concept we have defined another metric as *probability of acting well*. We have not only based situation assessment on adaptability of constraints, but also define adaptability as the base, to define similarity. So we have assumed a weight for each adaptation method. This let us consider the likelihood of deriving a good solution using that adaptation knowledge when anticipating the alternatives.

Dynamic similarity and *probability of acting well* together provide a framework to evaluate the solutions and rank the selected alternatives to derive the best solution.

4.4 Strategic Control

Considering interaction between knowledge resources breaks the strict consequence of predefined procedures to a more flexible one, by giving a global view of the whole reasoning in every step.

5 Computational Model

To implement the model we introduce Interpretation Network (IN). Each node in the IN represent a constraint with which we describe cases. The goal of building IN is :

- To propagate the condition along the network to realize which other constraints should be satisfied and is not directly clarified in the query case (constraint propagation net).
- To find a set of equal explanations for a constraint which enables us to re-interpret the problem or in the other word to expand the search space to get a more complete view of the problem.
- Weighting the similarity between those equal constraints and original one, dynamically considering situation.

- Specifying the adaptation knowledge, required for each equivalent constraint.

A similar work of using network has been reported in [14].The network has been called CRN (case retrieval network) and is in some sense similar to ANNs (Artificial Neural Nets) since the inferences are performed by propagations of activations. New cases are related to former ones by similarity computations. The difference between IN and CRN is mainly in considering adaptation knowledge, and also propagating constraint as a constraint propagation net. Considering adaptation knowledge enable the system to take not only the similar cases into account but also consider any possible interpretation. However the similarity of two methods lets those two to have some common advantages such as performing retrieval in a bottom-up fashion.

We define two type of links in the IN. One sort will present adaptability and will find similarity, while the other kind just propagate the constraint in order to realize what other restrictions it should consider and represent the constraint network. For the first kind of links we define a weight. The weight of the links will present the degree of similarity between two concepts.

The knowledge depicted in the network reflects adaptation knowledge as basis and similarity measure in the second place. The nodes in the network express the vocabulary needed to define the domain. All together we can see three containers of four containers in Richter model has been covered here. The last container which is Case-based itself. Therefore the network covers all the complementary knowledge necessary for case-base, and these complementary knowledge can interact with each other in a flexible way.

5.1 Definition of the Interpretation Network (IN)

We present the network by $N = \{\Lambda, \Theta, \Phi, \Omega, t\}$:
(Λ) which represents nodes in the network, is a finite set of constraints
(Θ) represents the set of links which express constraint network. They connect each constraint to another constraint with logical AND, which means the solution should satisfy both constraints.
(Φ) represents the set of links which express adaptation knowledge, each link in this sense is a pointer to adaptation knowledge which allows making that link. It is not important that what is the supporting source for adaptation knowledge. Therefore it enables us to use any available source of knowledge. This kind of link connects any two constraint with logical OR which means the solution should fulfill either of them.
(Ω) defines similarity measure. It is in fact the weight of the links. It includes the concepts of dynamic similarity and probability of acting well in it in the way that it derives from multiplication between these two metrics. When propagating an action in the network, the similarity measure between two nodes depends on the weight of the links in the path connecting these two nodes. It derives from multiplying the weight of the links in the path.
(t) identifies a threshold for propagation. The action will be propagated along the network while the total weight of a path is greater than t.

Figure 2 gives a general view of the network.

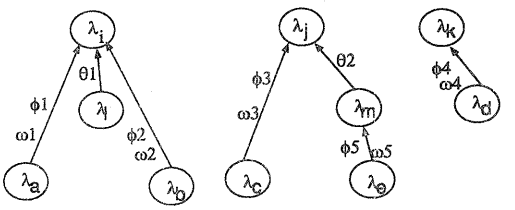

Figure 2: The Interpretation Network (IN)

5.2 How the Network Performs

Defining a query case for the network a set of constraint will be activated, then the action should be propagated appropriately. The system will do it in two steps. Step1 is activating the nodes using Θ links. The output of this step is having a set of constraints which are related to each other by logical AND.

The next step is activating the nodes using Φ links. In this step for each constraint we will have a set of possible alternatives which are connected to each other by logical OR. The system also measures the similarity between the given constraints and possible alternatives in this step. The similarity measure will be done using Ω functions. These functions as we explained are multiplication between two metrics: dynamic similarity and probability of acting well.

Dynamic similarity normally contains a set of issues which are important to make each link. It in fact considers the presence or absence of each constraint and assign a weight to this consideration, so that it can realize under these given constraints (situation) how similar are each pair of nodes.

Probability of acting well, however is predefined. The system will learn by modifying this metric. So it will dynamically change over the time by new experiences. It basically defines the strength of the knowledge source which has made the link.

Totally in this step system propagates the activation concerning the weight of links and continue propagation until it finds the total weight under the threshold. Finally instead of each constraint in the output of Step1 we will have a set of constraints which in fact is a set of possible formulation for that query case.

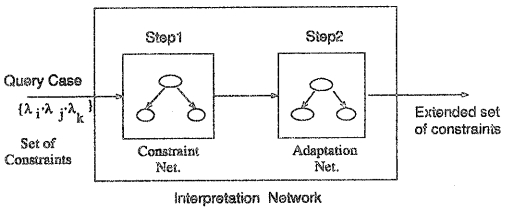

Figure 3: How the IN performs

In the output of the IN there is an extended set of constraints, in which for each constraint in query case and also additional constraints added in Step1, there is an equal constraint. The system then looks for a case in the case base which satisfy an appropriate subset of this extended set. Figure 4 makes this procedure clear.

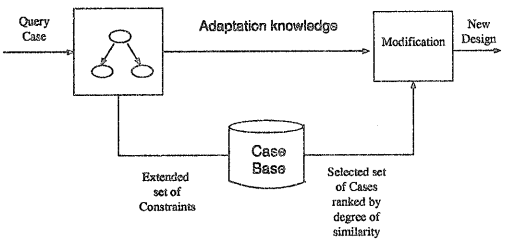

Figure 4: General view of the the system

5.3 Advantages of Using the IN

We can classify the advantages of using network in the following items:

- Prepare a good framework to integrate any available knowledge source for adaptation.
- Cover all the complementary knowledge required for reasoning with cases .
- In order to simplify the process of retrieval usually case base systems use indexing to make the process faster and by generalization protect the system against missing some relevant cases. The network will address these problems by considering any possible interpretation instead of generalizing, and measuring similarity for those cases which have the possibility to be similar and adaptable, instead of doing it for the whole case base to make it faster.
- Possibility to combine the network with constraint net
- Provide a good framework for interaction between knowledge resources by propagating the action and considering the weight.

6 Learning

One of the important concepts in CBR is learning. Learning in our system can be done in two ways: Learning in the IN, Learning in the case base.

Hammond [13] has defined an interesting framework for learning from failures in CHEF. Whenever CHEF gets aware of a faulty plan recognizes the cause of the failure by a causal model, then considers the possibility of having failure under similar situation and add a goal to its goals, which satisfying this additional goal will protect the system against having failure. Back to the IN, the similar action will be done in constraint network. It is by making a new Θ link that the system considers a constraint which satisfying it saves the system of repeating a failure.

Learning a new adaptation knowledge is possible through making a new Φ link. The system can also improve the performance by modifying the weight of the links using the experiences.

7 UI Design as an Application

Before presenting details we have to explain what is UI design what are the important issues and finally why creativity is necessary and why we think CBR can be appropriate.

UI design is a task that normally an expert relies on its experience to design a new layout. A user interface should be easy to understand, easy to use and should be aesthetically pleasing. Therefore although we can find a number of rules in this domain but the experts mainly rely on their experiences (cases). They have a library of experiences in which they know which of the designs were successful and which failed and why. So whenever they get a new demand they try to use these experience to design a successful UI.

The designer should also have a creative nature to understand the user's demand and interpret it in the right way which enables him to have wider view to the problem. Also sometimes it is necessary to extract some implicit knowledge from the experiences which sum to a novel design, which we call creative design.

To build our system we have collected a library of cases which is a collection of useful user interfaces. Over the time it will save more cases and get more complete. To describe any case we have defined a number of features with which we can cover the case description. In this regard we have considered the lessons each case can teach and the goals it can satisfy as the basic factors. The solution of the case is the generated code for that UI program and also the view of how it looks like. The cases are stored in a case base which contains their description, solution, and some warnings for the possible trade-offs as outcome.

The user of the system will explain his request with the terms that the system offers. In fact the user will explain the kind of input and output which he wants to have and also the aim of the UI by describing the function over the input which will give us the output. The request page with which the user explains his request to the system looks like Figure 5.

Choosing each items of input, output or function list in the request page gives another window in which the user can specify his demand in more detail. Figure 6 shows a sample of how the user explains the function.

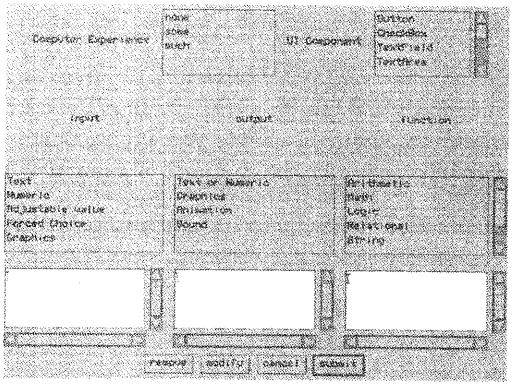

Figure 5: Request page

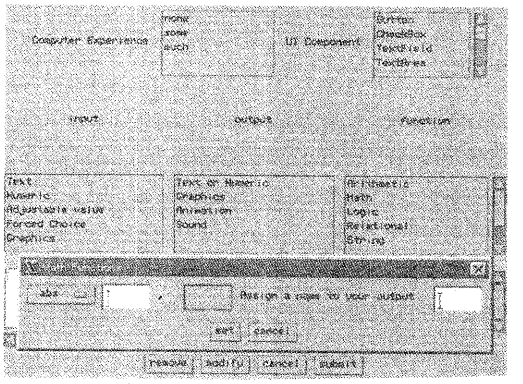

Figure 6: Detailed request page

The system tries to re-interpret the user request in any possible way and find a similar adaptable case in its case library. Then in the next step using adaptation methods it tries to adapt the solution to fit he problem. The kind of adaptation which we are using in this work are substitution and transformation. In this way for each possible adaptation we have defined a method (e.g., a method which substitute a button with a checkbox). So when the system finds the most similar case considering the necessary adaptations it will call these methods and modifies the case.

Dynamic similarity here is in fact the similarity between any two input considering the function which we are willing to do over them. For instance numeric input and adjustable value for input can be similar if the goal is just to compare two value, and it is possible through an

adaptation method which converts TextField to a Scrollbar.

Also the system has another advantage and that is, the final answer is still modifiable which means the user can modify the final version of solution in the way that it can delete some components or add some new components or even substitute some components, The system will apply these modification doing the same process of adaptation by calling an appropriate method.

The system finally stores the new case for future use if it is worth storing. To distinguish whether a case is worth storing, the system analyzes how many adaptation methods it has used to derive the solution. If the number of the adaptation methods are high the system assumes the case as worth storing since it can not be easily derived from the case base.

As the similar work we can point to the system AskJef [15] which gives a prototype of a human-machine interface similar to the one that the user is asking. It also gives some guidelines. However the adaptation is left to the user.

8 Conclusion

To raise the performance of a system it is not always necessary to validate the knowledge base. Adding creativity to the system enables the system with the limited knowledge base deal with a wider task. In this article we have introduced the model for creativity which we have selected as the base of our work. We have described the concepts of this model and for each of them we have suggested a way to implement it.

An application is in UI design. In this domain designers normally rely on their experiences to produce a good UI. Creativity is also an important factor in this field. We have explained how our system addresses these issues in this domain. To implement the system we have used Java language. UI cases are also written in Java.

References

[1] S. C. Shapiro, *Encyclopedia of Artificial Intelligence*, Wiley-Interscience.

[2] S. Bhatta and A. Geol, Innovation in Analogical Design: A Model-Based Approach, *Proceeding of third International Conference on Artificial Intelligence in Design (AID-94), Aug.15-18,1994, Lausanne, Switzerland, pp.57-74.*

[3] P. Gomes, C. Bento, P. Gago, E. Costa, Towards a Case-Based Model for Creative Process, *Proceeding of ECAI-96.*

[4] L. Wills and J. Kolodner, Towards More Creative Case-Based Design system, in D. Leake's *Case-Based Reasoning Experiences, Lessons & Future Directions.*

[5] J. Kolodner and L. Wills, Case-Based Creative Design, *AAAI Spring Symposium on AI and Creativity, Stanford, CA, March 1993. Reprinted in AI and Simulation of Behavior Quarterly (85), Autumn 1993.*

[6] M. M. Richter, The knowledge contained in similarity measures, *Invited Talk on the ICCBR-95, Sesimbra, Portugal.*

[7] F. Gebhart, A. Voss, W. Grather, B. Scmidth-Belz, *Reasoning with Complex Cases*, Kluwer Academic Publishers(1997).

[8] B. Smyth, M. T. Keane, Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval, *First European Workshop, EWCBR-93, Germany.*

[9] D. Leake, A. Kinely, and D. Wilson, Case-Based Similarity Assessment: Estimating Adaptability from Experience, *Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI Press, Melno Park, CA, 1997.*

[10] J. Kolodner, *Case Based Reasoning*, Morgan Kaufmann Publishers, 1993.

[11] A. Goel, Design, Analogy, and Creativity, *IEEE Expert, Vol.12, No.3 , May/June 1997.*

[12] W. Wilke, I. Vollrath, K. Althoff, R. Bergmann, A Framework for Learning Adaptation Knowledge Based on Knowledge Light Approaches, *Adaptation in Case-Based Reasoning: A Workshop at ECAI 1996 in Budapest.*

[13] K. J. Hammond, Case-Based Planning: A Framework for Planning from Experiences, *The Journal of Cognitive Science. Ablex Publishing, Norwood, Nj. Vol. 14, No. 3, September 1990.*

[14] M. Lenz, H. Burkhard, Case Retrieval Nets: Foundations, Properties, Implementation, and Results, *Technical report, Humboldt University Department of Computer Science, Berlin, 1996.*

[15] J. Barber, S. Bhatta, A. Goel, M Jacobson, M. Pearce, L. Penberthy, M. Shankar, R. Simpson, and E. Stroulia, ASKJEF: Integration of Case-Based and Multimedia Technologies for Interface Design Support, *Second International Conference on Artificial Intelligence in design, Pittsburgh, June 1992, pp. 457-476.*

[16] A. Gomez, M.L. Maher, Design by Interactive Exploration Using Memory-Based Techniques, *Knowledge-Based Systems. Vol. 9, No. 1.*