Model-Based Product Redesign

LI Zhan-Shan†,††,KOU Fei-hong†,††,Cheng Xiao-chun†††,Wang Tao††††

†College of Computer Science and Technology , Jilin University , Changchun ,130012 China
††National Laboratory of Symbol Computation and Knowledge Engineering, Changchun,130012 China
†††Department of Computer Science, The University of Reading, Reading, RG6 6AY, UK
††††Department of Computer Science and Engineering, Changchun University of Technology,Changchun,130012,China

Summary
Based on Model-based theories, this paper explores the application of model-based reasoning for product design, proposes the concept of redesign, analyzes design conflicts, then develops an algorithm to solve system redesign problems, and finally compares with existing research results.

Key words:
model-based reasoning, design conflict, redesign.

1. Introduction

When one has to design a new product in manufacturing field, often an existing , somehow similar product is used for the new design. Therefore design process is often a redesign. The reuse of a given design can be a good way of developing a new product. As reuse means reusing knowledge, machinery etc., redesign can be seen as a good approach to lowering cost and improving efficiency. If new requirements have arisen that ask for a famous brand new product, then trying to adapt an existing product is a waste of time and cost, to the problem, it is necessary that we should study a solution to decreasing the waste of time and cost for redesign. the advantages of redesign is that a part of the new product is known in advance, and conversely, the part that needs to be adapted, has to be determined. Although the problem has been investigated in reference[1], unfortunately, the authors did not formally characterize them. In recent, some researchers[2-3] have been applying Model-based reasoning methods to this problem. Their main idea is to regard new requirements for the system as constraints or observations in the system to be diagnosed, and compute the part of the system to be altered using the built system models, but their efforts were limited to reassign values to some attributes in the system, so that the results were not suitable to the situates that the components or structure needs to be altered. In the paper, based on results which we have obtained [4-5], we investigate the solutions to the problems described as above within the diagnosis framework, we propose the notions of redesign problem, redesign conflict, redesign diagnosis, redesign, and give an algorithm to solve redesign problem.

The rest of this paper is organized as follows. Section 2 first describes product redesign, then provides necessary definitions. Section 3 gives an algorithm to solving redesign problem according to the results in Section 2. Related work and conclusions are discussed in Section 4 and 5.

2. Product redesign

Product design plays an important role in the production process of manufacture, which has a direct effect on the quality and cost of a product, so how to improve the quality and efficiency of product design is a problem to which designers must face. When one has to design a new product in practice, often some existing somehow similar products are used as the basis for the new design, therefore design process is often a redesign. Therefore, the problem that designers must solve is how to redesign an existing product to satisfy the new requirements. This section characterizes the methods of solving the problem within the diagnosis framework. Reconfiguration for a product is a special case of design activity[6],which includes not only the choice of parts but also their connections and assigning values to parameters. The objective of reconfiguration for a product is the same as that of diagnosis finding out suspect components, besides, the process of configuration need decide what requirements to be met and product

Manuscript revised January 30, 2006.

configurator can compute proper components, the connections between them and assigning values to parameters, to satisfy these requirements. To present reconfiguration problem using diagnosis methods, we still use those terms in diagnosis.

Component is a part of a system, which provides some functions, from diagnosis viewpoint, if the choice of a variable value has an effect upon the effectiveness of a configuration, then the variable can be viewed as the cause for the abnormal behavior, such that the variable is regarded as a component. As follow, we illustrate the process of reconfiguration with an example[1].

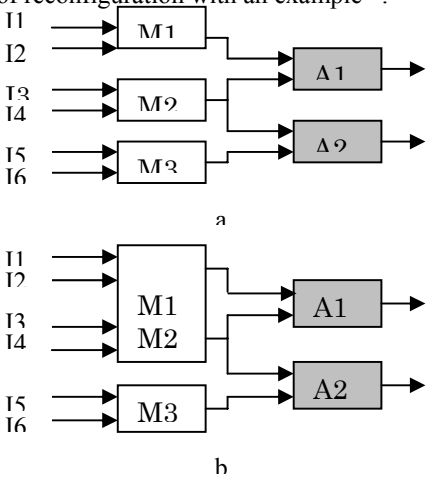

Fig2.1 redesign for a circuit

Example: figure 2.1 shows a circuit consisting of multipliers and adders to be redesigned, where $O=pI_1*qI_2(p=3,q=2)$,is the instance specification of multiplier in it, the behavior of a multiplier M is described by variables p ,q(instantiated 3 and 2) , function * and its variables I1 and I2. Therefore, to meet new requirements for the circuit, we can reach the reconfiguration through three methods:

Firstly, reassign values of parameters instantiation. For instance, Modify the specification to be $O=2I_1*I_2$ by reassigning the new value 2 to p and 1 to q.

Secondly, replace a component with another one. For example, we use adder specified as $O=4I_1+3I_2$ instead of the multiplier, which is considered as changing function, parameters and varieties of a component.

Finally, change structure of some parts of a system. There are two adders that are replaced by component M1M2 which is specified as $O_1=2I_1*5I_3*3I_4$, $O_2=I_1+2I_2*I_3$ See figure 2.1 b.

The redesign above mentioned, which is a descriptive modification indicates that it is feasible to do this modification. However, how to make these modification is usually done by experts of design department. Therefore, it is incomprehensive and less effective to solve this problem inevitably. In order to improve the quality and efficiency of the product design, this section we will describe automatically calculating the modification of system using model-based diagnosis. The following definitions are introduced necessary.

Definition 2.1 (Design Problem) In general ,we assume a design problem is defined by a pair (SD,SRS) where SD and SRS are set of logical sentences.

SD represents a system description(a design knowledge base),and SRS specifies the particular system requirements which is usually provided by users.

Definition 2.2 (Design) Given a design problem (SD,SRS) ,a design CONF is a disjunction of literals : $type(c_i,t_j)$, $val(c_r, a_s, v_t)$ $c_i,c_r \in COMPS$ $a_s \in ATTRS$, iff $SD \cup SRS \cup CONF$ is consistent and $SD \cup CONF| = SRS$.

This definition allows that a design is not only consistent with SD and SRS, but also SD and CONF can draw SRS. A design is a minimal design iff no other CONF′ such that $CONF' \subset CONF$ is a design. A minimal design can save the cost of a product.

Definition 2.3 (Redesign Problem) Let a design CONF for a design problem (SD, SRS) , a redesign problem for CONF is defined by a pair (SD,SRS′),$SRS' \cap SRS \neq \phi$ and $SRS' \neq SRS$,where SD and SRS′ are set of logical sentences.

Definition 2.4 (Redesign Conflict) A conflict C for a redesign problem (SD,SRS′) is a literal set of part of COMPS and ATTRS, i.e. $C=\{type(c_1,t_i),type(c_2,t_j),...type(c_n,t_k),val(c_i,a_j,v_k),...,val(c_r,a_s,v_t)\}$ such that SD and SRS is inconsistent with C.

Because customers alter the system′s requirement, a redesign conflict which is consistent with SD and SRS has become inconsistent with SD and SRS′.Therefore, the redesign conflict is relate to the modification to the existing system(the design CONF), which is the beginning we work. A redesign conflict C is a minimal redesign conflict iff no other C′ such that $C' \subset C$ is a redesign conflict.

Definition 2.5[7] Given C is a collection of sets, $\sigma \subset \cup_{S \in C} S$ is a hitting set of C, such that $\forall S \in C$, $\sigma \cap S \neq \phi$.A hitting set is minimal iff no proper subset of it is a hitting set.

Theorem 2.1 (Redesign Diagnosis) Suppose (SD, SRS′) is a redesign problem, CS is the collection of all minimal redesign conflict sets for (SD, SRS′). DIAG is a redesign diagnosis iff DIAG is a minimal hitting set for CS.

It is obviously that the literals of DIAG represents the components and values of CONF which is inconsistent with SRS′.We consider DIAG as the part of what to be altered to the original design CONF. As for different types of literals in a design, there are a lot of redesign methods dealing with them correspondingly. Here we mention three methods such as reassign value of attribute, replace component and change structure of the original design. In general, the complexity of the three methods is increasing step by step.

Definition 2.6 (Redesign) A redesign CONF′ based on redesign diagnosis DIAG is consistent with SD and SRS′, and SD∪CONF′∪CONF¥DIAG|=SRS′.

Given a general product tree , because of new system′s requirements SRS′, a design that is consistent with SD and SRS became inconsistent with SD and SRS′.There are more than one redesign diagnosis for a redesign problems, As for a DIAG, According to the theorem 2.1 ,the literals of DIAG is what need to be alerted, so that CONF¥DIAG is the unnecessary changed literals of the original design CONF, i.e. it is the remaining parts of CONF,Which is also a subset of redesign.

In the next section we provide an algorithm to solving redesign problem based on redesign diagnosis DIAG.

3. Redesign Problem Solving

According to the results discussed in Section 2, solving redesign is focus on CONF¥DIAG. In practice, we reach the redesign via altering the literals of DIAG. Thus how to alter the literals became the most important task. We now employ an algorithm (cf. Figure 3.1) based on the generic product tree. . Because there are two different types of literals in DIAG, val and type, we adopt two functions (methods reset and replaceComps)to deal with them respectively in step 3. As for literal val, configurator reset value of the attribute. But for literal type, our algorithm is to find a replaceable component instead of the component labeled by this literal. Actually, extending a component is to set values of all attributes of it. Generally speaking, the complexity of method reset is lower than that of method replaceComps. So we should alter val literal firstly. Our algorithm sorts the order of the two types of literals based on the complexity of them(method sort in step 2). Whatever a val or type literal, if their methods can not find a new value or component that are consistent with SD,SRS′ and CONF′ ,we drop them from open_list.

Algorithm Reconfigure (DIAG,SD,SRS′,T)
{
1. Initialize: open_list= DIAG; i=0;j=0; isRedesign=false;
2. Sort(open_list);
3. while (open_list ≠∅)
choose the first literal L from open_list, data=L;
if data is a val literal, i=resetValue(data ;T);
else Given R is the set off replaceable components of data in T,calli=replaceComps(data,R);
case i is
- 0: CONF′=CONF′ ∪ {data}; isRedesign=true; return CONF′;
- 1: CONF′=CONF′∪{data}; delect L from open_list goto 3;
- 2: delect L from open_list,goto 3;

4. Insert the literals of the unextended component into UnExtendComps, call j=extend(CONF′,UnExtendComps);
5. If j =0 then isRedesign=true;
6. Return CONF′;
}

Fig. 3.1

If we can not obtain a redesign after all literals of DIAG having been altered, Our algorithm will get all unextended components in the generic product tree and extend them into CONF′ (method extend)in step 4. There may be many unextended components in the generic tree. Which ones we should first extend can find a solution quickly and easily. We can make use of some heuristics to guild solution search. One way is extending the literals in SRS′¥SRS , which represent system′s new requirements that the original design can not satisfy and should have a prior to the others, so that configurator could consider them preferentially. Another way is to extend the literals respecting the extended literals. There are many other ways to guild extending components. We will discussed them in detail in future.

It is possible that the system′s requirements is unreasonable or beyond the SD. Therefore, when having extended the whole generic tree, the algorithm can not find a redesign all the same. It means that SD is inconsistent with SRS′. Here we should change the system′s requirements.

In addition, there may be many design diagnosis for a redesign problem. Using our algorithm, we can have more than one redesign, i.e. a redesign problem may have many solutions. Among these solutions, which one is the best should be decided by the use of valuation function, experts or decision-maker.

4. Discussion and Related work

We have presented the definition of redesign aiming at the existing product design and the solving algorithm using the redesign diagnosis. In the face of requirement changing quickly, designers must think out different kinds of new products to adapt to this situation. In fact, in order to improve the quality and efficiency of product design, most new products are developed on the basis of some existing similar products. Therefore, designing a product often can be seen as redesign. In a certain extent, the ideal product redesign is the key to success of enterprise. Product redesign has became a problem faced by

enterprises. There are many experts having been done research for this problem and presented some methods to solving it.

From a common viewpoint of consistency-based reasoning, diagnosis is similar to configuration. Stumptner and Wotawa[2,3] have presented their method based on model diagnosis to solving redesign. Compare to their work, we both make use of solving conflict based on consistency diagnosis, but the difference is the point of view on the same problem and the capability of solving algorithm. In one place, SD is different, they consider a particular system(in this case the existing configuration) and a description of its behavior as SD. They solve diagnosis based on the existing configuration, which is a set of resetting the modes of the components in the system. i.e. the components, attributes and the description of behavior of reconfiguration is not beyond the existing configuration. Confronting with all kinds of user's new requirements, specially some new functions realized by the components besides SD they defined, their method can not solve this problem. This paper is not only to reconfigure the components of the existing configuration that inconsistent with new requirements, but also to extend new components and attributes in the generic tree. Therefore, the capability of our solving algorithm is more powerful than theirs. Another difference is the mode of treating with diagnosis. They consider a diagnosis component as a parameter that has three modes, so that a diagnosis is set of mode setting of related components that is consistent with SD and OBS. If the diagnosis satisfies with a filter condition, it is a reconfiguration. The diagnosis we defined is a set of components of the existing configuration that is inconsistent with new requirements, which may contain components and attributes not in the existing configuration but in the generic tree. Furthermore, our solving method is to alter the literals of DIAG via resetting value and replacing component to seek redesign solution.

5 Conclution

In this paper, we have described the application to product design based on model-based diagnosis and presented the concepts of redesign diagnosis. In additional, we have developed a problem solving algorithm for redesign based on diagnosis.

Acknowledgment

Our researches had been supported by the National Natural Science Foundation of China.

References

[1] Bakker R.R, Eldonk S.J.M and Mars N.J.I. The use of model-based dagnosis in redesign. In: Cohn edit, 11th European Conference on Artificial Intelligence, John Wiley &Sons Ltd,1994

[2] Stumptner M and Wotawa F. Model-based reconfiguration. In: *Proceedings of Artificial Intelligence in Design (AID-98)* , Lisbon, Portugal, 1998.

[3] Stumptner M and Wotawa F. Reconfiguration using Model-based Diagnosis.In: *Proceedings of the International Workshop on Diagnosis,1999*

[4] Console L, Friedrich G eds. Model-Based Diagnosis. Basel-Switzerland, Science Publishers,1994：

[5] JIANG Yunfei LI Zhanshan. On component replacement and replacement tests for model-based diagnosis. CHINESE J. COMPUTERS,2001,24(6):666-672)

[6] Li Zhan-shan,Wang Tao and Sun Ji-gui. System Replacement Repair and Reconfiguration by Using Model-Based Diagnosis.Chinese J.of Jilin University(Science Edition),2003,41(1): 45-48

[7] Stumptner M.,Anoverview of knowledge-based configuration. AI Communications, 1997,10(2):1-15

[8] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987,32(1):57-96

Li Zhan-shan received Ph D., from Jilin Univ. in 2000. After working as a lecturer (from 1999) in the Dept. of Computer Science and technology, he has been an associate professor at Jilin Univ. since 2001. His research interest includes model-based diagnosis, knowledge representation and problem solving of configuration, and intelligent plan and decision making.