Case Adaptation in a Case-based Process Planning System

Hao Yang† and **Wen F. Lu**

Department of Mechanical and Aerospace Engineering
University of Missouri-Rolla
Rolla, MO 65401
wflu@umrvmb.umr.edu

Abstract

This paper describes the case adaptation in a case-based process planning system: PROCASE. PROCASE is an acronym for Process Routines Organized as Case Archives with Simulation Environment. In a case-based process planning system, a new process plan is generated by adapting an existing similar process planning case. Case adaptation is an important and, most of the times, difficult issue. This is because first, usually, an existing case may be a similar case but not an identical case. Adaptation is essential to tailor this similar process planning case to generate a new process plan which can produce exactly the new part needed. Second, adaptation involves many reasoning processes which embeds a great amount of knowledge. To encode such knowledge for computer simulation is not a plain task. The case adaptation in PROCASE comprises case modification and case repairing. This paper will first briefly introduce the case representation and case retrieving in PROCASE. Then the rest of the paper will present the case adaptation in PROCASE.

† Currently is with Wizdom Systems Inc. 1300 Iroquois Ave., Naperville, IL 60563, xyzhu@interaccess.com

Introduction

Case-based reasoning (CBR) (Riesbeck, Schank, 1989) has been applied to a wide diversity of areas such as military strategic planning (Goodman, 1989), conceptual design (Sycara, Navinchandra, 1991), and assembly planning (Pu, Reschberger, 1991). However, few papers have been published in applying CBR in manufacturing process planning. All the systems developed in the domain of mechanical design and manufacturing did not have rigorous simulation mechanisms, therefore, could not have powerful and complete case adaptation modules. This paper introduces the case adaptation in a case-based process planning system.

In order to convert the design of a mechanical part from the blue print to a real part, a manufacturing process plan which contains all the detailed instructions of each machining process needs to be generated. This planning process is costly and usually heavily relies on the experience of human experts. Traditional computer aided process planning (CAPP) systems apply group technology (GT) or rule-based expert system to achieve automated process planning. However, using group technology can not achieve a fully automated system (Chang, 1990). On the other hand, rule-based systems (Atling, Zhang, 1989) are costly to build and expansive to maintain. To overcome this limitation, a case-based process planning system PROCASE was developed by the authors to explore applying case-based reasoning in generating process plans for machining of rotational parts (Yang, Lu, 1993). *Figure 1* shows the general structure of PROCASE. PROCASE contains four major elements: *retriever*, *modifier*, *simulator*, and *repairer*. The retriever retrieves the "most similar case" from the *plan case base* and uses it as the "plan candidate". If the retrieved plan can not produce exactly the part required, the plan will be modified. However, the modification of the plan is accomplished at an abstract level. All the parameters of the process (feed range, depth of the cut, *etc.*) in the plan have not been determined yet. On the other hand, the plan is not guaranteed to be perfect (or even correct). The simulator is then engaged to assign (instantiate) all the necessary specifications to the plan, as well as to check the feasibility of the plan. If any error is detected during the simulation, the repairer is activated which utilizes the failure information provided by the simulator to repair the plan. And, again, the repaired plan is verified by the simulator. If the repairing is successful, PROCASE presents the successful plan and stores the successful plan in the plan case base.

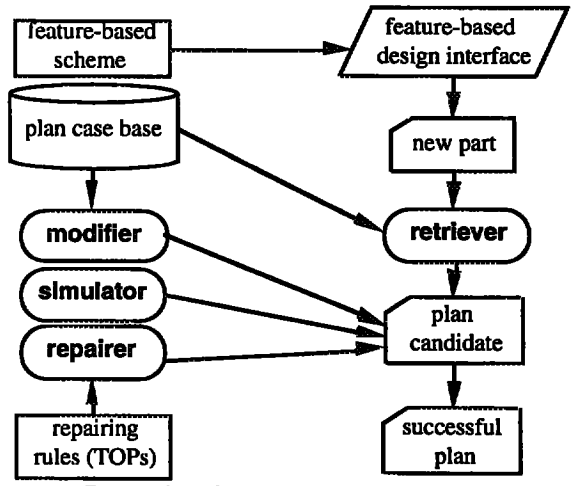

Figure 1. The structure of PROCASE

An Introduction of PROCASE

A feature-based representation scheme was developed for PROCASE to represent rotational parts (Yang, Lu, Lin, 1992). *Figure 13* in Appendix provides some examples of the features used in this paper. In the proposed part representation, each geometric feature is called a *part primitive* (or *primitive* for short) and is assigned a unique code, called *primitive code*. Then, a part can be described using a *part description string* (or *part string*). A part string usually has a *main string* and a *sub-string*. The main-string is a list of part primitives along the axial. A sub-string is also a list which may contain keyways, holes, heat-treatment, *etc*. Each cutting process is viewed as the removal of a geometric feature from the current base material, and it is given a code, called *operation code*. Since each of the intermediate part status can also be represented with part primitives, a process planning case is represented with all the intermediate *part strings* with arcs pointed by operation codes and the clamping conditions. Therefore, the causal relation of the part status and the process can be represented symbolically. *Figure 2* shows an example of partial process plan case. There are two cutting processes involved: grooving and chamfering, which changes the part states from PD31_PD19 to PD31_PD17 and then to PD31_PD07.

PROCASE uses features as indices to retrieve a similar case. The retriever calculates the similarity between the two parts according to the primitives contained in two parts. The similarity value is calculated in two stages: static evaluation stage and dynamic factor adjustment stage (Yang, Lu, 1993). In the static evaluation stage, the similarity value is calculated based solely on the geometric features contained in two parts. In the dynamic factor adjustment stage, the similarity value calculated in the static evaluation stage is adjusted through the sub-features of the two parts.

For simplicity, only the static similarity factor calculation is introduced here.

The static similarity equals:

similarity=

$$\frac{\text{matched primitives in both main strings} \times 2}{\text{total primitives in both main strings}}.$$

The matched part primitives are evaluated by largest common sequence (LCS).

Modification of the Retrieved Case

The process plan that the retriever finds, usually, will generate a part which has some features that are wanted and some other features that are not desired. Also, some desired features may be missing. The modification is to add and to delete some processes in the plan so that the modified plan to make a part that retains the wanted features, discards the unwanted features and adds the missing features. The algorithm

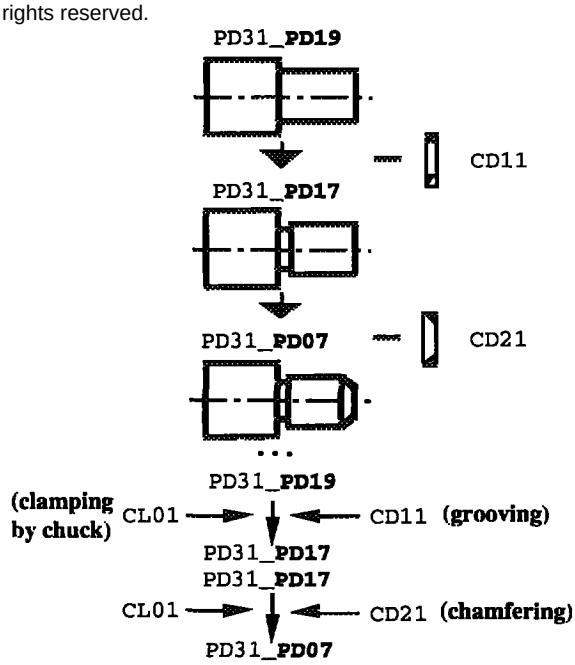

Figure 2. An example of partial process planning case

of case modification is not to be listed here due to the limitation of the size of this paper. The implementation detail of the modification is seen in (Yang, Lu, 1993). Generally, if the modifier finds some features (primitives) of the part that need to be added to the part of the retrieved plan, it first searches the *modification case base*, which stores the previous successful modification cases, to find an applicable modification case. If it fails to find one in the modification case base, it looks through the *plan case base* to find any partial plan in the previous planning cases which can be used to modify the retrieved plan.

Simulation

A case-based planning system does not pretend to be error free. The errors may arise because of insufficient (or wrong) information about the problem, a misjudgment in retrieval mechanism, or a faulty adaptation rule or algorithm. PROCASE establishes a verification mechanism which can detect any fault in the plan due to various reasons.

In PROCASE, the verification is divided into backward and forward stages. First, in the backward verification stage, the simulator starts from the finished part and tries to fill the material "back" to the blank stock so that the part "recovers" to its original form of blank stock. The simulator assigns all the parameters of the processes such as the length and the depth of the cut or the facilities and the time to perform the heat-treatment. This is known as instantiation. The instantiation has to be backward, because only the final state of the part is known and it contains all the details of the desired part. *Figure 3* gives an example of the backward verification. The descendent part status

PX81_PD31_PD78_PD18 united with the process CD01 results in the antecedent part status PX81_PD31_**PD18** (the bold case indicates the "active" features, which means these features are to be changed in form or size in the current process). In the backward verification, the operation code, the initial part status, and the resulting part status are the antecedent part of a simulation rule. The consequent part of the rule is a sequence of operations which will assign the values (*e.g.* dimensions, hardness, tolerances) to the original part status and the process (operation).

After the backward verification, the antecedent part status of each process has been instantiated. The forward verification, which goes the opposite direction, is then engaged to verify each process. The forward verification sounds redundant. However, it is necessary in PROCASE, because, forward verification is a true simulation. In the forward verification stage, starting from the stock materiel, the simulator simulates every process against every intermediate part status. Such a simulation knowledge is more natural than the backward verification. Flaws of the plan are easier to be detected due to the same reason. In PROCASE, the detected error is used to repair the plan. The forward verification performs the following tasks: (1) It verifies the feasibility of each manufacturing process, (2) It checks the clamping conditions and the machinability condition, (3) It decides the actual precision and surface finish the process can reach and (4) It counts the total cost of the process plan.

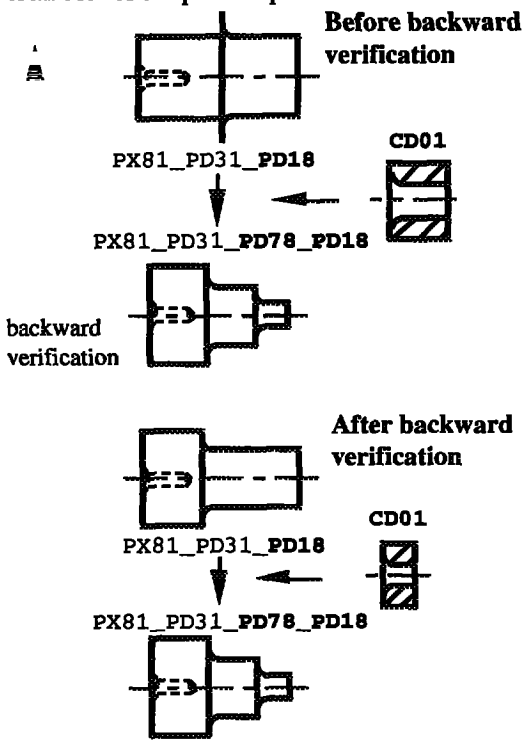

Figure 3. An example of backward verification

Case Repairing

The structure of TOP (Thematic Organizing Packet) is adopted to achieve repairing in PROCASE. This process is similar to the plan repairing in CHEF developed by Kristian Hammond (1986).

A repairing TOP in PROCASE contains a failure type, a set of therapies associated with this failure type and the algorithms to apply the therapies. Most of the times, these repairing therapies are instructive rather than decisive. They are tried one by one. Many attempts, however, largely increase the chance of repairing success. This is like a doctor to treat a patient. Given certain symptoms, a doctor will try certain medicines. If the medicines do not work in the next few days, the doctor will change them to another group of medicines.

The whole process of repairing is like this: The simulator proceeds backward verification first. If the backward verification passes successfully, the plan is delivered to forward verification. If the forward verification is successful, the plan is output to the user. Otherwise, the repairer is activated to repair the plan. The error message detected in the forward simulation is used as the index to find the repairing TOP. For example (*Figure 4*), if the error message is: process blocked, not clampable, feature(BP_CL_FE), which means the process can not be proceeded, because the feature is not clampable. The therapies under which are: change clamping feature; change clamping methods; and exchange process sequence.

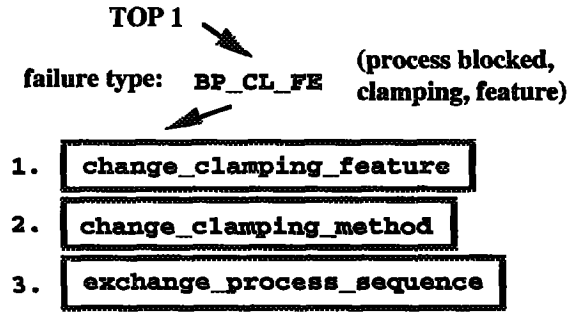

Figure 4. An example of a repairing TOP

What kind of failures PROCASE might encounter? For a machining process, the failure may occur if the material is too hard to cut. Also, failure may occur if the clamping is applied on the surface that is to be machined or the feature is not clampable (*Figure 5*). Failure may occur if the feature is not able to be generated by the process.

Sometimes, the process is applicable. The failure can not be detected until the end of the process, when the finished part is compared with the desired part. If the adapted plan can not produce the finished part which meets all the specifications (*e.g.* dimension, tolerance, surface finish, material property, *etc.*) of the desired part, the plan is considered a failed plan. In order to

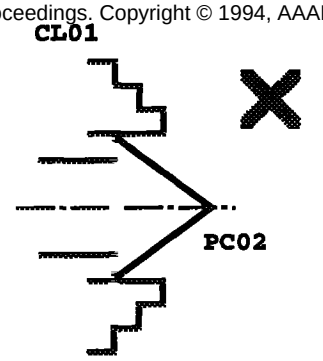

Figure 5. An example of unclampable feature

detect the reasons existed in the early stage of the machining processes, which finally result in a failed plan, PROCASE keeps the record of the precision degrade in the whole simulation process. A warning message will be generated when any quality degrade is detected. This warning message may be used to guide the repairing process.

Currently, there are eleven possible failure types (error messages as well as warning messages) collected by PROCASE. Five of them are error messages such as the aforementioned `BP_CL_FE`. Six of them are warning messages. Real world problems may generate much more error and warning messages. It takes time to list all of them. As an experimental system, PROCASE is not going to develop a complete list of the messages but rather a methodology in dealing with such problems.

An example

The following is an example introduced to explain the adaptation algorithm. The example is kept as simple as possible to reduce the length of illustration.

Initially, there were seven cases in the case library. *Figure 6* shows two finished parts of two of the seven cases. The new part input to the system is shown in *Figure 7.*

The retriever searches the case library and finds the most similar part(with similarity 50%) and the process plan case `prochis007`, which is shown in *Figure 8.* The modifier is activated to modify the plan so that the new plan will produce the new part.

The modification is to be made such that the part primitive `PD43` in case `prochis007` is substituted by `PC01`. The modifier finds the process making `PC01` in `prochis002` (PROCASE knows the equivalence between `PC01` and `PC02`). The extracted process is shown in *Figure 9.* This partial process is inserted to `prochis007` so that the modified process plan would produce the part `PC01_PD07`. *Figure 10* shows the modified plan.

The modified plan is then delivered to backward verification. In this case, the backward verification went smoothly. However, in the forward verification, an error is detected when the clamping is applied on feature `PC01` (in *Figure10*, the primitive code with

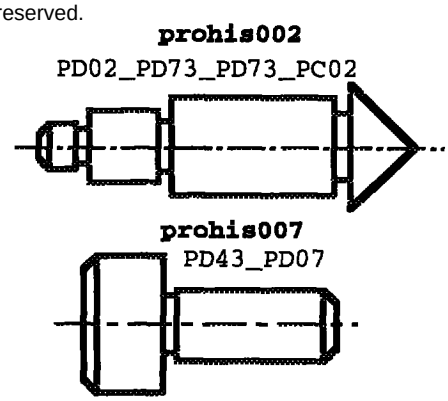

Figure 6. Cases in PROCASE case library

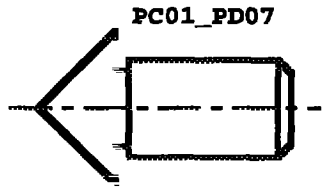

Figure 7. The new part

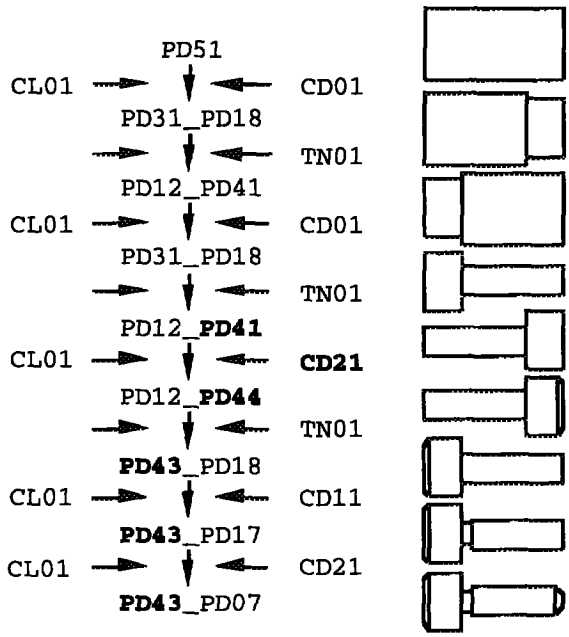

Figure 8. The retrieved most similar case `prochis007`

bold faces are clamping features). The process is blocked and an error message `BP_CL_FE` is sent to the repairer.

Through the simulation error message `BP_CL_FE`, the repairing TOP in *Figure 4* is activated. *Figure 11* depicts the flow of the repairing graphically. The repairer applies each of the repairing therapies one by one until the repairing succeeds. The first repairing therapy 'change clamping feature' does not succeed because, although it is possible to move the clamping

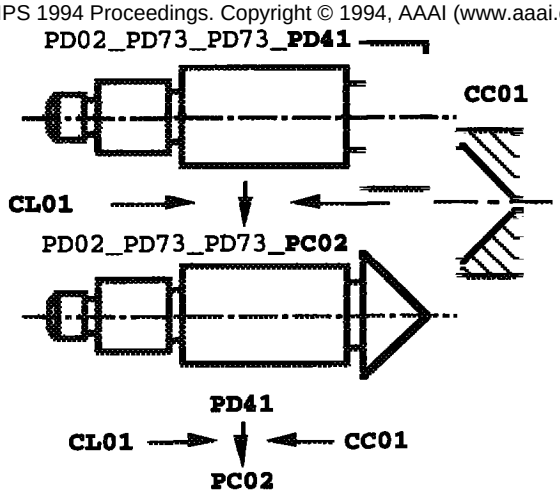

Figure 9. The partial plan is extracted from case `prochis002`

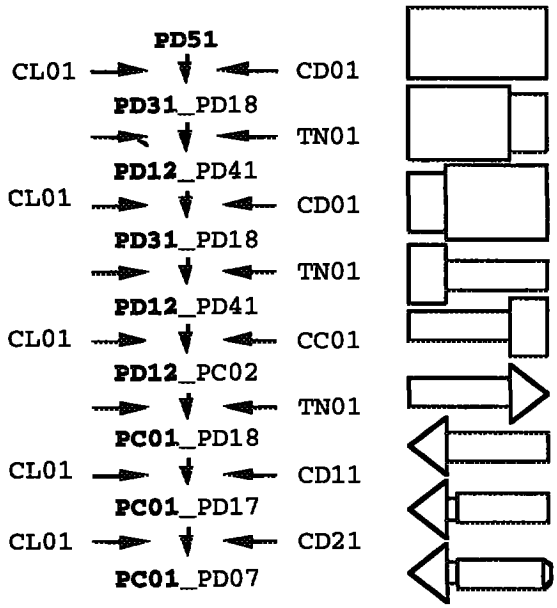

Figure 10. The plan modified from `prochis007`, (The bold cases show the clamping features)

from feature `PC01` to feature `PD18`, this feature itself is active feature, which means it is to be machined so that clamping and machining conflict. The second repairing therapy 'change clamping' failed because clamping method 2 (one side clamp by chuck, one side center hole holding) and clamping method 3 (center hole holding at both ends) need center hole(s) being machined first. Then the repairer starts the third repairing therapy in the TOP: 'exchange machining sequence'. This therapy starts the repairing from the process blocked, then it exchange the process with the next process in the process plan until forward verification returns OK. This repairing therapy works in this case. The resulted plan is listed in *Figure 12*.

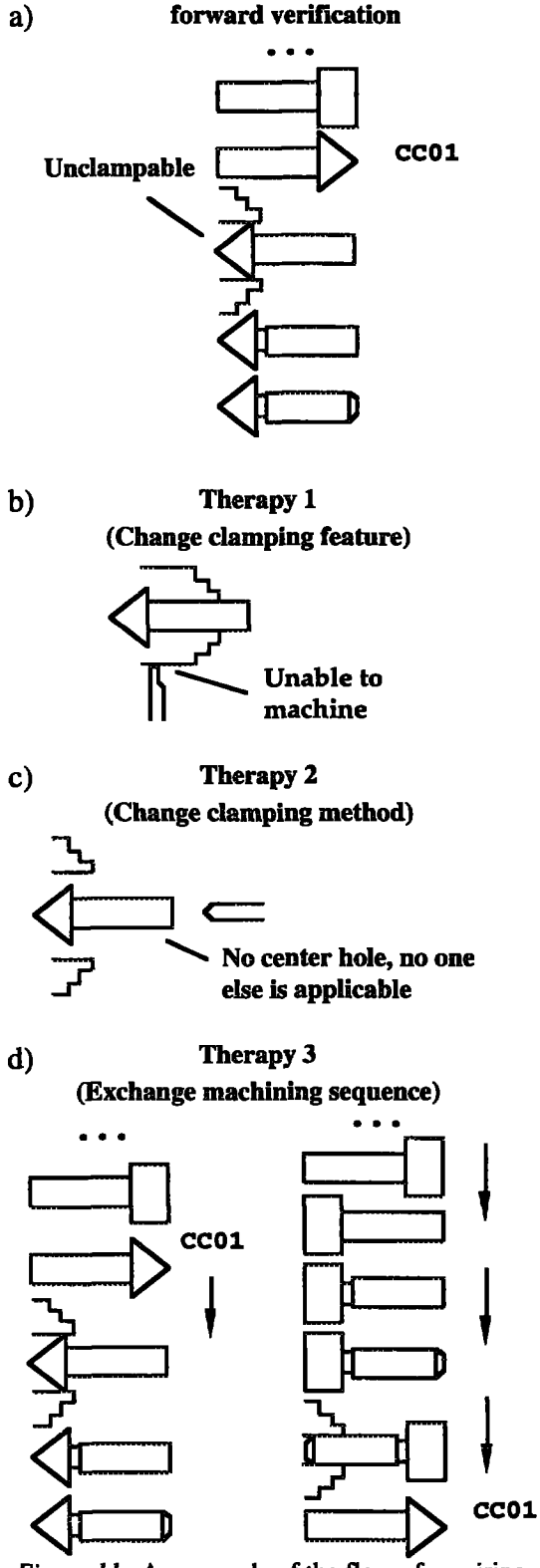

Figure 11. An example of the flow of repairing

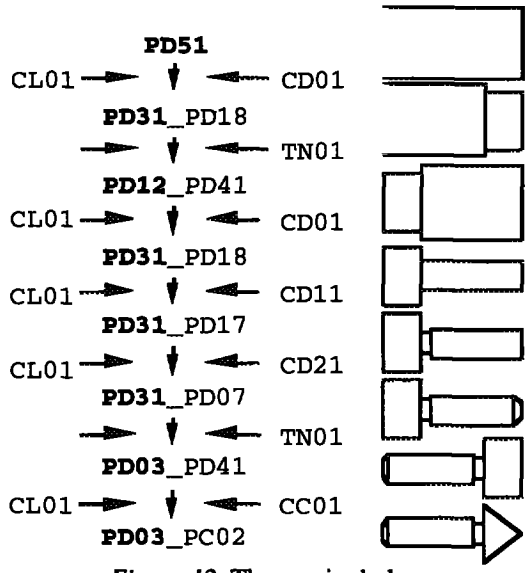

Figure 12. The repaired plan

Conclusions

The nature of process planning is case based. Process planning is a very promising application area for case-based reasoning. However there is sparse study in this area. PROCASE is a system developed by the authors in an attempt to apply the theory of case-based reasoning in manufacturing process planning. This paper introduced the case adaptation in PROCASE.

It should be pointed out that the case adaptation algorithm developed in PROCASE was largely inspired by Kristian Hammond's work in case-based planning (CHEF) (Hammond, 1986). Certainly, most of the issues addressed in CHEF are applicable to other planning systems. Although the implementation of the theory is very application specific, the readers might find many parallelisms between PROCASE and CHEF.

The result of the testing system is satisfactory. However, in order to implement a real world case-based process planning system, a much bigger system is expected to achieve a sophisticated system.

Appendix

The following (*Figure 13*) are some part primitives and their codes introduced in the paper.

The following are some cutting processes and a clamping operation as well as their operation codes introduced in this paper.

1. CD21 -> cut a chamfer (*Figure 2, 8, 10, 12*).
2. CD11 -> cut a groove (*Figure 2, 8, 10, 12*).
3. CD01 -> peripheral cut (turning) (*Figure 3, 8, 10, 12*).
4. CC01 -> cut a cone shape (*Figure 9, 10, 11, 12*).
5. CL01 -> clamping by chuck, from left (*Figure 8, 9, 10, 12*).

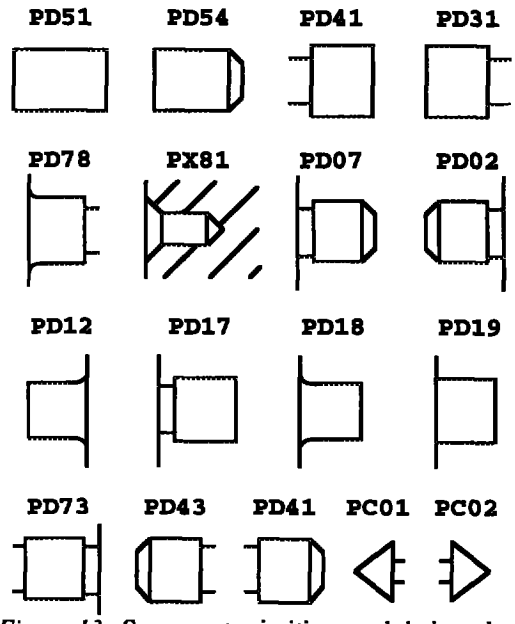

Figure 13. Some part primitives and their codes introduced in this paper.

References

Atling, L., and H. Zhang, 1989. "Computer Aided Process Planning: the State-of-the-art," *International Journal of Production Research*, 27(4): 553-585.

Chang, T.C. 1990. *Expert Process Planning for Manufacturing*. Addison-Wesley

Goodman, M., 1989. "CBR in battle planning", *Proceedings of the DARPA Workshop on Case-based Reasoning*, Pensocola Beach, FL, : 363-373

Hammond, K.J., 1986. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory*, Ph.D. diss., Dept. of Computer Science, Yale University

Pu, P. and M. Reschberger, 1991. "Case-based Assembly Planning", *Proceedings of Case-based Reasoning workshop*, Washington, D.C., : 245-254

Riesbeck, Christopher K. and Roger C. Schank, 1989. *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates Inc.,

Sycara, K. and D. Navinchandra, 1991. "Influences: A Thematic Abstraction for Creative Use of Multiple Cases", *Proceedings of Case-based Reasoning workshop*, Washington, D.C., : 133-144

Yang, Hao, & Wen F. Lu, 1993. "PROCASE: A Prototype of Intelligent Case-based Process Planning System with Simulation Environment", *Computers in Engineer 1993 (ASME)*, San Diego, CA, : 571-577

Yang, Hao, Wen F. Lu and Alan C. Lin, 1992. "A Frame Work for Using Case-based Reasoning in Automated Process Planning", *Proc. of Winter Annual Meeting, ASME*, PED-Vol. 95, : 101-114