

An Agent-Based CAD Framework for Pay-for-Use Design

Xiaobo Peng^{1a} and Derek Yip-Hoi^{2b}

¹*College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen, China*

²*Department of Engineering Technology, Western Washington University, Bellingham, WA, USA*

^a*pengxb@szu.edu.cn*, ^b*Derek.Yip-Hoi@wwu.edu*

Abstract.

An agent-based framework was proposed for pay-for-use design applications. Under the framework 3D CAD models become commodities independent of the CAD systems in which they are created. In this way a user can easily shop around the Internet for the 3D models without the cost of purchasing or maintaining CAD software. In the paper the architecture of the framework was provided first. Secondly a procedural modeling algorithm which enables the framework to create the neural model according to functional and/or structural inputs specified by the user was presented. Then a model mapping method was advanced to translate the neural model to a set of CAD modeling API's. Finally two practical examples were illustrated, which showed the feasibility of the methodology presented in the paper.

Keywords: CAD, Pay-for-Use, agent, design framework

1 Introduction

CAD system evolution is currently well within the 3rd generation. Today feature-based, parametric modelers are widely available[1]. These provide CAD users with the greatest flexibility in creating and modifying models as well as supporting the part family design. Distributed collaborative design has also received much attention recently in CAD field[2,3], and grid computing is incorporated into product design[4]. At the same time, more sophisticated programming APIs to CAD systems and Integrated KBE/CAD systems[5] are making possible customization of these systems so that modeling activities can be driven by functional inputs rather than generic modeling commands. These are opening the door to the integration of Pay-for-Use techniques with CAD systems.

The proposed Pay-for-Use CAD methodology adds to the foundation for distributed CAD services, a key feature for next-generation design tools[6]. If 3D models are placed in the Pay-for-Use framework as a kind of commodity independent of the CAD systems in which they are created, CAD system

owners/developers can offer their experiences in 3D modeling on the Internet, while a potential customer can easily shop around the Internet for the best purchasing or leasing deal for the 3D model without the cost of purchasing and maintaining CAD software. In many cases, the real investment in a CAD system goes way beyond the cost of the modeler to the investment in creating 3D models. Minimizing this investment is difficult because of the complexity of these environments.

The objectives of the paper are as follows: To develop and validate a new design framework that promotes CAD modeling, utilizing distributed CAD resources offered on the Internet as Pay-for-Use services; To develop new methodologies where gaps exist in the current technologies that will make up the framework. Key amongst these are procedural modeling of 3D CAD models, and model mapping which translates neural model to different CAD API's.

2 Architecture of the Framework

To demonstrate how the Pay-for-Use design is accomplished, the following work flow example is used:

A user wishes to create a spur gear. However, the user is not a CAD expert and the design is required in short time. The user decides to try Pay-for-Use CAD service for modeling components on the Internet:

Step 1: The user searches the Internet for Pay-for-Use spur gear design agents(SA) and downloads one from a CAD application server to the local workstation.(DA)

Step 2: Using the design agent, the user inputs functional parameters about the gear like module, number of teeth and so on through the agent's interface.

Step 3: The design agent creates a procedural model of the gear by the inputs, and maps to CAD system dependent modeling instruction sets(MI), then broadcasts request for services(RS) with MI's across the Internet to CAD service agents provided by CAD service servers and willing to create the spur gear model. These service agents may run any of the major CAD applications.

Step4: The service agents return lightweight models and the design agent shows them to the user, together with the deal options and prices.

Step5: The user selects one model and finishes the payment using Alipay or something like.

Step6: The selected service agent transfers the model data to the design agent so that the user can download or use it.

In this way the user purchases or leases a CAD model on the internet without owning any CAD system. The CAD system owner or developer offers experience in modeling a component as a paid service on the Internet.

Besides design agents, the framework can also include manufacturing agents like prototyping agents, so that the user can design the part and verify the design rapidly. Fig.1 presents a high-level view of the Pay-for-Use CAD paradigm.

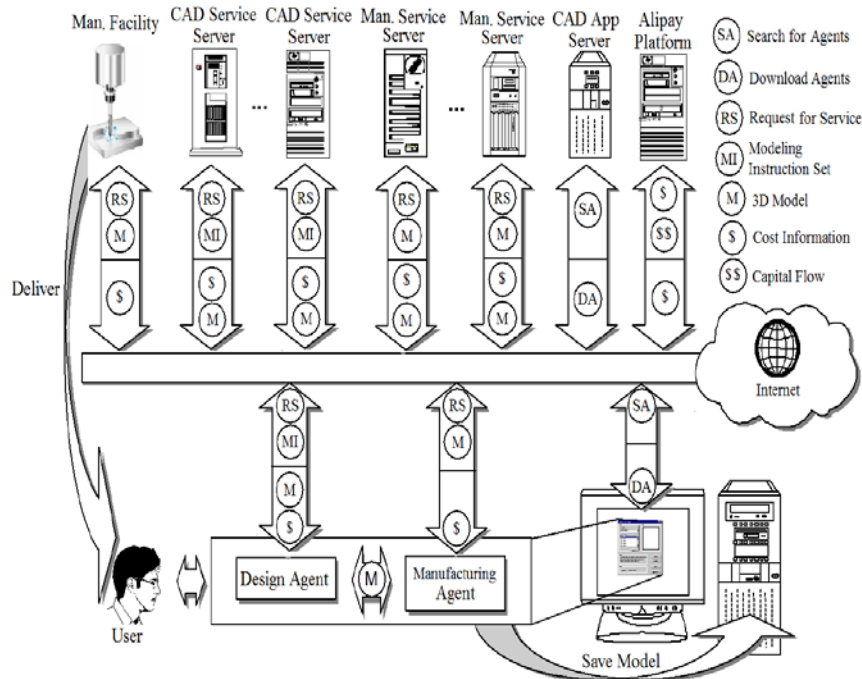


Fig.1 The architecture of the Pay-for-Use design framework

3 Procedural Modeling

From the work flow it can be seen that the key of the framework are the neural procedural modeling of 3D models, and model mapping which translates neural models to different CAD API's.

3.1 Expression of the Procedural Model

According to Group Technology, for over 75% of the parts on the market each can be classified into a group and developed by modifying the group pattern(GP). The framework establishes a group library(GL) which covers the common part groups on the market. The GL is organized in hierarchy by group functions and structures, so user can easily find the group where he wants to develop his part.

Each group in the GL has a GP. GP is in fact an integrated part containing almost all structures that the parts in the group may have. Fig.2(a) shows the GP of a shaft group used for gear reducers. It has seven shaft segments and any shaft with no more then seven segments can be developed by modifying it. GP mainly records three kinds of information: a construction tree(CT), a set of functional and/or structural inputs(FI), and a dependence matrix(DM).

CT is like a CSG/B-rep hybrid structure. Fig.2(b) is the CT of the left two segments of the shaft in Fig.2(a). Circular nodes are leaf nodes(LN) and square nodes are intermediate nodes(IN). IN is obtained by the BOOL operation between its child nodes. For example, *N1* is got when chamfer *C1* is subtracted from the

original shaft segment $S1$. LN is also called Atomic Body Node, representing a basic geometric body which can't or is unnecessary to be separated further in the sense of modeling.

The data structures of CT nodes are shown in Fig.3. IN node decides the BOOL operation type by *Op Type* to create itself. LN records the type and geometric data of the basic body. Different basic body types have different feature data formats.

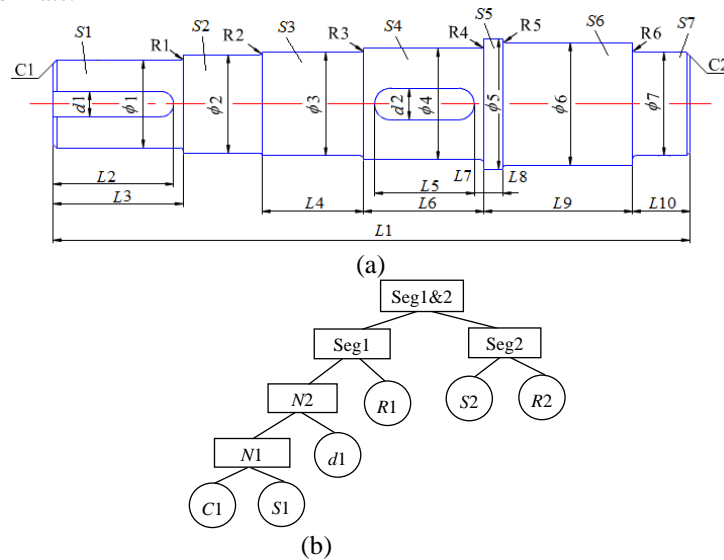


Fig. 2 Group pattern and its construction tree

There are three layers of dimensions in the framework. The top layer is the model dimensions(D_m), i.e. the functional and/or structural dimensions like those in Fig.2(a). They are input by the user. The middle layer includes the position dimensions(D_p) and the shape dimensions(D_s) of each basic body(*Pos Dim* and *Shape Dim* fields in LN structure). Position dimensions are used to locate the body while shape dimensions to describe the body shape. Body shape can be expressed as a bounding box by D_s if the body has an irregular shape. The bottom layer is the feature dimensions(D_f) of geometric elements needed to create the body(*Geom Data* field in LN structure).

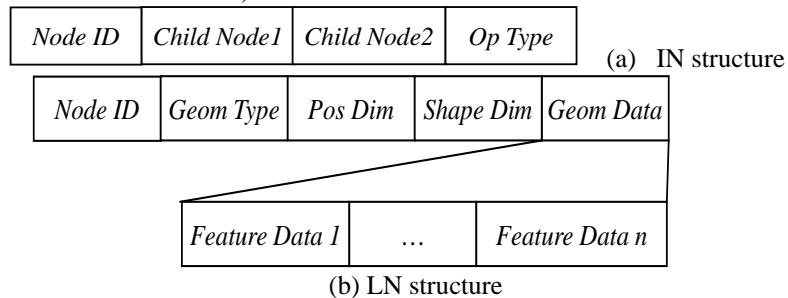


Fig.3 Data structure of CT nodes

3.2 Model Maintenance

Because GP has the maximal structures of the group, the user may want to delete some he thinks unnecessary. He can do this by setting some D_m to zero. For example, if he set $L3$ in Fig.2(a) to zero, shaft segment $S1$ will disappear in the design result. In this case, the framework needs to update the CT and recalculate the model so that a new part is developed from the GP.

For each LN or basic body B_i , its dimension set

$$D_i = [D_{pi}, D_{si}, D_{fi}]^T = G_i(D_m^T)$$

(1)

G_i is the mapping function from model dimension space to basic body dimension space.

If $\forall d \in D_{si} \& d = 0$, then B_i degenerates and needs to be deleted from the CT.

When one basic body is deleted, it may cause some other bodies to be deleted as well. For example $C1$, $K1$ and $F1$ should be deleted too if $S1$ in Fig.2 is deleted. Before recalculating the model all the bodies to delete need to be found out.

If GP has n basic bodies, it maintains a $n \times n$ dependence matrix A which records the adjacent relations between the basic bodies. $a_{ij} \in A (i, j \in [1, n])$

stands for the adjacent relation between basic body B_i to B_j . a_{ij} can be one of the following values:

- 1) $a_{ij} = 0$, if $B_i \cap B_j = \Phi \parallel i = j$. B_i and B_j are not adjacent.
- 2) $a_{ij} = 1$, if $B_i \cap B_j \neq \Phi$. B_i and B_j are adjacent.
- 3) $a_{ij} = 2$, if $B_i \ni B_j$. B_i is dependent on B_j . Relation \ni is directional.

If B_i is dependent on B_j , then B_i should be deleted as well when B_j is deleted.

In table 1 is part($S1$, $S2$ and $S3$) of the DM of the shaft in Fig.2(a). Based on the CT and the DM, model maintenance algorithm is designed as follows:

Algorithm ModelMt

Input: CT, D_m , G , A

Output: Updated CT

Table 1 DM of the shaft in Fig.2(a)

	$C1$	$K1$	$F1$	$F2$	$F3$	$S1$	$S2$	$S3$
$C1$	0	1	0	0	0	2	0	0
$K1$	1	0	0	0	0	2	0	0
$F1$	0	0	0	0	0	2	0	0
$F2$	0	0	0	0	0	0	2	0
$F3$	0	0	0	0	0	0	0	2
$S1$	0	0	0	0	0	0	1	0
$S2$	0	0	0	0	0	1	0	1
$S3$	0	0	0	0	0	0	1	0

Step1: Calculate $D = [D_1, \dots, D_n]^T$ by Eq.1;
Step2: Search D , if $\forall d \in D_{si} \ \& \ d = 0$, put B_i into degenerated body set T ;
Step3: Set $i=1$;
Step4: Get t_i from T ; if t_i is NULL, go to Step8;
Step5: Get the column number j of t_i in A ;
Step6: For $(k=1 \dots n)$, if $a_{kj}=2$, add B_k to T as its tail;
Step7: $i=i+1$, go to Step4;
Step8: Traverse CT by Depth-First-Search method:
 If node $N \in LN \ \& \ N \in T$, delete N from CT;
 If node $N \in IN$,
 If N has no child, delete N from CT;
 If N has only one child, replace N by its child;
Step9: return CT.

3.3 Model Mapping

The construction tree actually records the 3D modeling process of a part. However, it can't be understood by the CAD systems outside the design agent. The design agent needs to map it to the service agents on different CAD systems. Model mapping creates modeling API sets which can be called by CAD systems to create the 3D model represented by the CT.

There are two kinds of model mapping: body mapping and operation mapping.

Body mapping occurs at each leaf node, creating basic body according to its *Geom Type* and *Geom Data*. For most *Geom Types*, a basic body can be created by an operation of extruding, revolving, sweeping or lofting. Body mapping can be extended to adapt to very complex shapes by extending the Feature Data field structure to B-rep structure.

Operation mapping occurs at each intermediate node. It merges two child nodes by the BOOL operation recorded in *Op Type*.

The model mapping is entirely the correspondence between the geometric types of the nodes and modeling API's. The details are beyond the paper and not discussed here.

4 Experimental Results

The framework has been implemented using C++ under the Window environment. The CAD service servers and design agent servers are distributed over a local area network. A design agent is downloaded to the user's computer which runs on an Intel Core 3.4GHz dual processor platform without any CAD system installed. Service agents run on some different computers, being developed on Acis or Solidworks. The payment process has not been practically realized but been simulated.

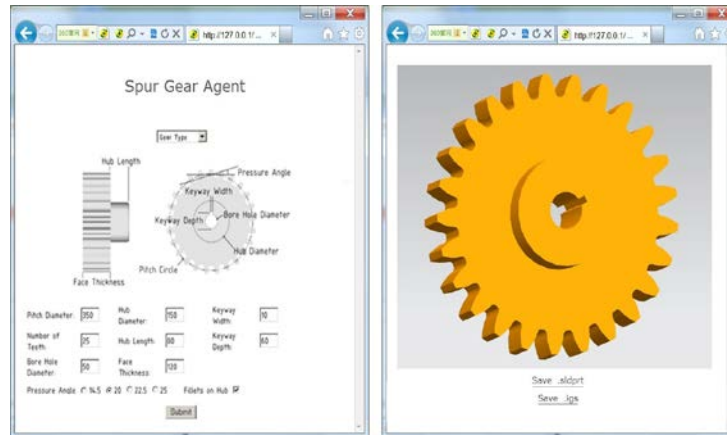


Fig. 4 Spur gear design

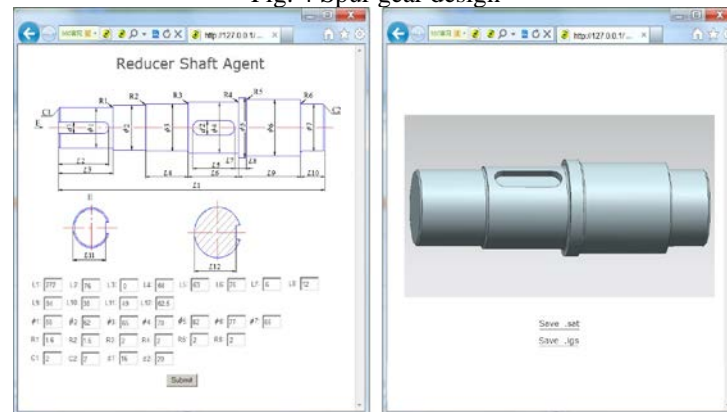


Fig. 5 Shaft design

Fig.4 is the spur gear design mentioned in Section 2. The user inputs the functional parameters of the gear, pitch diameter of 350mm and 25 teeth for instance through the design agent's interface to create the gear. A Solidworks based service agent is selected and the result model can be saved as a .sldprt file. For the example in Fig.2, the user sets the dimension $L1$ to 272 and $L3$ to 0. $S1$ and $S2$ are removed and the CT is updated accordingly. An Acis based service agent is selected and the result model can be saved as a .sat file, shown in Fig.5.

5 Conclusions

An agent-based framework was presented for Pay-for-Use design. Under the framework a design engineer without CAD background can create the 3D model of his design through the Internet by functional and/or structural inputs. The proposed framework could cut the user's cost of purchasing, installing, using and maintaining a CAD software, meanwhile the CAD system developer or owner that invest in customization of CAD systems can, if they desire, recoup part of this

investment cost. To realize the framework, the procedural modeling algorithm was presented to enable the framework to create neural model according to the functional or structural inputs specified by the user. And the model mapping method was advanced to enrich the framework an ability to translate the neural model to CAD-system-dependant models. Further work, integrating KBE with the procedural modeling for example, is needed so that the framework can create a totally new part with very complex structures.

Acknowledgements

This work was financially supported by Shenzhen Basic Research Program (JCYJ20120613162656264).

References

- [1] Shah, J. J. and M. Mantyla: *Parametric and Feature-based CAD/CAM: Concepts, Techniques, and Applications* (John Wiley & Sons, Inc, USA 1995).
- [2] Cutkosky, M.R., Tenenbaum, J.M., Glicksman, J.: Madefast: Collaborative Engineering over the Internet, *Communications of the ACM*, Vol. 39, No.9(1996), p. 78-87.
- [3] Shapirshteyn, Y., Foster, C.V., John, J.E. et al: Building Internet-Based Virtual Environments Collaborative Design, Co-Designing 2000 Conf, Coventry Univ., UK, Sept 11-13, 2000, p. 117-122.
- [4] Shi, H., Tang, M., Chou, S., Dong, J.: A grid computing integrated distributed CAD system, *CSCWD 2004-8th International Conference on Computer Supported Cooperative Work in Design - Proceedings*, 2004, pp 673-678.
- [5] Zhang, F.; Xue, D.: Distributed Database and Knowledge Base Modeling For Concurrent Design, *Computer Aided Design*, Vol.34, No.1(2002): p.27-40.
- [6] Szykman, S., Sriram, R.D., Regli, W.C.: The Role of Knowledge in Next-generation Product Development Systems, *ASME Journal of Computation and Information Science in Engineering*, Vol. 1, No. 1, 2001, pp 2-13.