A Component-based Software Development and Execution Framework for CAx Applications

N. Matsuki, H. Tokunaga, H. Sawada

Digitalization of the manufacturing process and technologies is regarded as the key to increased competitive ability. The MZ-Platform infrastructure is a component-based software development framework, designed for supporting enterprises to enhance digitalized technologies using software tools and CAx components in a self-innovative way. In the paper we show the algorithm, system architecture, and a CAx application example on MZ-Platform. We also propose a new parametric data structure based on MZ-Platform.

Keywords: component-based development, CAD, parametric modeling.

1 Introduction

We have developed a new component-based software development framework called MZ-Platform. This research, financed by METI (Ministry of Economy Trade and Industry), started in 2001 and is planned as a 5 years research project [1]. The aim of this project is to develop a new tool for the manufacturing industry, particularly for small and medium size enterprises, that enables engineers themselves to develop new CAx (i.e., CAD, CAM, CAE, CAT as a whole) programs and strengthen their competitiveness.

Information technology (IT) tools, typified by CAD, have become indispensable for manufacturing enterprises in order to enhance their productivity. However, the costs of purchasing IT tools, maintaining the systems, and training engineers and operators for tools, are a heavy burden for small and medium-size enterprises. The burden becomes much heavier, if they try to customize tools to make the most of the ability.

According to our survey, current CAD tools have abundant general functions, but are too general for small and medium-size enterprises, because CAD tools are designed to satisfy the diversified requirements of the aerospace, shipbuilding, and automobile industries. Particularly die and metal mold design for mechanical parts, in the design of which small and medium-size enterprises play a key role in Japan, requires specialized techniques and skills that are difficult to replace by current CAD tools, which mainly focus on supporting tasks in the early stage of product design.

We conclude that easy-to-use software development tools, generous supplies of software parts (components) and semi-finished CAD (or CAx) tools will encourage small and medium-size enterprises. MZ-Platform is designed for this purpose and we have developed basic software modules of the MZ-Platform: Component-Bus, Application-Builder, XML-Component-Transmutation, and Remote-Component-Collaboration. We have also developed an application called MZ-Checker, which can verify the quality of 3-dimensional geometry data, such as the distance or the break angle of two adjacent free-form surfaces. We consider that the MZ-Checker development has attested the capability of MZ-Platform as a development tool. In this paper we show the algorithm, the system architecture of MZ-Platform, and a new parametric modeling scheme that overcomes the current parametric data issues.

2 Background and related work

The idea of component-based development (CBD) is well known in software engineering [2]. In CBD, software systems are built by assembling components already developed and prepared for integration. Component-based software engineering (CBSE) has become a subdiscipline of software engineering, and much research has been done, but mainly on software architecture and software architecture description languages [3]. On the other hand, commercial CBD tools, such as Visual Basic, .NET framework of Microsoft Corporation [4], are quite powerful, but they do not have special features directly related to CAx application development. In the development of the CAD system, Dassault Systems, the major CAD vendor, has adopted a component-based development concept called the OM component model [2] for its product CATIA. The OM component model will be utilized when we enhance the CATIA functions. Though they seem to have ample functions, we do not use them because our component-framework is not based on CATIA.

In order to design a tool to make CAx systems, flexibility of component connection is a key concept, because CAD data represented in object-oriented language, such as Java, becomes a component.

3 MZ-Platform: a component-based software development framework

MZ-Platform is a fully event-driven component development and execution framework, based on Java and JavaBeans component model technology. The architecture is shown in Figure 1. When users want to make a CAx software tool through the use of MZ-Platform, the first thing they have to do is to consult the Component Library (Fig. 1). Many components have been prepared, such as GUI components, graphic library components, database access components and components for remote access in Component Library. As a result, users may find components that are commonly used in CAx tools in the Component Library.

Some functions specific to user-demands have to be designed and developed in the conventional programming environment, and these functions are to be made in the form of components. After storing these components to Compo-

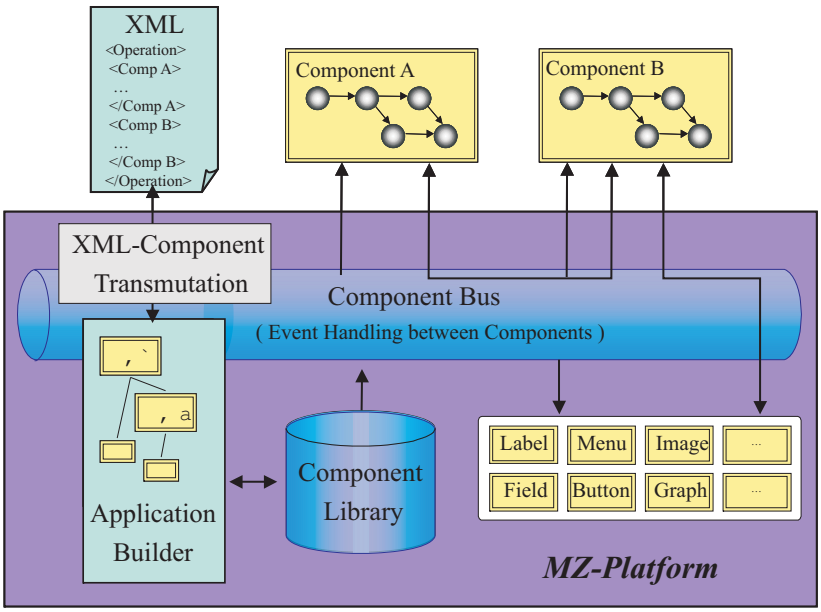


Fig. 1: Architecture of MZ-Platform

nent Library, users can develop a software tool using Component Builder loading components from the Library, and "wiring" those components. Throughout the process, users can check the actions of components and assembled components (applications) anytime they want through the functions of Application Builder. In this way, we suppose that nonspecialists in software can develop a CAx tool very quickly.

3.1 Component bus

The core of MZ-Platform is the event-handling module called Component Bus. Component Bus controls all events from components using the connection objects. When an event from component A to component B has occurred, Component Bus receives this event and, if component B is ready to activate, invokes component B and sends the event to component B. To establish a connection between components, Component Bus uses the reflection of Java language, which enables all the methods that the component has to be seen through a reflection interface.

Connection of objects is designed to propagate events. The concept of a connector object plays an important role in the study of Architectural Definition Language (ADL), but few currently available component frameworks have such a kind of intermediate object. Almost all component frameworks connect components directly, because of simplicity. We employ connector objects because they allow us abundant flexibility to change the component assembly in a dynamic way, namely, to change the connection while the program is running.

It is frequently seen in CAx applications that, according to the change of real number of geometry dimensions, the algorithm (and program) is to be switched. The problem is that there is no absolute dimension value (the threshold value) at which it should switch, but it depends on the environment like the accuracy of computer hardware. For example, at the corner of break lines we usually define a semicircle to smooth them. But if the break angle becomes near 180 degrees, we can no longer define a semicircle at the corner because the radius goes to infinity. Usually, the switch of the algorithm is coded within one component, but, it is difficult to switch correctly as the threshold value depends on the environment. Component Bus can handle this switching much more easily because the switch of program can be coded as the assembly of components, and the threshold value can easily be accommodated.

3.2 Component Library

We developed GUI (Graphical User Interface) components, 3-dimensional geometry visualization components based on Java3D, and other components helpful for building CAx applications, such as a component that calculates the distance of geometry entities. These components are serializable JavaBeans and are stored in JAR format files. Component Bus loads components from Component Library. If a new component is developed, it has to be stored in Component Library to be utilized as a part of MZ-Platform.

The abundance of Component Library is one of the key issues for MZ-Platform to be an easy-to-use tool and, at the same time, assistance on how to use the components is also important. As this project is planned to continue until 2005, we try to enrich Component Library continuously.

3.3 Application Builder

Application Builder is the user interface module to define the component assembly. An example of a user defined component assembly screen is shown in Fig. 2. Each rectangle of the screen represents a component, and the lines between the rectangles represent the method invocations, the name of which is shown above the line.

Once the connection between components is defined, Component Bus invokes the components and creates a connector object simultaneously.

Users can select several different modes of Application Builder using the buttons at the bottom of the screen, which are application execution mode, screen layout mode and

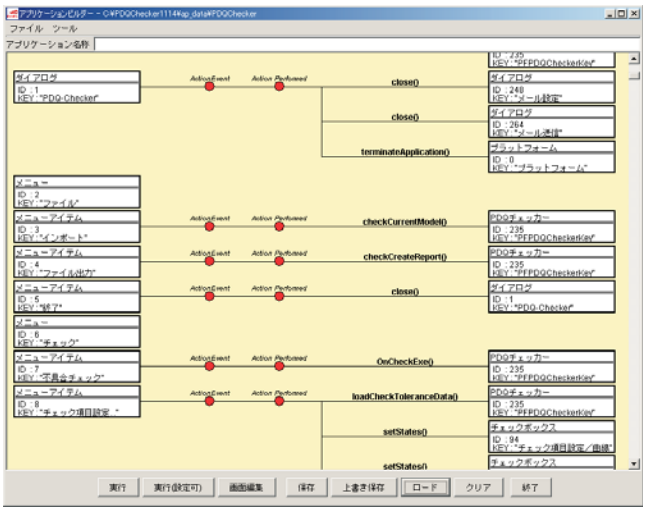


Fig. 2: Application Builder of MZ-Platform

load/save components. During the execution of Application Builder, users can alter the properties of components like colors of the screen background, message string of dialog and similar attributes of components. Also, the usability of the screen layout of an application can be verified while the software is under construction, because the components on Application Builder are all activated.

Application Builder supports a compound component. Several components are assembled into a line component; namely, a hierarchical component can also be defined in the same screen.

3.4 XML-component transmutation

Component integration information defined by Application Builder can be stored in the XML (eXtensible Markup Language) format file. In the XML file, comments for component and argument explanation are also added. Fig. 3 (left) shows an example of component assembly information and the corresponding application.

In this example, the XML file has more than four thousand statement lines. This XML file contains the program to make the surface shown in Fig. 3 (right) and the geometry data of four special curves to define the surface. This shows that not only a CAx function such as "create a surface from four bounded curves", but also surface data in parametric representation can be stored and shared in the XML file. XML-Component Transmutation module can read this XML file as a component assembly information, and pass this information to Component Bus as if it was defined by Application Builder.

Moreover, if an application receives a message of this XML data, XML-Component Transmutation translates the message and load components interactively. Using this function, for example, the application screen layout can be dynamically altered depending on the message outside the application.

3.5 Remote component collaboration

Recently, it has become common that several enterprises are involved in the design and manufacture of a new product. To support these processes, collaboration of distributed software is one of key features of CAx tools. MZ-Platform has the collaboration capabilities to invoke a remote component and send a component from one MZ-Platform to the other. Remote component methods (or services) can be found using UDDI (Universal Description, Discovery and Integration) protocols [5]. Messages are sent by SOAP (Simple Object Access Protocol) format, and they can reach the other MZ-Platform site if it is inside the firewall.

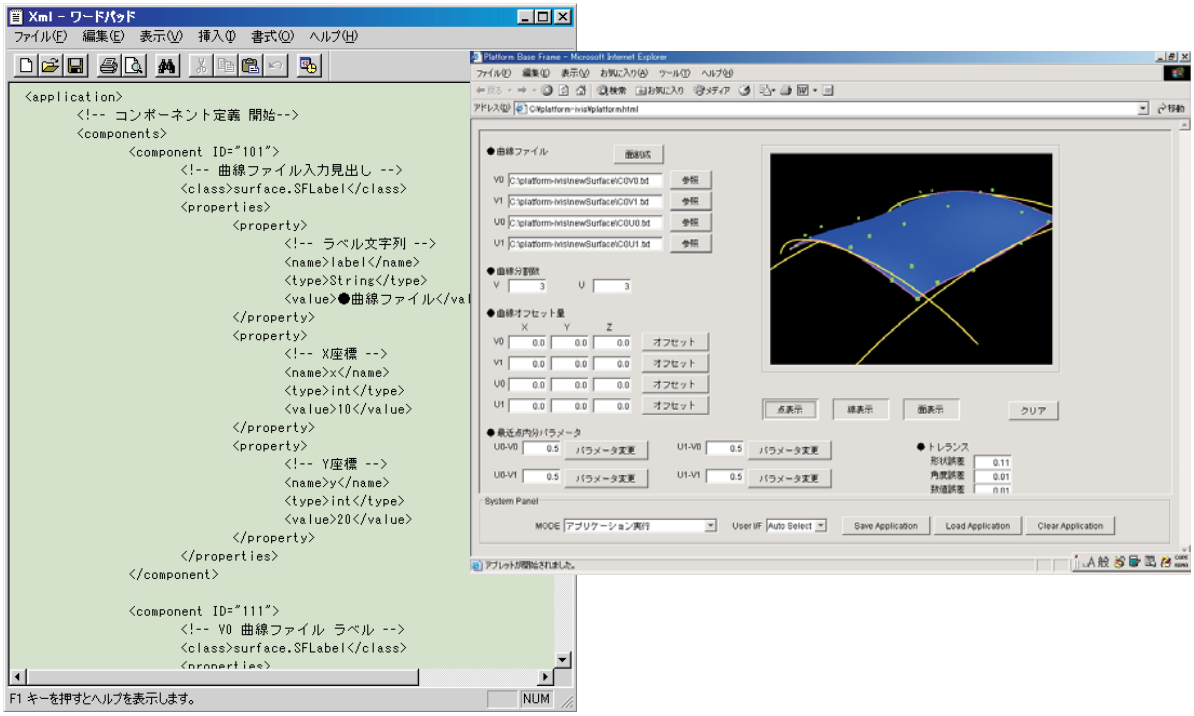


Fig. 3: Component Assembly information stored in XML format and the corresponding application

There are several major protocols: CORBA (Common Object Request Broker Architecture) [6] defines RPC (Remote Procedure Call)-like remote object invocation protocols; Java has RMI (Remote Method Invocation) to use the service of a remote Java object. The feature of Remote Component Collaboration is that users can assemble remote components in the same way as local components in their own Component Library. Component Bus creates an XML message if an event for a remote component occurs. A process for remote communication called Broker creates a SOAP message and sends it to the destination Broker. During this procedure, the UDDI server provides destination information to Broker, as shown in Fig. 4.

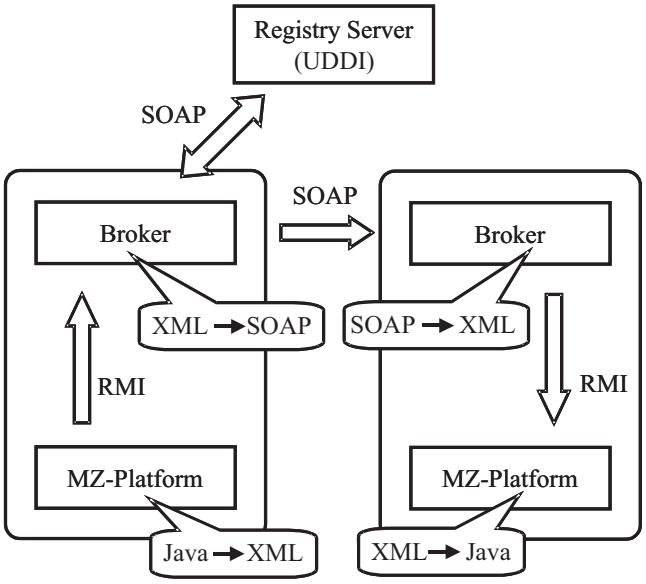


Fig. 4: Architecture of remote component

4 Applications

To prove the effectiveness and usability of MZ-Platform, we have developed an application called MZ-Checker (Fig. 5).

The main function of MZ-Checker is to verify the quality of 3-dimensional CAD data: to calculate the angles between a selected axis-vector and surface normal vectors that are finely sampled, and to display them on the screen as color-coded surfaces for the purpose of metal-mold model checking. We have proved that MZ-Platform is capable of developing commercial level programs. We believe that 6 months of development is very short compared to the same level application development. Moreover, the easiness of adding a new function is shown by the example, which was developed and tested by MZ-Checker in just two days.

As a CAx tool, MZ-Checker has many features. SASIG (Strategic Automotive product data Standards Industry Group) has announced that CAD data of inferior quality is recognized as a major cause of rework and cost by many automobile industry organizations [7]. JAMA (Japan Automobile Manufacturing Association) [8] and SASIG have published a guideline aimed at preventing inferior quality CAD data from being created, called the PDQ (Product Data Quality) guideline. MZ-Checker is the only tool that has full conformity with the PDQ guideline. Furthermore, MZ-Checker can load various types of CAD data, such as STEP (AP203, AP214), IGES; it is also used as a viewer of CAD data. As a part of this research project, we started distribution of MZ-Checker for small and medium enterprises.

5 Component representation of parametric modeling

MZ-Platform provides an environment for creating geometry modeling applications as a completely event-driven component assembly. The XML-Component Transmutation module can store component assembly in XML format. This shows that MZ-Platform can be regarded as a parametric modeling framework.

History-based parametric data in the current CAD tool is a collection of CAD operation (or command) names, argument geometry data, input parameters to these operations, and the assembly structure of operations [9]. History-based parametric modeling in the current commercial CAD tool has an imperfection that its data has no warranty to be re-executable,

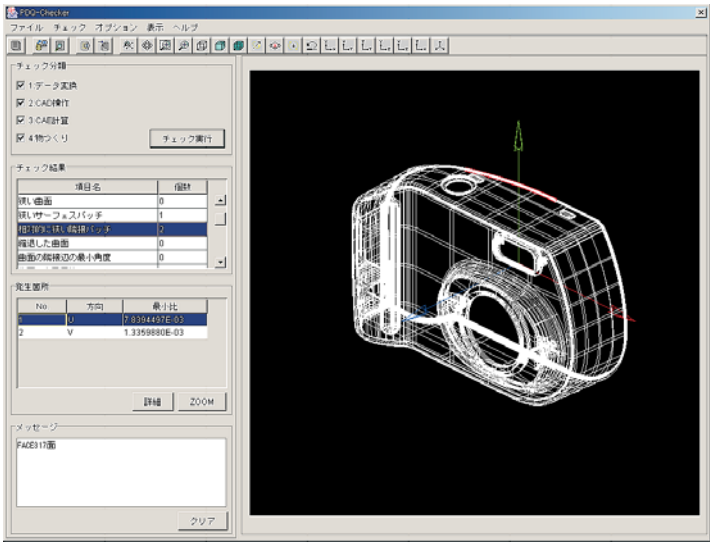
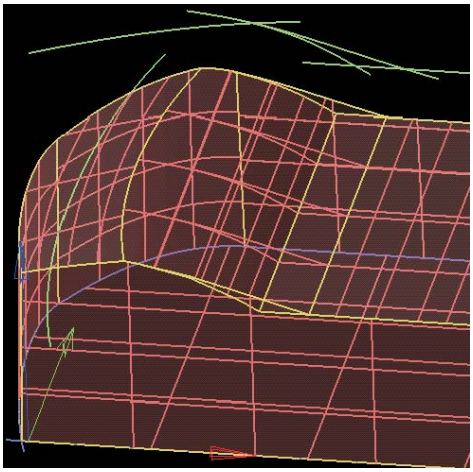
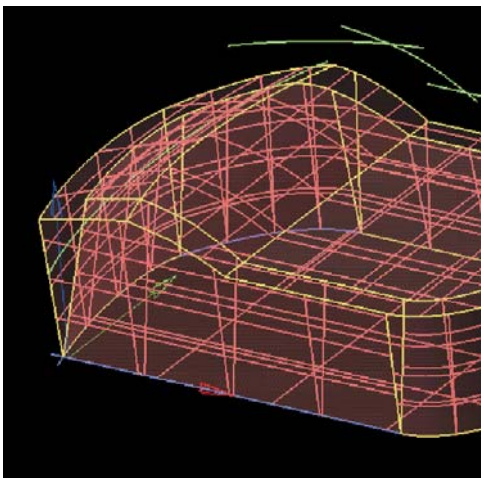


Fig. 5: MZ-Checker



(a) curve replacement example



(b) sweep angle modification example

Fig. 6: Parametric representation of a model and result of modification

if there is a slight difference between the CAD tools it defined and the CAD tool it runs. This imperfection arises from the nature of parametric data, which is deeply involved in the algorithm of geometric modeling or its programming codes. That is, if the name of an operation in parametric data is modified, it is the same as the parametric data is being modified.

When the new CAD version is released, a part of the program codes is modified or upgraded, and the operation that is defined by these modified program codes will behave differently than it used to. This is the essential problem of parametric data, and we call this a problem of the "parametric data not being persistent".

As shown above, the component assembly information written in XML format can be parametric data. If the component is designed to be a suitable size and some tacit factors that cause the inconsistency are controlled from outside the component, parametric data represented in XML format becomes persistent.

This is an ongoing research item and we will show the results in future papers. The parametric data example we have developed is shown Figure 6. In this example, a curve in the model can be replaced by an arbitrary curve. In Fig. 6(a), the 2nd of the three curves is replaced by a new curve. Fig. 6(b) shows that the sweep angle is changed from 2 degrees to 10 degrees.

6 Conclusions and future work

We have developed a component-based software development framework MZ-Platform. It enables engineers in small and medium-size enterprises to develop programs specific to their needs. Basic modules of MZ-Platform have been designed and developed. There is plenty of scope for improvement. For example, the user-interface of Application Builder can be simpler and more friendly to nonspecialists in programming. As mentioned above, we will to improve and enrich the components from now on.

Concerning the parametric modeling problem, we are planning to design a more complicated CAD model with fillet surfaces. It will prove that the proposed parametric data in XML format could be a candidate for parametric data of the next generation.

References

[1] Matsuki N.: "A National R&D Project Plan to Establish Manufacturing Information Infrastructure in Japan". Proc. of ITIT Symposium on Development Manufacturing Technology Infrastructure, AIST, (2001), p. 6–8.

[2] Crnkovic I., Larsson M.: *Building Reliable Component-Based Software Systems*. MA (USA): Attech House 2002, ISBN 1-58053-327-2.

[3] Medvidovic N., Taylor R. N.: "A Classification and Comparison Framework for Software Architecture Description Languages". *IEEE Trans. On Software Engineering*, Vol **26** (Jan. 2000), No. 1.

[4] http://www.microsoft.com/net/basics/whatisasp

[5] http://www.uddi.org

[6] http://www.omg.org

[7] http://www.sasig-pdq.org

[8] http://www.jama.or.jp

[9] Anderl R., Mendgen R.: "Parametric design and its impact on solid modeling applications". ACM Proc. 3rd Symposium of Solid Modeling, (1995), 1.

Mr. Norio Matsuki
Dr. Hitoshi Tokunaga
Dr. Hiroyuki Sawada

Digital Manufacturing Research Center (DMRC)

National Institute of Advanced Industrial Science and Technology (AIST)
1-2, Namiki, Tsukuba
Ibaraki, Japan