EMAF: An Enterprise Manufacturing Application Framework Integrated Environment

Sai Peck Lee, Siew Khim Thin, Hong Song Liu
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur
email: saipeck@fsktm.um.edu.my

Abstract

A comprehensive overview of the integrated development environment (IDE) of an enterprise application framework on manufacturing application domain which is currently being developed is presented. This IDE is developed to support in the design, development, debugging and deployment of manufacturing application systems. This is achieved through the use of integrated reusable components that are provided by the enterprise manufacturing application framework (called EMAF). The IDE provides most project management functions with a complete support of edit, compile and debug cycle. It also allows development of interface specifications for manufacturing application systems. The manufacturing applications dealt within this project include production planning, scheduling, inventory control, process control and inspection. Via the IDE, a new manufacturing application can be built from a set of integrated reusable components of the EMAF and then adapted to suit specific needs of the reuser. The details of the internal structure of the IDE and its graphical user interface (GUI) are also discussed.

Keywords: Integrated Development Environment, Enterprise Application Framework, Component-based Development, Manufacturing Domain, Object-oriented Technology.

1. Introduction

Almost all organizations need some comprehensive software to support their organizational functions. In the business world, software products and services are often information-intensive and must be developed or brought to the market fast in order for the organizations to stay competitive. Today, software has been increasingly recognized as a vital corporate resource, which has to be developed, integrated and disseminated across the whole organization within a short time span to serve the users’ needs. It is a fact that software products will become larger in size and more complex, but the expected development cycle of a software product will become shorter rather than longer. This situation has caused the emergence of a number of concepts, such as Object-Oriented Technology (OOT), systematic software reuse, component-based development and application framework [1, 2, 3]. The enterprise manufacturing application framework to be proposed in this paper, called EMAF, will be developed by exploiting all these concepts in an attempt to meet the new challenges.

The Integrated Development Environment (IDE) of EMAF is itself an object-oriented framework with an environment to support for the development of different manufacturing applications within the manufacturing application domain [4]. An application domain can be defined as a family of systems with some features in common and others that differentiate
them. The IDE of EMAF is divided into two parts, the front-end and the back-end. The back-end of the IDE provides a means of communication to some common components or component systems of the manufacturing application domain that are stored in a repository. These components or component systems can be customized, adapted and connected easily to develop different applications within the manufacturing application domain. They are reused to increase the reliability and efficiency of a manufacturing application at the same time reducing its maintenance cost and time-to-market. The front-end of the IDE is made up of a Graphical User Interface (GUI), which consists of several tools and guidelines for application developers to reuse the integrated manufacturing components or component systems.

An information system has been recognized as a key to competitiveness by providing the right information to the right people at the right time. Manufacturing organizations have been at the forefront of the business process drive. With major change as the order of the day, these organizations have found that their information systems need also to be changed. Once, the organization's decision-making authority and its information systems were centralized, now the opposite is true. The agile, quick responding manufacturer of the new century will need more than just decentralized data distribution. It will also need personalized, point-of-use information systems capable of immediate, real-time access by empowered workers at all locations, from the office through the plant to the distribution center.

This paper reviews the IDE of an enterprise application framework in the domain of manufacturing. The IDE will serve as a framework for designing, developing, debugging and deploying manufacturing application systems, such as for production planning, inventory control, forecasting, statistical quality control, etc. The IDE also supports the development of manufacturing applications capable of running on a single machine or on a distributed network, which are essential for an enterprise organization.

2. Literature Review

Object-Oriented Technology (OOT) is a hot topic in recent years even though its inception began in the late 1960s. Objects directly relate to the real-world entities. OOT is the idea of using software objects simulating real-world objects for prototyping and developing applications [5]. One of the reasons why object-orientation is beneficial is that programmers can design programs in the context of the kinds of objects and their relationships modeled as in a real-world system. Another benefit is faster development because software is built from standard objects, reusing existing models of corporate processes and rapid prototyping of models from methodologies. OOT indeed has the potential to improve the software development process.

Software is difficult to develop, modify and maintain. Most software are delivered late and over budget [6]. Even in this age of technological advancement, programmers still have to create software from scratch due to the lack of reusable components. By using systematic reuse strategies, a software company can have a competitive advantage over its competitors by delivering quality software products faster and by adapting more quickly to new changes [7].

It is proven that a higher level of abstraction gives greater potential for reuse [8]. There is a consensus that there are many opportunities for reuse that span the entire life cycle of products in a systems development environment. There is a spectrum of reuse, or reusable
resources. Reuse is the use of existing artifacts of software development such as design, specifications, code, documentation and test plans [9], elsewhere within a project or on other projects. Since software development schedules, estimates and costs are heavily influenced by the amount of new code that has to be designed and developed, if software development is on the critical path of a project, reusing of the existing design, specifications and code can have a significant positive impact on costs and schedules for a project.

A significant portion of the study concentrates on systematic reuse in which organizations design software to be reusable. For example, in two of the divisions of Hewlett Packard (HP) [10], reuse is a critical ingredient in achieving productivity and quality objectives. Available design, specifications and code are used several times, the accumulated fixes in each use result in a higher quality product. The reuse technology provides components that can be easily connected to develop a new system. The developer does not have to know how the component is implemented but only need to know how to use it. The resulting system will be efficient, easy to maintain and more reliable. Most works in software reuse address composition technology [11], where components are considered to be largely atomic and ideally unchanged when reuse, although some adaptation may be required. The components are the building blocks used in constructing and deriving a new software.

Component-based software development focuses on building a large software by integrating previously existing software components. This approach can potentially be used to reduce software development costs, assemble systems rapidly, and reduce the spiraling maintenance burden associated with the support and upgrade of large systems. At the foundation of this approach is the assumption that certain parts of a large software reappear with sufficient regularity that common parts should be written once, rather than many times, and that common systems should be assembled through reuse rather than rewritten over and over again. Reuse of components is more productive than reuse of code [12] since reuse of higher-level units involves less application or implementation specific constraints.

Building a program with reusable software components is a good idea, but the integration of different components is very difficult. However, development frameworks, which let a reuser adopt code patterns in his/her project, provide at the source-code level what components provide at the binary level, i.e. the ability to build an application with reusable objects. Frameworks are skeletons that define the basic design of an application. They provide code fragments as the foundation of the application. Moreover, frameworks are much more than mere class libraries. They consist of large sets of code components and include the glue that lets the various classes cooperate. In other words, frameworks are sets of abstract classes with predefined interfaces and problem-adopted event handling. They are more customizable than most components and have more complex interface.

Some of the first frameworks developed to support GUI applications include Apple's MacApp and Next's NextStep in the 1980s. Others of the better-known framework projects were Taligent's CommonPoint developed by Apple and IBM, and Microsoft Foundation Classes (MFC) by Microsoft.
3. Motivations

This research project is motivated by the lack of components or application frameworks within the manufacturing domain and the lack of a well designed environment such as an IDE for developers to reuse components. An IDE deals with the implicit invocation and control of development tools. It provides a consistent view of development artifacts and easy-to-use interfaces to generate, access and control them. This project also involves the development of utilities into the graphical IDE, such as editor, class browser, debugger, compiler, profiler, code checking and testing.

Most of the software developers are looking for a seamlessly integrated tool family offering one step solution for software programming, debugging, testing, documentation, etc. In other words, a well-developed IDE must provide functionality for analysis together with GUI-based operation recording and playback, and integrate them together with other tools for source code static analysis. An IDE involves not just in establishing a technical framework, but also the handling of data and control integration.

The international marketplace is changing rapidly, with customers becoming much more demanding with respect to quality and service. Manufacturers have to evolve into more robust organizations. The need for a computer solution that can support the business strategy and integration of a manufacturing organization is growing. Many software vendors are only offering disparate package systems in support of manufacturing.

Computer-based technology for planning and control has been recognized as a key to competitive manufacturing, providing the right information to the right people at the right time. These manufacturing applications consist of production planning, inventory control, work-in-process tracking, scheduling, production operation control, capacity planning, process capability analysis, process control, control chart plotting, acceptance sampling, statistical analysis, etc. A problem facing this industry is that these applications are not designed to work together and are difficult to integrate. In addition, maintenance of incompatible computer systems developed and supported by different organizations for engineering design and product release is difficult.

Since it is not really that difficult to fix software, implementation of a software can be started first and then adjust it later. However, this can lead to a very serious problem especially in traditional manufacturing industries, where a mistake in design is very costly to correct in later phases. Quality is lower by bugs fixes and patches in the final product and the software no longer corresponds to the specification. These problems can be minimized if not eliminated by using a reusable manufacturing application framework to manage change and modification.

4. The IDE of the Enterprise Manufacturing Application Framework (EMAF)

4.1 Overview of the IDE

The IDE of the enterprise manufacturing application framework (EMAF) is developed on top of Linux platform and is divided into two parts, the front-end and the back-end as shown in Figure 1. The front-end is made up of a window-based user interface. It contains all the common features of window-based applications such as, menu bar, tool bar, icon buttons, etc.
It is developed using pure Java JDK-1.2.2. The purpose of using Java is to make the IDE's front-end platform independent. The choice of a platform is one of the most important considerations for a software development. All software has to run in a platform, which is considered as an operating system of a computer. There are many kinds of operating systems in the market such as, Unix, WindowNT, OS/2, Linux, etc. The problem is that a software running on top of a platform will be limited by its platform, which means that a software cannot run in other platform other than its own platform. By using Java this will not be a problem. Java is an object-oriented programming language developed by Sun Microsystems. The Java technology is interpreted, which means that as long as a platform contains the Java interpreter, Java Virtual Machine (JVM), the platform can run any software application that is developed in Java without any modification. Furthermore, it is very easy to develop a high performance GUI in Java because multiple concurrent threads of activities are supported by the multithreading feature built into the Java programming language. A number of application programming interfaces (APIs) are also provided.

The back-end of the IDE is supported by a set of common components or component systems of the manufacturing domain for developing most of the enterprise manufacturing applications, such as production planning, inventory control, material requirement planning (MRP), statistical quality control (SQC), acceptance sampling, etc. These components or component systems are the high-level view of the main algorithms and logic processes of all the manufacturing software applications that are reused from the IDE. In a manufacturing organization, information is very important and it has to be provided accurately and correctly to the right person at the right time. In order to achieve the goal of developing comprehensive and fast response information systems, C++ object-oriented programming language is used to develop the IDE’s back-end. C++ is one of the most powerful object-oriented programming languages. Besides its common object-oriented features, such as object class definition, inheritance, polymorphism, it also has other features like templates and operator overloading which can enhance the reusability of a component or component system. Furthermore, the components or component systems can be compiled into the native code in order to make the logic processes process faster.

The distributed client-server system is becoming more popular recently due to the growth of internet, world-wide web (WWW) and electronic commerce (E-commerce) applications. It has become an essential feature for most of the enterprise applications. In this project,
Common Object Request Broker Architecture (CORBA) is used as the middleware in order to make the components or component systems distributed and interoperable. CORBA is promoted by the Object Management Group (OMG). It provides a lot of facilities and features to enhance a distributed software application, such as its Interface Definition Language (IDL), Object Request Broker (ORB) and its client-server architecture. The IDL is used to achieve language independence, i.e. two components written in two different languages can communicate with each other. The ORB is used to achieve platform independence, i.e. two components located in two different platforms can communicate. Therefore, a manufacturing application developed from the IDE can be distributed in anyway and can communicate with the applications from other platforms as long as they are CORBA-compliant.

4.2 The IDE's Front-end Internal Structure

The front-end of the IDE is the graphical user interface (GUI) whose internal structure is component-based. It is composed of six major components or component systems, respectively named events manager, editor manager, project manager, options manager, manufacturing wizard manager and code generator. Each of these component systems handles specific functions and are interrelated with one another as shown in Figure 2.

The events manager is a component system that manages all the events generated at the front-end of the IDE. Every event generated will be sent to the events manager which will handle the event by invoking the right function method based on the event received. Every event generated has an ID number associated with it. This ID number is the key for the events manager to identify the right function method to be invoked. In order to recognize the ID number associated with an event, every function method has to be registered to the events manager initially. For example, a function method that displays an open file dialog has to be registered to the events manager with an ID number "1" initially. When a user selects the menu "File->Open", an event with ID number "1" will be generated and sent to the events manager. When the events manager receives the event, it will invoke the function method to display an open file dialog. Sometimes, an event will cause the events manager to invoke several function methods. This is because as long as the function methods are registered with the same ID number, they will be invoked by the same event.

The IDE is also developed to handle multiple files at the same time. This is to cater for the development of complex application software, where numerous source files are involved. It
allows many files to be opened at the same time, and thus allowing the software developer to do coding and debugging at the same time having a good view of all the source files. The IDE handles multiple files at the same time with the assistance of the editor manager. The editor manager is the text editor manager of the IDE. It is a component system to keep track of all the files opened by the IDE. This component system handles the function methods normally contained by a text editor such as, create a new file, open an existing file, save a file, close a file, find words in a file, replace words, undo, redo, cut, paste, etc. It keeps track of information such as what are the files opened, which files are modified, what is the modification for each file, which one is the active file, etc.

The options manager is a component system for handling all the configurations or setting of the IDE. The configurations of the IDE are the size and location of the IDE displayed on the screen, the recent files or projects opened by the IDE, the font type, style and size of the IDE editor, etc. Normally the configurations of the IDE will remain the same with the configurations of its last execution. Therefore, the information of the configurations have to be saved into a file when the IDE exits. This file is known as IDE configuration file. Every time the IDE starts, the options manager will read through the configuration file and initialize the IDE. If the options manager cannot find the file, it will initialize the IDE with default setting and generate a new configuration file. This will normally happen the first time when the IDE is executed. Each time when the IDE configurations are changed, this file will be overwritten by the options manager, in order to keep the most recent setting of the IDE.

To keep track of all the source files under the same application software, the IDE is developed to include a component system called project manager. Every time the IDE starts a new project, the project manager will generate a folder and a project file. The folder is to keep all the files under the project and the project file is to keep track of the information such as, type of project (whether it is production planning, inventory control, statistical quality control or acceptance sampling), name of project, and all the names of the files under the project. Whenever the IDE performs some changes to the project, such as adding or removing a file from the project, the project manager will update the project file. It has to be done this way as when the IDE compiles the project, the information in the project file is used to do the compilation.

The Manufacturing wizard manager is an important component system of the IDE. In developing a manufacturing application from the IDE, some wizards are used to guide the software developer to reuse and customize the back-end components or component systems of the IDE. Therefore, the manufacturing wizard manager can be considered as the bridge connecting the front-end to the back-end of the IDE. In the IDE, there are many different kinds of manufacturing wizards developed to cater for the development of different kinds of manufacturing applications. The manufacturing wizard manager is the component system that decides which manufacturing wizard will be used for developing a particular manufacturing application. From a particular manufacturing wizard, the software developer will have to go through some questions related to the manufacturing application to be developed. After the software developer has finished answering all the questions, the manufacturing wizard has all the information it needs to identify the back-end component systems to be used and then customize them in order to develop the particular manufacturing application. Towards the end, the manufacturing wizard will pass all the information to the code generator component system.
The code generator is a component system that generates code and files. The code generator will generate the necessary code and files for the development of a particular manufacturing application based on the information sent from the manufacturing wizard.

In order to have a view of how these component systems are interrelated with one another, an example of developing a production planning application is described. When a software developer selects to develop a new production planning application, an event will be generated and sent to the event manager. The event manager analyzes the event and calls the manufacturing wizard manager. The manufacturing wizard manager will display the appropriate manufacturing wizard for developing the production planning application. After going through the manufacturing wizard, the information collected from the reuser will be analyzed by the manufacturing wizard manager. The manufacturing wizard manager will then send some instructions to the code generator to generate the code and files needed to develop the production planning application. At the same time, the manufacturing wizard manager will send an event to the events manager in order to call the project manager to generate the project file of this project. When the code generator is generating code and files, the project manager will update the project file. After generating all the code and files needed, the code generator will send an event to the events manager and the events manager will call the editor manager to display all the necessary files for the software developer to edit or perform customization. Figure 3 is a sequence diagram showing the flow of events described above and illustrate how the component systems are interrelated with each other based on the example above.

![Fig. 3: Sequential diagram for new project creation.](image-url)

*Option manager is not included in this example*
4.3 IDE's Front-end GUI

The IDE's front-end GUI is developed using pure Java JDK version 1.2.2. Java contains a lot of advanced component classes for developing the GUI, such as Java Swing packet. The packet is enhanced from Java AWT packet to make it more reliable and easy to reuse. The IDE's front-end is a window-based GUI environment designed for developing manufacturing applications. The main purpose for developing the IDE's front-end is to provide an easy means for the developer to reuse and customize the IDE's back-end component systems, in order to deliver a powerful and easy to maintain manufacturing application software in a shorter time. It is designed to be user friendly and provides a lot of guidelines and wizards to help the developer in manufacturing application development. Briefly, the GUI contains a menu bar, an icon bar, a project window, a text editor desktop and an output window as shown in Figure 4.

The IDE's menu bar has most of the common development application menus, such as File, Edit, View, Insert, Project, Build, Tools, Window and Help. In the File menu, there is a "New" menu item, which is used to create a new application project and a new file. When selecting the "File->New", a "New" dialog box will be displayed. In this dialog box, there are two tab windows, one is "Applications" tab and the other one is "Files" tab as shown in Figure 5 (a) and (b) respectively.
Selecting the "Applications" tab will display a list of pre-built generic manufacturing applications that can be customized by the developer via the IDE. These applications are given names such as Forecast, Schedule, Inventory Control, Production Planning, MRP, Control Chart, Statistical Tools, Sampling Plan and Normal applications. The creation of each of these manufacturing applications is associated with the invocation of an application framework via some wizards to guide the developer to reuse and customize the framework. Developing a normal application which is just an empty application will not trigger any wizard. This is to give the maximum freedom to the developer to develop whatever program he/she likes, without the limitations imposed by the framework. Selecting the "Files" tab will display a list of file types, such as C/C++ header file and source file, Java file, IDL file, and normal text file. Selecting a different file type will provide different extension for the given filename, such as when selecting C/C++ header file type, a ".h" extension will be added to the given filename; and when selecting IDL file type, an ".idl" extension will be added to the given filename. From the file extension, the IDE's text editor will recognize the type of the file and highlight the key words in the file. For this tab, there is an "Add To Project" option, which is to let the newly created file to be added to the current project. In the File menu, besides the "New" menu item, there are many other menu items such as "Open" to open a file, "Save" to save a file, "Close" to close the active file, "Open Project" to open a project, "Save Project" to save a project, "Close Project" to close project and "Exit" to exit the IDE.

The IDE contains a text editor. The text editor is the main application for the developer to do their coding. It allows the developer to write, change and replace code. Most of the text editor functions are in the Edit menu. The Edit menu contains the menu items like undo, redo, cut, copy and paste, which are very helpful functions for editing code. The text editor also contains a powerful search engine. Figure 6 shows the Find Text dialog box. From the dialog box, a word can be searched in many alternatives, such as search by considering the case of the word, search by matching the whole word, search in a forward or backward manner, search the whole file or select text only, search starting from the cursor position or start from the beginning of the file. This search engine is to let the developer locate certain words in a file easily.

The Project menu contains the menu items related to project development. In a project development, a number of files are involved. The menu items in the Project menu are such as
adding new or existing file to the project, removing file from the project, creating a makefile of the project and editing the project makefile.

The Build menu contains the menu items like “Compile”, “Rebuild”, “Compile Project”, “Rebuild Project”, “Run”, and “Debug”. These menu items are common functions for software development. They are used to compile, debug and execute the current application being developed.

Figure 7 shows the icon bar of the IDE’s front-end GUI. The purpose of the icon bar is to let the developer execute certain functions of the IDE faster by just one click. As such, the icons displayed on the icon bar are the most frequently used functions in software development, such as compile, run and debug. Actually all of the functions in the icon bar can be found in the menu bar. Besides the icon bar, there is another method of fast execution function supported by this IDE, which is the shortcut key. For example, pressing the keys Ctrl+S at the same time will execute the save function to save the current editing file.

In the GUI, there is a project window on the left. The project window will show all the files under the project currently being developed by the IDE. The project window is acting as a project files navigator to let the developer easily navigate through all the files under a project. At the top of the project window are four icon buttons. The first one is for adding a new file to the project, the second one is for adding a new folder to the project, the third one is for removing a file from the project and the last one is for adding an existing file to the project. All of these functions can be found in Project menu. Adding these icon buttons at the top of the project window allows the developer to execute those functions faster and easier. At the bottom of the GUI, is an output window. This window will show all the messages during the compilation or execution of a project. The messages shown in the window let the developer do debugging. All the files opened by the IDE will be displayed inside the text editor desktop. This allows the developer to handle all the opened files easily by grouping them inside the desktop.

4.4 The IDE's Back-end Component Systems

The back-end of the IDE provides a means to interface with the integrated enterprise manufacturing application framework stored in the repository. The IDE’s back-end is divided into three layers as shown in Figure 1. The top layer is the client manufacturing component systems layer. This layer contains the component systems which are used to develop the client part of a manufacturing application software. The second layer is the CORBA middleware layer. This layer contains the component systems that provide features like language and platform independent in order to achieve distributed object computing in heterogeneous environments. The bottom layer is the server manufacturing component systems layer. This layer contains the common component systems which are used to develop the server part of a manufacturing application software. Manufacturing is a very huge domain,
as such, this project initially provides component systems for supporting development of application software within the essential subdomains of the manufacturing domain. For this part of the project, it focuses on the two important parts in manufacturing, which are the Production Management (PM) and the Statistical Quality Control (SQC).

Production Management is to control the production and inventory in a manufacturing firm by conducting several activities such as production planning, inventory control, capacity planning, etc. For Statistical Quality Control, SQC methods are useful tools for appraising and monitoring quality performance and are key ingredients to successful application of quality control. Quality control relies on the continuous monitoring of quality of the input and output of the processes producing products and services. For example, a control chart is used to monitor the quality of the products in the process of the production line.

4.4.1 Production Management

There is a widely used generic version of production management system, called the Manufacturing Resource Planning (MRP II) System [13]. The common structure of the MRP II system is shown in Figure 8.

![Fig 8: Generic model of Manufacturing Resource Planning (MRP II) System](image)

Each of the boxes represents a separate component system. Each of the component systems offers different services. The arrows in the diagram indicate information flows. This system is divided into three levels. The first level is the Top Management Planning level, which consists of production planning and resource planning subsystems. Some manufacturing companies also include business planning and marketing planning subsystems into this level. At this level, there are long planning horizons (about five years) and broad decisions. The second level is the Operations Management Planning level, which consists of Master Production Scheduling (MPS), Rough-Cut Capacity Planning (RCCP), Material Requirement Planning (MRP) and Capacity Requirement Planning (CRP) subsystems. At this level, there are short planning horizons and more specific decisions. The last level is the Operations Management Execution level, which consists of the Production Activities Control (PAC) subsystem. This level is to make sure productions and jobs are in schedule.
4.4.2 Statistical Quality Control (SQC)

SQC is the application of statistical techniques to ensure satisfactory quality. There are three major subgroups of techniques: statistical process control, acceptance sampling and statistical techniques as shown in Figure 9. In Statistical Process Control (SPC), tools called control charts are used primarily to prevent or detect production of defective products (i.e. finished goods, assemblies or components) or services. This allows the process to continue or stop and inspection is carried to find the cause. There are two types of control charts: control charts for attributes and control charts for variables. Acceptance sampling is the application of statistical techniques to determine whether a quantity of material should be accepted or rejected, based on the inspection or testing of a sample. To judge the quality of products or services, statistical techniques such as frequency distribution, measures of central tendency and dispersion, regression and probability distribution can be used [14].

The development of the object-oriented application framework is based also on the common application systems of the SPC and acceptance sampling, which are widely used in quality control of manufacturing. The construction of a control chart application system basically involves implementing a reusable production control component in the framework for the purpose of adaptation or configuration. A generic production control component will be provided by the application framework to support a variety of control charts such as R-chart, X-chart, n-chart, etc, in one of its variability points meant for adaptation according to the specific needs of the reuser.

5. Concluding Remark

There is a strong request for an integrated development environment for the development of enterprise information systems. This paper describes the architecture and functionality of the IDE for an enterprise manufacturing application framework. The enterprise application framework provides a comprehensive, Java-based platform for developing manufacturing applications that simplifies application development and deployment, supports heterogeneous client and server platforms, leverages existing skills and assets, and delivers a scalable, reliable, and manageable environment for the reuser.
One of the reasons of using CORBA as the middleware for the framework is that the component systems can be distributed and support client-server architecture, which is essential for enterprise manufacturing application systems. However, the problem of using CORBA is perhaps related to the fact that the learning curve for reusing components would be longer and tougher, since the reusers have to know not only the object-oriented language for assembling of components, they also need to have the knowledge to program in CORBA. Besides, CORBA also limits and restricts the customization of components interface. This would reduce the reusability of the application framework.

Currently, the above problem of CORBA is being solved by porting the framework into a complete Java environment. By using Java for the development of the entire IDE, there is no need to use CORBA as middleware for the application framework, as Java has an API called Java Bean API with the distribution capability, which can be used to develop distributed components. This will reduce the developers’ learning curve as they do not need to know CORBA language other than the language used to construct the component systems. Java was not chosen initially due to the longer response time of the Java application processes. This is also a main problem faced by most Java application developers. However, this problem can be subdued by using a more powerful and advanced computer hardware especially with the present tremendous improvement of computer hardware in terms of speed and reliability.

References


