A Flexible Process Monitoring and Control System Based on Data Engine and Layout Engine

Zuye Zhao, Lichao Zhang, and Yusheng Shi

***Abstract*—Due to the complexity of deployment and the lack of self-adaptability for different kinds of terminals, the traditional flexible process monitoring and control systems can't satisfy the constantly changing customer requirements in the ever-shortening development time. To solve these problems, a new flexible process monitoring and control system is presented in this paper. By combining with the description information based on monitoring object, the general client program which consists of data engine and layout engine can generate totally different contents of client program in real time. As the core component of general client program, the layout engine is responsible for the generation of the user interfaces of client program, and the client program can adapt to different resolutions because of the grid-based locating method adopted by layout engine. With the help of data engine, the data from various data sources can be managed by a unified framework. Finally, by adopting the completely runtime-oriented engine architecture, the users can modify the client programs by themselves when the system is running. In order to prove the features mentioned above, the system has been implemented in a servo-press-based production line.**

***Index Terms*—Data Engine, Layout Engine, Process Monitoring and Control System, Grid Layout, XML**

I. Introduction

PROCESS monitoring and control system (PMCS) is one of the most critical parts of automated manufacturing system, and its purpose is to establish a interaction between the users and the devices. In order to implement the interaction, the PMCS must include the following functions: (1) data acquisition and data visualization; (2) command input and command routing. With the rapid development of economic and technology, manufacturing enterprises tend to use a manufacturing system which can rapidly respond to the changes of products, so that they can have the capabilities to meet constantly changing customer requirements and produce high quality products in the shortest possible time and at lowest possible cost [1]. But the early PMCS is oriented towards a specific device and application. For example, Song [2] used C# language, SQL Server database and RS485 interface to implement a coal mine monitoring system. Shi [3] designed a broaching machine online machining process system based on Kistler 9232A surface mounted strain sensor and National instruments (NI) PXI module. Due to using general programming languages (C#, C++) and hardware APIs (Application Programming Interface) which are bound to a specific device, the above systems are hard to achieve rapid response to the changes of hardware and system function.

To be compatible with various devices and applications, many researchers have proposed more open and flexible solutions for PMCS. Zhang [4] proposed a configurable PMCS which consists of the tools such as project manager, GUI developer, system builder, and etc. By using these tools, the developers can quickly build a monitoring system for specific application. In order to describe the PMCS in a simpler way, Sun [5] adopted XML document and role-based access control mechanism to save the system information. Besides the above experimental systems, many well-known automation system manufacturers have also presented commercial flexible PMCS, such as WinCC (Siemens), iFix (GE), InTouch (Wonderware), and etc. By now WinCC is the most representative system, and has been used in some key fields of manufacturing [6]-[8].

According to the process of development and running, the above systems all can be divided into two stages: the design stage and the running stage. In the design stage, a new role called system engineer is introduced to design the client programs by visualization development tools [4]. In the running stage, the users (equipment operators, process engineers, quality inspectors, and etc.) can access the preset contents which have been designed by the system engineers. Although these PMCS solutions can simplify the design process and reduce the development time, the architecture based on two separated stages actually does not change the fundamental development process, and they still suffer from a number of problems as follows.

1) The system client programs are lack of adaptability for various kinds of terminals. Due to adopting the visual development techniques, the client program can only be designed for a specific screen resolution. If the terminal's resolution has been changed, the layout of client program may get messed up, and some useful information may be lost (Fig. 1). As the result of that, the developers have to design customized client programs for different kinds of terminal. Because of the

Manuscript received July 11, 2012; revised August 5, 2012. This work was supported by National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant No. 2010ZX04001.

Zuye Zhao is with State Key Laboratory of Materials Processing and Mold & Die Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. (phone: +86-27-87558465; fax: +86-27-87558465; e-mail: zhaozuye@gmail.com).

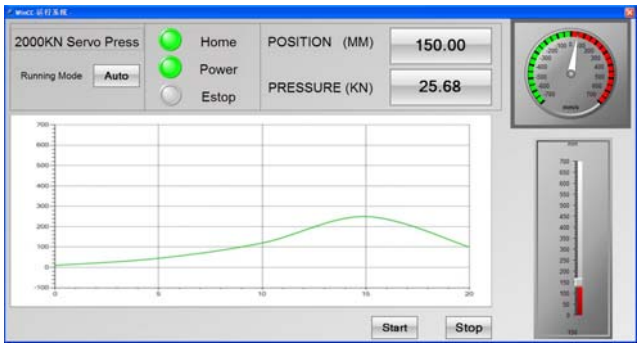
Lichao Zhang (corresponding author) is with State Key Laboratory of Materials Processing and Mold & Die Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. (phone: +86-27-87558465; fax: +86-27-87558465; e-mail: dr.teac@gmail.com).

Yusheng Shi is with State Key Laboratory of Materials Processing and Mold & Die Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. (phone: +86-27-87558465; fax: +86-27-87558465; e-mail: shiyusheng@263.net).

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

diversification of terminals, the development and update of client programs will lead to lots of repetitive work

.



A. At 1440×900 pixels. The client program's contents are complete.

B. At 1024×768 pixels. Some controls on the right of the client program are lost.

Fig. 1. Display of a client program created by WinCC

2) The deployment of system is complex. With the development of network manufacturing system, the developers need to create many kinds of client program for the ever-increasing users and terminals. Moreover, in order to guarantee the normal running of the system, each terminal has to be installed with a series of runtime libraries. For these reasons, the difficulties of system deployment and maintenance will increase greatly, especially when the network topology or the core function is changed. By adopting the Browser/Server model, the complexity of deployment can be reduced effectively, but it still can't take place of the Client/Server model because of the problems on compatibility, functionality and security.
3) The client program can't be modified in runtime. Due to the separated stage of design and running, the client program can only be modified in design time. With the expansion of manufacturing system, the number of users and corresponding terminals continuously increases. If the modifications for each user still depend on the traditional development mode, it will certainly bring a huge workload, so the system can't respond changes quickly.

To deal with these problems, a flexible process monitoring and control system based on layout engine and data engine is presented in this paper. Also, a monitoring-object-oriented method for describing client program and a unified running resources dispatch framework are introduced. By adopting these methods and engines, the system finally realized the single-client-based multi-platform deployment, and when the system is running, the users can modify the client programs by themselves according to their demands. The rest of the sections of this paper are organized as follows. Section 2 describes the overall architecture of the system. Section 3 presents the deployment and generation of client programs, and an implementation of the system is presented in Section 4. Finally, the conclusions are made in Section 5.

II. System Overview

A. System Architecture

In order to solve the problems mentioned above, the new flexible PMCS must have five features as follows. (1) The system only uses one kind of client program which doesn't include any specific contents before the system is running. (2) All the interactive contents of client program are generated in runtime according to the description information stored on system servers and the properties of system terminals. (3) The data processing procedure is separated from the user interface, and all monitoring data is managed by a unified framework. (4) The layout of client program is based on relative position, and thus it can be compatible with different resolutions. (5) The description information of client program is bound with the user login information, and thus the users can get their personalized client program on any terminal. To realize these features, the final system must include the basic parts as follows: (1) general client program, (2) user information server, (3) system resource server and (4) history data server (the architecture is logic-based, and in physics some different parts can be deployed on the same computer). Fig. 2 illustrates the basic parts of the system.

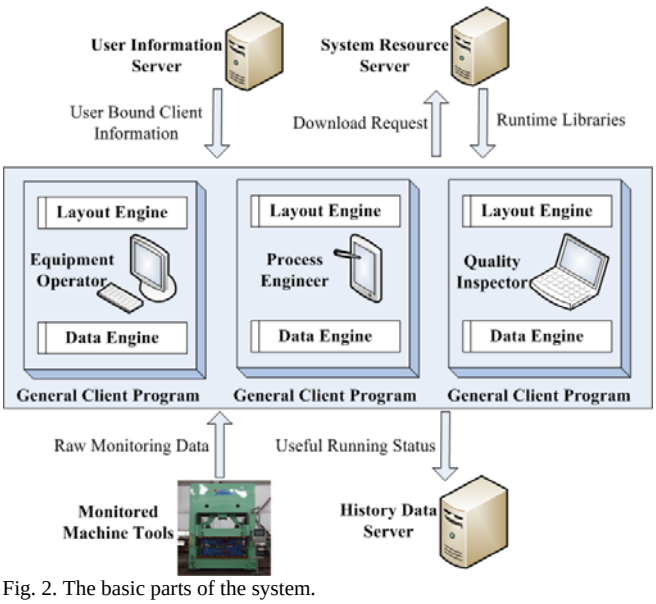


Fig. 2. The basic parts of the system.

General Client Program: The function of general client program is similar to the browser in a network system, and there are no any specific contents in a general client program. Be considered as an interpreter, the general client program

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

receives the certain kind of formatted information and generates the interactive contents of client program in runtime. The data engine and layout engine are the key components of general client program. The data engine provided a unified management for the data from different sources (sensors, databases, user interfaces, and etc.). It not only makes the system be compatible with various data protocols, but also achieves the complete data-driven running of system. The layout engine is responsible for the generation of user interfaces and the presentation of monitoring data.

User Information Server: As mentioned above, the specific contents of client program should be bound to the users. Therefore, besides the authentication information and the permission information, the user information server is used to save client program information too.

System Resource Server: The general client program doesn't include any runtime libraries and running resources, so the system resource server is introduced to provide running resources for the general client program. After obtaining the information of client program, the service management module of general client program will download the corresponding running resources from system resource server automatically to guarantee the normal running of system.

History Data Server: As a standard component of PMCS, the history data server is responsible for the storage of historical monitoring data.

Based on the parts mentioned above, the typical running process of the system is shown in Fig. 3.

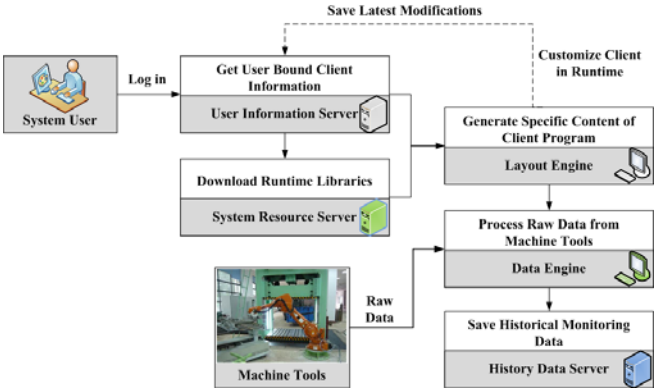


Fig. 3. The typical running process of the system.

B. *Data Engine and Monitoring Object*

From a macro perspective, the normal running of monitoring system depends on the flow of data between different functional modules. To deal with the data exchange between various data sources, a management mechanism called "data engine" is proposed. Meanwhile, as the core part of data engine, the concept of "monitoring object" is also put forward. Be different from the isolated monitoring data in the traditional sense, the monitoring object, which can describe certain status or characteristic of the system (e.g., slider speed, hydraulic oil temperature and cutting pressure), is the result of the combination of related discrete data. The data sources of monitoring object include sensors, databases, xml files and even a block of text from the input box of client program. By applying the plug-in technology, the monitoring object can be compatible with any data source under the uniform data access interface.

The purpose of data engine is to establish the mapping relationships between the discrete data from different data sources and the useful status information of the manufacturing system. The process of creating mapping relationships is as follows. (1) The developers create a uniform data access interface which includes two basic functions (setString and getString), and the key parameter of these functions is the name of data item which needs to be accessed. Whether OPC server or XML file, all kinds of data sources used in the system must implement the data access interface in the form of DLL (Dynamic Link Library), and thus the data engine can access any data source in the same way. (2) By adopting (1), the relationships between the monitoring objects and the raw data from different data sources can be described. X_N is the name of data item in different data sources, and Y is the name of final monitoring object. When the system needs to read a monitoring object, the data engine first accesses every data item in the mathematical expression, and then the final result will be calculated automatically. The mapping relationships can only be established by the system engineers, and all of them must be stored on system servers. (Fig. 4 shows the program used to define the mapping relationships).

$$Y = F(X_1, X_2, X_3, \ldots X_N) \quad (1)$$

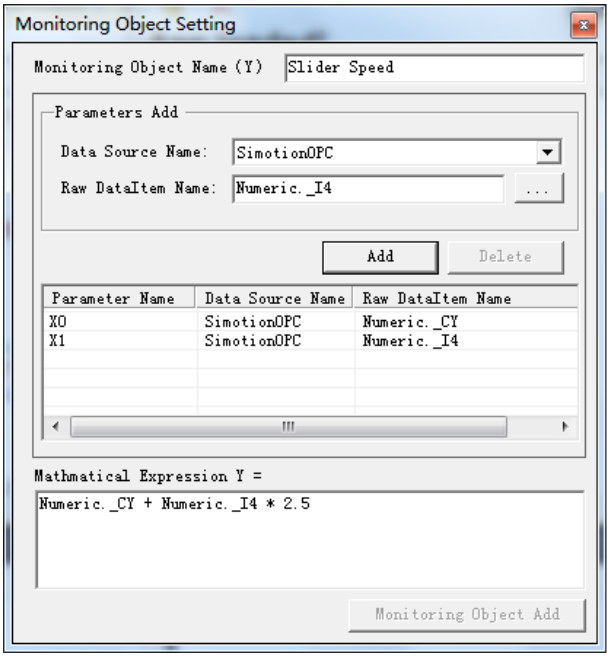


Fig. 4. The program used to define the mapping relationships.

After the underlying data mappings have been established, a container control named "DataContainer" is designed to make the data visualization controls of client program associate with the monitoring objects. To achieve that goal, the DataContainer must include two basic functions: (1) Provide the properties to describe the process mode for data interaction. (2) Make the users have the capabilities to replace the corresponding data visualization control for certain monitoring object in runtime.

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Based on the functions above, the following properties should be included by DataContainer.

Monitoring object name: The mapping relationships used by data engine are get by the name of monitoring object.

Data triggering mode: All the data operations of client program can be regarded as event triggering. For example, the periodical data refreshing is based on the tick event of timer, and the writing of system parameters is based on the click event of write button. The mode of data interaction is defined by this property.

Bound data visualization controls: In order to display the information of monitoring object, the DataContainer is usually bound with several data visualization controls. When the system is running, each DataContainer traverses all the data visualization controls bound with it, and then displays the monitoring object according to the type of data visualization control.

Data script: By setting the property, the final data shown in data visualization controls can be modified by the users according to their requirements.

C. Layout Engine

After the data engine running normally, the layout engine will generate the user interfaces based on the resolution of terminal. The controls are positioned with absolute coordinates is the major reason for layout chaos when the resolution changes. In order to avoid this problem, the layout engine adopted grid-based layout mode, in which the location and size of control are all described by the relative positions of the grids that covered by the controls, so the layout of client program can be irrelevant to absolute coordinates.

In the design process, the main window of client program is divided into several grids first (the grid is a rectangle), and then the basic controls are filled into the corresponding grid areas according to their preset sizes, which are indicated by m×n grids. The grid areas which have been covered by the controls can further be divided into finer grids, and then the new grids can be filled by new controls. By repeating the operations, the final layout can be created. To be compatible with the grid-based layout mode, the controls used in the client program should include the following properties, and the schematic is showed in Fig. 5.

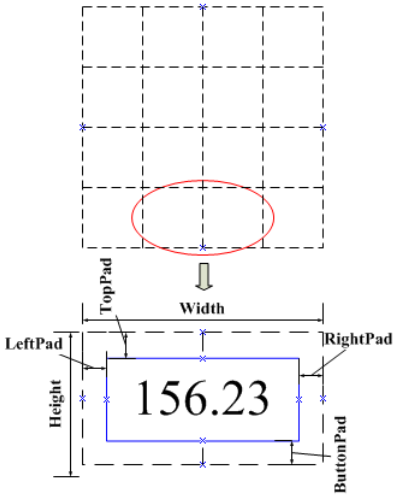


Fig. 5. The key properties of the controls used in the client program.

Width: The number of horizontal grids covered by the control.

Height: The number of vertical grids covered by the control.

Top Padding, Left Padding, Bottom Padding, Right Padding: The spacing between the edges of the grids and the corresponding edges of the control filled in the grids.

IsFixedAspectRatio: Due to adopting the grid-based layout mode, the shape of control in client program may change with the resolution, but some special controls need to keep their aspect ratios unchanged for better visual effects. The property specifies whether the aspect ratio of control can be changed.

CanBeCovered: Specifies that whether the grid areas covered by certain control can be filled by other new controls.

IsDataContainer: If the property is true, a same size DataContainer which pack itself will be created while the data visualization control is generated, and then the system engineer can bind a monitoring object to the DataContainer. If the property is false, the data visualization control will not be bound with any monitoring object, and thus the display data of this control can be set freely.

After finishing the design, the client program information should be stored on the system servers. The hierarchical architecture of XML is very adaptive for describing the grid-based layout, so the XML document is adopted to save client program information. 'Control' tag is the core part of description document, and it includes two kinds of sub tags: one is used to save the properties of control itself, includes the location information and the binding DataContainer's information. The other one is used to save the properties of control covered grid areas, includes the number of redivided grids (row count, column count), the new controls in this area, and etc. The description document of client program formatted in XML is shown in Fig. 6.

```
<Control name = "speedGauge">
    <Location>
        <X>0</X>
        <Y>0</Y>
    </Location>
    <Size>
        <Width>1</Width>
        <Height>1</Height>
    </Size>
    <Padding>
        <Top>0</Top>
        <Left>0</Left>
        <Bottom>0</Bottom>
        <Right>0</Right>
    </Padding>
    <FontWeight>1</FontWeight>
    <CanBeCover>true</CanBeCover>
    <IsDataContainer>false</IsDataContainer>
    <IsFixedAspectRatio>false</IsFixedAspectRatio>
    <DataContainer>
        <MonitoringObjectName>slider speed</MonitoringObjectName>
        <DataTriggeringMode>Timer</DataTriggeringMode>
        <RefreshCycle>10</RefreshCycle>
        <DataScript></DataScript>
    </DataContainer>
    <Grid>
        <Row>4</Row>
        <Column>4</Column>
    </Grid>
    <Control name = "speedTextBox">
    </Control>
</Control>
```

Fig. 6. The description document of client program formatted in XML.

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Based on the controls with the properties mentioned above and the layout engine, the design process of client program is as follows.

Step1: Divide the blank window of client program into m×n grids, and then drag and drop the basic controls into the main window.

Step2: Set the properties of basic controls.

Step3: Set the properties of DataContainers, and then bind the monitoring objects to corresponding controls.

Step4: Divide the grid areas covered by existing controls for further detailed design.

Step5: Save the information of client program by the XML document mentioned above, and then the document is uploaded to the system servers automatically.

Step6: Bind the description information of client program to the corresponding users.

III. The Deployment and Generation of Client Programs

The deployment of process monitoring and control system can be divided into Client/Server mode and Browser/Server mode. Because of the good performance, abundant third party libraries and the simple interface, the Client/Server mode is adopted by most monitoring system now. But with the increase of the system scale, the deployment is more and more complicated. On the contrary, the deployment process of Browser/Server mode based on general browser is simple. However, it can't realize complex functions with good performance and its compatibility is limited. By combining the advantages of the two modes, an architecture based on general client program is proposed. Be similar with the browser in Browser/Server mode, the general client program doesn't contain any specific contents. When the user logs in, the user interfaces and execution logics are generated in real time based on the client information stored on system servers. In order to satisfy the runtime-oriented flexibility of the system, every user must have the capabilities to modify the client program in runtime, and thus the client program information should be bound to the user information. The modifications of the system are uploaded to the system servers in real time, so the personalized modifications made before will still remain completely when a user visit the system on different terminals.

Based on the architecture mentioned above, the system can be divided into three main stages (Fig. 7), which involve the developer, the system engineer and the end user.

Programming Stage (Developer):

Step1: Develop the general client program.

Step2: Develop the data access functions for different data sources used in the system based on the uniform interface in data engine.

Step3: Develop the DataContainer control and different kinds of data visualization controls according to the basic properties mentioned above.

Design Stage (System Engineer):

Step4: Build the mapping relationships between the monitoring objects and the discrete data from different sources by mathematical expressions.

Step5: Design the standard reference client programs for different kinds of users, and then bind the client program information to the corresponding users.

Step6: Install the general client program on each terminal of the system. Store the running resources on system servers.

Running Stage (End User):

Step7: Log in the system, and the specific contents of client program are generated by layout engine.

Step8: The raw data is processed by data engine and the final information of monitoring object is presented by data visualization controls created before.

Step9: Modify the client program according to actual requirements in runtime, and all the modifications are stored on the system servers.

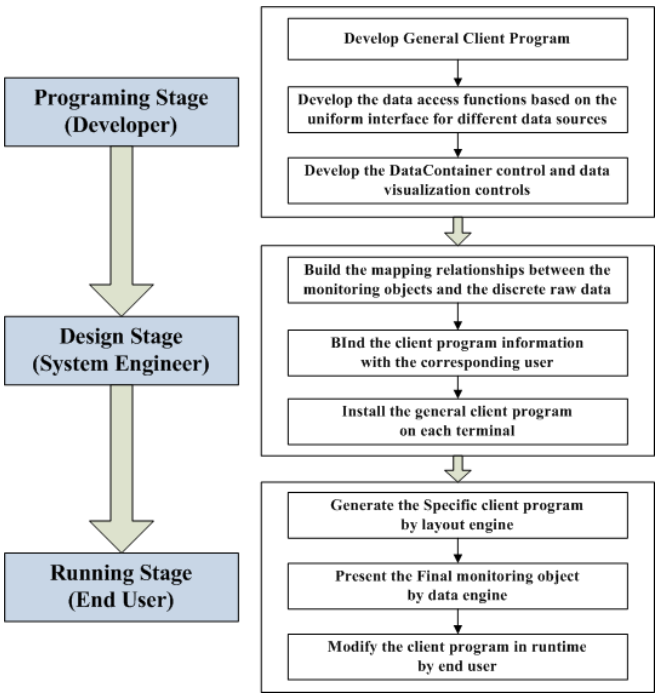


Fig. 7. The complete development and running process of the system.

IV. Implementation and Results

The system presented in this paper has been implemented in a servo-press-based production line. The production line consists of four parts: heating furnace, delivering frame, industrial robot and servo press (Fig. 8). The working procedure of the production line is as follows. (1) The industrial robot puts the metal sheet on the delivering frame. (2) The metal sheet is delivered into the furnace for heating. (3) After reaching the preset temperature, the delivering frame pushes out the heated metal sheet. (4) The industrial robot puts the metal sheet on the working area of the servo press. (5) The servo press stamps the sheet by the preset motion curve. In the whole process, the variables which need to be monitored are temperature of heating furnace, velocity of the robot, pressure of the servo press, and etc. According to different working steps, the client programs can be divided into heating furnace client program, robot client program and press client program.

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

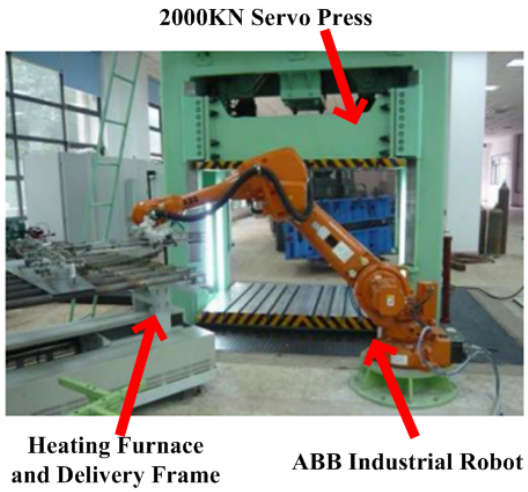


Fig. 8. Servo-press-based production line.

The hardware architecture of the system is tabulated in Table I.

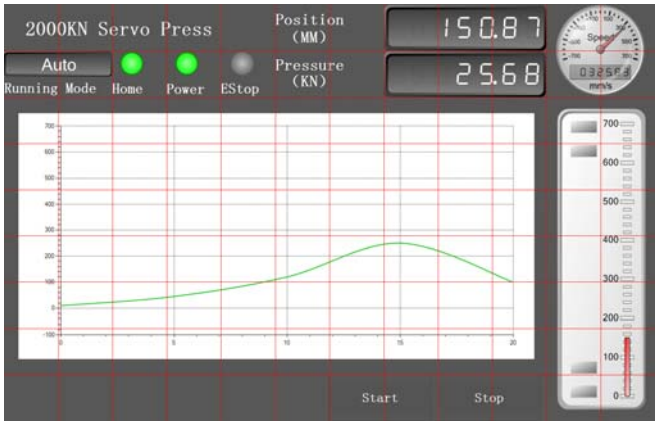
TABLE I
THE HARDWARE ARCHITECTURE OF THE SYSTEM

Hardware Module	Product Model	Data Access Mode
Heating module	Euroherm 3208	Driver and API
Delivering motor	Panasonic MDMA302P1H	Driver and API
Industrial robot	ABB IRC5	OPC
Press motor	Siemens Simotion D425	OPC

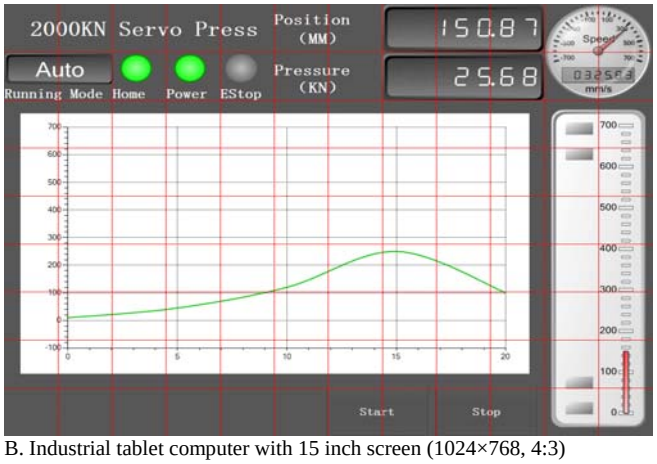
The terminals of the system include two types: (1) Desktop PC with 19 inch screen (1440×900, 16:10); (2) Industrial tablet computer with 15 inch screen (1024×768, 4:3). During the development, the developer programs the uniform data interfaces for different kinds of hardware first. Then the system engineer builds the mapping relationships between the monitoring objects and the discrete underlying data from sensors and other sources. Finally the role-based standard client programs are created by the system engineer. When the system is running, the user bound contents of client program will be generated in real time, and the user can modify the client program under their permissions in runtime. Fig. 9 displays the client programs on the terminals with different resolutions, and the layout engine makes the client program adapt to different resolutions.

V. CONCLUSIONS

In this paper, a flexible process monitoring and control system based on data engine and layout engine is proposed. Based on the data engine, a unified data exchange framework for various data sources is created. With the help of layout engine, the client program can adapt to different resolutions. The general client program which consists of data engine and layout engine is the key part of the new system. The general client program is like the browser used in Internet, and it doesn't include any specific contents. When the system is running, the interactive contents of client program are generated dynamically according to the user bound client information stored on user information server and the running resources downloaded from system resource server. The system has been implemented in a servo-press-based production line, and the features of the system have been proven effectively. In future work, the combination of the system and the enterprise IT system will be focused.



A. Desktop PC with 19 inch screen (1440×900, 16:10)



B. Industrial tablet computer with 15 inch screen (1024×768, 4:3)

Fig. 9. Display of a client program created by WinCC at 1024×768 pixels, and some controls on the right of client program are lost.

REFERENCES

[1] X.W. Xu, S.T. Newman, "Making CNC machine tools more open, interoperable and intelligent-a review of the technologies," *Computer in Industry*, vol. 57, No. 2, pp. 141-152, Feb. 2006.

[2] D. Song, Z. Ren and Y. Gu, "Design and Implement of An Intelligent Coal Mine Monitoring System," in *Proceedings of the 8th International Conference on Electronic Measurement & Instruments (ICEMI' 2007)*, pp. 849-852, Aug. 2007.

[3] D. Shi, N. Gindy, "Industrial Applications of Online Machining Process Monitoring System," *IEEE/ASME Transactions on Mechatronics*, vol. 12, No. 5, pp. 561-564, Oct. 2007.

[4] G. Zhang, Y. Chen, C. Y and Z. Zhou, "Configurable monitoring system for industrial process control," *Int. J. Adv. Manuf. Technol.*, vol. 29, No. 3-4, pp. 336-341, Jun. 2006.

[5] G. Sun, Y. Chen, Z. Zhou and Z. Min, "A configurable access control system for networked manufacturing monitoring using XML," *Int. J. Adv. Manuf. Technol.*, vol. 39, No. 11-12. pp. 1252-1261, Dec. 2008.

[6] M. Abdel-Bary, S. Abdel-Samad and K. Kilian, "Automatic control system for the COSY-TOF vacuum system," *Nuclear Instruments and Methods in Physics Research A*, vol. 539, pp. 93-99, 2005.

[7] J. Cui, X. Zhang, J. Zhu, "Design and Realization of Remote Monitoring System for Automatic Forced Fitting Production Line," in *Proceedings of 2nd International Conference on Manufacturing Science and Engineering*, vol. 201-203, pp. 2354-2359, Apr. 2011.

[8] Z. Jiao, H. Cui, Q. Guo, "Monitoring System Design of Fuse Automatic Assembly Machine based on WinCC," in *Proceedings of International Conference on Materials and Manufacturing (ICMM 2011)*, vol. 299-300, pp. 1182-1185, Sep. 2011.

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)