Proceedings of the 1998 IEEE
International Conference on Robotics & Automation
Leuven, Belgium • May 1998

Systematic Creation and Application of Virtual Factory with Object Oriented Concept

Ming-Hung Lin and Li-Chen Fu
Dept. of Computer Science and Information Engineering
National Taiwan University,Taipei,Taiwan,R.O.C

Abstract

This paper proposes a systematic method of creation and application of a virtual factory. Basic definition of a of virtual factory is first given , followed by its foundation principles and architectural overview, and then a systematic method based on object technologies is proposed to create the virtual factory. Specially, for a real manufacturing environment, a corresponding virtual factory can be created via the polymorphism parameterized universal virtual factory with several interactions. The virtual factory can be linked to the real factory and enable simulation-based control by means of the switching architecture we proposed. Its reliable predictions can not only prove the production scenarios but also improve the decision making process of acquisition managers in the applications for flexible automated production. The present work is being applied to the experimental robotized flexible assembly system developed by Intelligent Robotics and Automation Laboratory in National Taiwan University.

1 Introduction

To increase global competition, the manufacturing industry must bring well-designed, well-manufactured and more competitively priced products to the market. So, industries need to be able to quickly develop new and modified products at the right quality. The representative system and software support tools for automated manufacturability are discussed in great detail [1]. Today, manufacturing industries are relying increasingly on distributed manufacturing enterprises organized by multi-enterprise partnerships, which makes production tasks more challenging. To prove the production scenarios and to support the generation of more reliable estimates of production costs and schedule, modeling and simulation are deemed to improve production flexibility and, hence, to reduce production costs. These requirements drove the design of a new CIM system, whereby McGehee provides an overview of the MMST CIM system framework which is based on open distributed systems and object technologies. [2]. The vision of virtual manufacturing is to provide a capability to "Manufacture in the Computer". In essence, virtual manufacturing will ultimately provide a modeling and simulation environment so powerful that the fabrication of any product, including the associated manufacturing processes, can be simulated in the computer [3]. The term virtual manufacturing should be constructed in a broad sense to include not only production, but also suppliers, customers, and other processes that impact production. There are three paradigms of virtual manufacturing [4], namely, Design-Center virtual manufacturing, Production-Center virtual manufacturing, and Control-Center virtual manufacturing. At the highest level, virtual manufacturing is expected to realize the following benefits: affordability, producibility, flexibility, shorter cycle times, responsiveness, by referring to the description of the virtual manufacturing. A virtual factory refers to manufacturing activities carried out not in one central plant but rather in multiple locations by suppliers and partner firms [5], To build it, one must have a deep understanding of the manufacturing capabilities of all parties in the production network. This paper provides architectural overview of a virtual factory and a systematic method of creation and application of a virtual factory. The following consists of six sections. Section 2 describes the foundation, architectural principles and overview of a virtual factory, Section 3 introduces an *polymorphism parameterized universal virtual factory* and it's components and functionality, and the steps for the creation of virtual factory from it. Section 4 introduces a *switching architecture*, describes how to apply the virtual factory, and finally explains the link between the virtual and real manufacturing and the enabling the of simulation-based control via the switching architecture. Section 5 describes a simple example. Finally, some conclusions are made and some future research directions are given in Section 6.

0-7803-4300-x-5/98 $10.00

2 Virtual Factory Foundation, Principles and Architecture Overview

Essentially, the virtual factory is required to integrate and automate the manufacturing process virtually. Specially, the following functions are to be provided:

- **Visualization**: simulating the factory operation and showing factory equipment with friendly user interface or virtual reality technologies.
- **Modeling and Simulation**: supporting simulation to fit the virtual environment to real control system and enable the simulation-based control, including tracking of work-in-progress and machine monitoring, control and diagnosis, process monitoring, control and diagnosis .
- **Virtual Cooperation**: across the geographically distributed enterprise.
- **Virtual Prototyping**: generation and maintenance of virtual process and virtual product specifications and recipes.

Traditionally, software system for manufacturing has been characterized as expensive to develop and difficult to maintain and extend. Object technology represents a revolutionary approach to develop the software system. A detailed description of the object-oriented approach can be found in [6]. The virtual factory can be developed using object-oriented analysis, design, and rapid prototyping methodologies. A virtual factory is comprised of a set of objects with well defined services and a graphical object-oriented user interface under direct manipulation. In addition to object-oriented foundation, virtual factory is based on several fundamental principles which are simply reflection of visions of the virtual factory architects. These principles are discussed in the following:

- **Integrating the Legacy Data**: It is able to integrate the already legacy manufacturing database.
- **Fully Integrated Dynamically Bound**: Each application is comprised of objects with services, and the external application requiring the services does not bind to the object until run time.
- **Transparent Object Location**: Any object can communicate with one another without knowing its location.
- **Generic Data Encapsulation and Abstraction**: It enables the virtual factory framework to be used in alternative manufacturing domain.

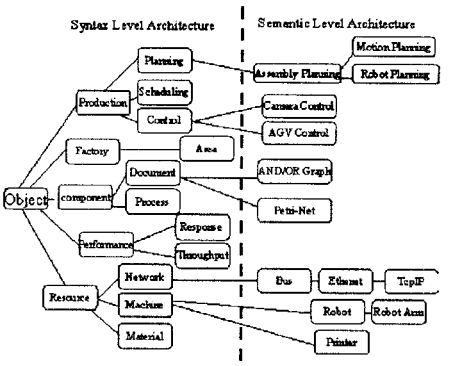

Figure 1: The virtual factory architecture overview

- **Dynamically Bound Polymorphic Behavior**: It can accommodate a wide variety of object types without type specific code.

The architecture of a virtual factory is two levels, the syntax level and the semantic level. The object class inheritance hierarchy represents the syntax level. There exist the most abstract and generic classes, which are reusable across a variety of domains of manufacturing. The class abstraction consisting of both data and procedures which enable software entities to model the virtual world and can be changed without affecting other objects. The semantic level is represented by the object instance with dynamically polymorphic binding. Each instantiated object provides a set of well-known services. The architecture overview is shown in Fig 2. Together, these concepts provide a powerful developing for software engineering in general and system architecture in particular. The following sections describe the method for creating a virtual factory from the real word.

3 Systematic Creation of Virtual Factory

The development of a virtual factory is much complicated, including determining what to model, and determining the degree of abstraction and level of depth required. Besides, we also have to consider how to fit the virtual environment into the real world, i.e., the capability of virtual corporation and its flexibility. This paper provides a systematic method for developing a virtual factory via *Polymorphism Parameterized Universal Virtual Factory* with several interactions. A *Polymorphism Parameterized Universal Virtual Factory* (PPUVF) is composed of four main parts which are *Virtual Factory Probe*, *Virtual Factory Shell*, *Virtual Factory Broker*, and *Virtual Factory Wizard*. A virtual factory probe is responsible for extracting information and data , producing syntax level architec-

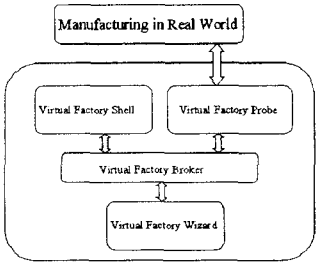

Figure 2: The Polymorphism Parameterized Universal Virtual Factory

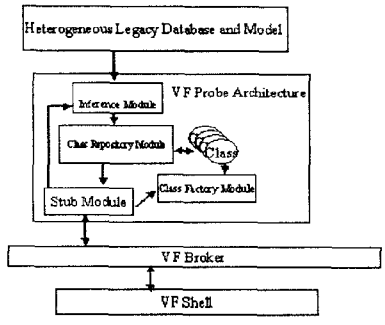

Figure 3: The Virtual Factory Probe

ture of physical system, receiving external command from virtual factory wizard and constructing active object. A virtual factory shell is related to the top-level system, which can be polymorphic parameterized to any variety of virtual factory by virtual factory wizard. A virtual factory broker is the virtual factory production bus. It lets objects transparently make request to and receive responses from objects located locally or remotely. The client is not aware of the mechanism used to communicate with, to activate, or to store the server objects. A virtual factory wizard can be based on enterprise knowledge database and choice of the designer, construct a well-define virtual factory with several interactions. The PPUVF is shown in Fig 2. The components of the PPUVF for each part is described in the following sections.

3.1 Virtual Factory Probe

This is an entity that handles the extracted information from already legacy manufacturing database or a real factory. This part has four functional modules:

Inference Module: This module is directly connected to a real factory, using Prolog-like inference production rule, extracting the data and information, and constructing the corresponding classes which are then sent to class repository module.

Class repository module: This is related to class manager, storing and managing all classes which are sent from inference module.

Class factory module: It can construct the semantic level object instance according to the need of stub module, and perform simulation-based planning and controlling which are linked to real factory.

Stub module: This module provides static interfaces to each service exported, also provides a run-time repository of information about the classes that the probe supports, i.e., the objects that are instantiated. It communicates with the virtual factory wizard in heterogeneous production network with the help of virtual factory broker.

The relationship among the modules in a virtual factory probe is shown in Fig 3.

3.2 Virtual Factory Shell

Initially, a virtual factory shell contains the highest level abstraction classes, such as production performance class, enterprise class, factory class, unit and component class, assembly class, and production control class. The type and implementation of each class can be polymorphically parameterized to object instance with respect to the requirement. The virtual factory shell with instantiated objects can use the service exported by virtual factory probes by querying the interface of virtual factory probe. It acts as the kernel of a virtual factory and also provides the visualization of the whole virtual factory.

3.3 Virtual Factory Broker

Virtual factory allows a client to dynamically build and invoke requests on objects. The client specifies the object to be invoked, the method to be performed, and the set of parameters through a call or a sequence of calls. The client object typically obtains this information from an *interface repository*. Virtual factory broker is responsible for registering the exported classes of virtual factory probe with the interface repository, and routing the call from virtual factory shell to appropriate virtual factory probe.

3.4 Virtual Factory Wizard

A virtual factory wizard can construct a well-defined virtual factory with several interactions based on enterprise knowledge database and choice of the designer. The following introduces the steps of virtual factory wizard for creating a virtual factory.

1. Select the real enterprise area and factory which will be made into virtual factory, and if there are already complete enterprise knowledge database, goto step 3.

2. Start the virtual factory probe for each real factory or enterprise area, and the virtual factory broker in the virtual factory production network, waiting for each virtual factory probe to respond with the status of completion of the probing work.

3. Obtain the service or class description from the interface repository, and adding it into enterprise knowledge database.

4. Make analysis and choose the functionality from enterprise knowledge database or the designer that the virtual factory must contain, and then create the parameter argument list, and sending the parameter argument list to both virtual factory shell and virtual factory probes.

5. Parameterize the classes of virtual factory shell and virtual factory probe, and instantiated objects, and binding the interface between virtual factory shell and virtual factory probe.

6. Generate and manages object reference and broadcast the presence of the object servers so that the virtual factory broker will update the interface repository.

7. Activate the virtual factory shell that uses the service provided by virtual factory probe and enable the simulation-based control which is linked to real factory, and if the virtual factory has not been completed, goto step 3 else closing virtual factory wizard, the creation of virtual factory is done.

4 Application of Virtual Factory

In addition to development of a formal, structured methodology for designing the virtual factory, we must consider the integration of a virtual factory and a real factory to support the reliable estimates of production and optimal process control of manufacturing. The fundamental notion of virtual factory is that it is a computer-based, simulated product development that enables us to "make it virtually" before we "make it for real". Furthermore, we must feedback more information into our virtual factory to allow some modification and, thus, there should be a integration architecture to handle it. For clarity of discussion, a *virtual prototyping* refers to the product and the operation which are generated using simulations of the planned production by virtual factory. On the other hand, a *real prototyping* refers to the actual production processes. Any prototyping is also an object instance with some interfaces. The following introduces a *switching architecture* that integrates both the real factory and the virtual factory, which is shown in Fig 4. Such architecture contains three parts, the following are detailed description of each part:

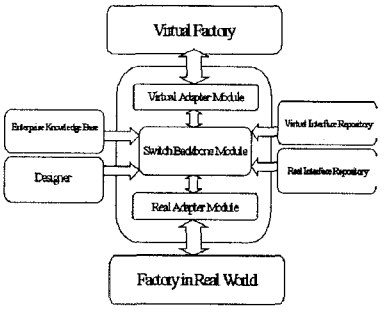

Figure 4: The switching architecture

Switching backbone Module: This module is responsible for switching the virtual prototyping and the real prototyping according to knowledge database or the designer. The information from virtual factory can influence the real process capability, and the information generated by the real word will feedback to the virtual factory. So, a knowledge base must provide the information about the point of time or the parameter, feedback from the real world and forward simulation-based control to the real word. The module shifts the "intelligence" to the data itself. Multi-level and distributed security are also an issue.

Virtual adapter module: This module is linked to virtual factory broker. Firstly, when this module receives the interface request from the virtual factory broker, it will generate a request for the virtual prototyping object interface to the switching backbone module, and wait for its response.

1. If the result returned is virtual prototyping object interface, it will return the result directly to virtual factory broker.
2. If the result returned is real prototyping object interface, it will transform the result to virtual prototyping object interface and send it to virtual factory broker.

Real adapter module: This module is linked to the real factory. When this module receives the interface request from the real factory, it will generate a request for the real prototyping object interface to the switching backbone Module, and wait for its response.

1. If the result returned is real prototyping object interface, it will return the result directly to the real factory.
2. If the result returned is virtual prototyping object interface, it will transform the result

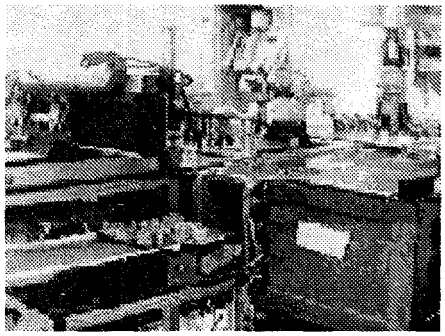

Figure 5: The two-robot assembly cell

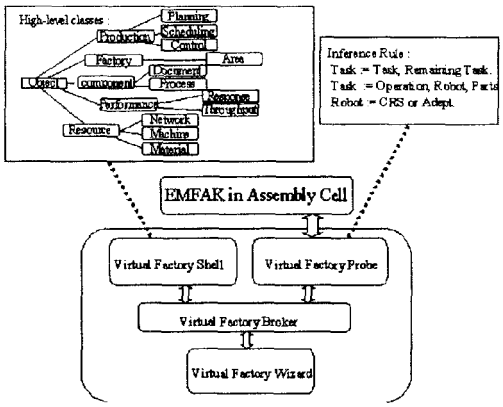

Figure 6: The initial status of PPUVF

to real prototyping object interface and send it to the virtual factory broker.

5 A Simple Example

The example environment in our laboratory is a two-robot assembly cell shown in Fig 5, which is dedicated to assembling various types of mechanical parts serially sent in through a conveyor belt. It is based on the control kernel EMFAK [7] [8] [9], whereby we can easily integrate different pieces of equipment together. Initially, there are some inference rules in the inference module of the virtual factory probe, and some basic highest abstraction classes in the virtual factory shell. Besides, there is an empty interface repository in virtual factory broker, and the whole PPUVF is shown in Fig 6. Next, we proceed with the virtual factory wizard, where the inference module will extract information from EMFAK, generate the corresponding abstract classes, and add them into the class repository module. The stub module will broadcast the interface of the classes into production network, and the virtual factory broker will update its interface repository after the probing is completed, which

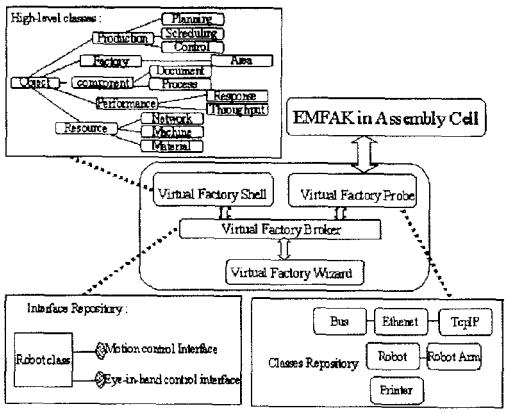

Figure 7: The status of PPUVF after probing

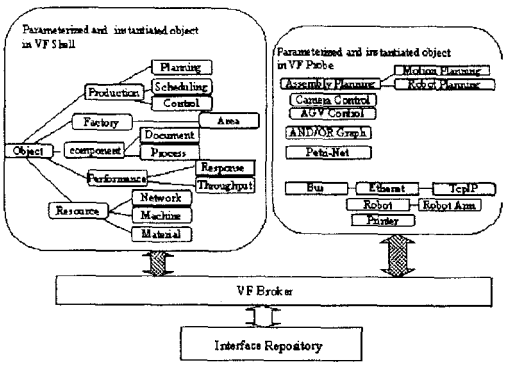

Figure 8: The final virtual assembly cell

is shown in Fig 7. Finally, we parameterize the virtual factory shell and virtual factory probes and binds the interface between them based on enterprise knowledge and choice of the designer with several interactions. Then the final virtual assembly cell is shown in Fig 8. After the creation of virtual assembly cell, under the switching architecture, we link the interface repository of the virtual factory broker in this virtual assembly cell to switching backbone module. Whereas the interfaces provided by EMFAK are also linked to switching backbone module. Then, EMFAK will exchange information with virtual assembly cell, and the virtual assembly cell will get feedback from EMFAK and forward simulation-based control to EMFAK, based on the switching between the interfaces of prototyping object instance(including virtual prototyping object and real prototyping object). The integration of EMFAK and the virtual assembly cell is shown in Fig 9.

6 Conclusion

The vision of virtual factory provides a capability for "Manufacture in Computer", and virtual factory

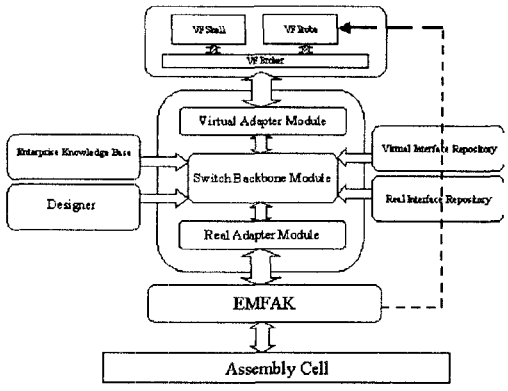

Figure 9: The integration of EMFAK and virtual assembly cell

provides a powerful simulation environment. Such powerful modeling and simulation will cover almost all aspects in the production including production planning, scheduling, accounting and related process. This paper presented the description of a virtual factory, and, from the point of view with object concept, describes the foundation and principles of a virtual factory and its architecture. It also provides a systematic method for building the virtual factory via the polymorphism parameterized universal virtual factory that can be used to extract information to properly model the virtual factory with respect to the real production. Using the object representation, we can represent all types of information, which can be transparently shared among all production softwares. We also consider the requirement of supporting the ability to share objects and integrate processes across geographically distributed enterprises in the production network. Another topic about relationship between the real factory and the virtual factory is much challenge to production manufacturing, and switching architecture is proposed to resolve the interaction between the real and the virtual environment, Based on choice of the designer or enterprise knowledge, we can dynamically make the decision for the type of binding the prototyping object interface, to link real and virtual processes, and to link to the manufacturing control system. By an example presented, we can construct a virtual assembly cell from the real assembly cell, and provide simulation-based control. After this paper is completed, which consists of both theoretical development and practical example, there remain various kinds of work that may improve the presently proposed. They will be left as our future work and are described below.

Verification and measurement : For virtual factory, it refer to the methodologies to support the verification and measurement of the virtual factory.

Friendly visualization : The representation of information to end-user in a meaningful and friendly way, the virtual reality technologies will be included.

Inference rule: The intelligent rules to extract the information from the real manufacturing automatically.

The knowledge database for integration: The information from the virtual factory can influence the real process capability with some encryption method provided by knowledge database, and the information generated by the real words will feedback to the virtual factory. So knowledge base must provide the information about the point of time to feedback from the real world or forward simulation-based control to the real word.

References

[1] S.K. Gupta, Diganta Das, W.C. Regli and Dana S. Nau*Automated Manufacturability Analysis : A survey.* http://www.isr.umd.edu/labs/CIM.

[2] John McGehee, John Hebley and Jack Mahaffey, "The MMST Computer-integrated Manufacturing System Framework," *IEEE Transaction on Semiconductor Manufacturing*, vol. 7, no. 2, pp. 107–116, 1994.

[3] "Virtual Manufacturing User Workshop Final Report," *Virtual Manufacturing User Workshop*, Dayton, Ohio, 12-13, July, 1994.

[4] "Virtual Manufacturing Technical Workshop Final Report," *Virtual Manufacturing Technical Workshop*, Dayton, Ohio, 25-26, October, 1994.

[5] "The Virtual Factory Information Sharing Requirements Framework," http://www.hbs.edu/mis.

[6] Robert Orfali, Dan Harkey, and Jeri Edwards, *The Essential Distribuuted Objects Survival Guide.* John Wiley,Inc., 1987.

[7] T.-S. Huang, L.-C. Fu, and Y.-Y. Chen, "Design and analysis of a dynamic scheduler for a flexbile assembly system," *Proceedings IEEE International Conference on Robotics and Automation*, pp. 3334–3339, 1997.

[8] C.-S. Jann and L.C.Fu, "Flexible control system for robot assembly automation," *Proceedings IEEE International Symposium on Assembly and Task Planning*, pp. 286–292, 1995.

[9] H.-S. Huang, L.-C. Fu, and J. Y. jen Hsu, "Rapid setup of system control in a flexible automated production systems," *Proceedings IEEE International Conference on Robotics and Automation*, pp. 1517–1522, 1997.