Studies on Data Management in Manufacturing Line Monitoring and Control

Hideyuki Takada

Abstract

In recent years, as general purpose computers like personal computers spread into control systems in manufacturing fields, the control systems are expected to be integrated with business systems such as quality management systems, cost management systems and ERP(Enterprise Resource Planning) systems. This integration makes it possible to deliver data rapidly between control systems and business systems, compared with traditional methods like manual inputs and batch based transferring, and is considered to optimize production activities by dissolving information delay and improving data accuracy.

To achieve this integration, frameworks for cooperation and data exchange among applications like COM (Component Object Model) by Microsoft and XML (eXtensible Markup Language) are not enough. In control systems, sensor data are sequenced by the time stamp order for each facility or each equipment, and these data are meaningless as themselves in business systems. That is, a new contrivance is required to reform the sensor data in control systems to useful data in business systems.

For the purpose of effective management of product quality and cost, it is required to provide useful information which is tightly connected with object flow, such as "where this object was passed and how was its status" and "what object is (was) existing in which processes and how was its status." To realize this requirement, the following characteristics of data in manufacturing systems should be handled. The first one is the temporal property. Although temporal database languages like TSQL and TQuel have the ability to deal with temporal aspects, an object itself has a temporal aspect in manufacturing systems. For example, the value field in a sensor object should have a valid value when accessed by an application. The second one is the change of identity and form of objects. Workflow models and some geographical data models handle the change of the object identity. These models, however, cannot handle the change of object form. A manufactured object may exist in one manufacturing process as a point or in some manufacturing processes as a line. The last one is multi-dimensionality of manufacturing result data. Since the concepts of views in object-oriented databases mainly focus on restructuring class hierarchies, they cannot represent the aspects of data in manufacturing systems focusing on one of the time, a manufacturing process and a manufactured object.

In this thesis, the data management models for manufacturing systems are presented. First of all, a temporal object model based on temporal validity is proposed. The model defines the concept of temporal validity to assure the correctness of temporal objects. Secondly, a multidimensional data utilization model for manufacturing management is proposed. The model has the ability of reforming sensor values to multidimensional aspects of the entire plant, each of manufacturing processes and histories of manufactured products. Thirdly, the Time/Place/Object model to represent the organization and the behavior of a manufacturing line is presented. This model enables the integration of the monitoring function and the control function by the ECA mechanism.

These models are incorporated with database middlewares and applied to real plants. The implementation issues are also introduced.

Contents

List of Figures

List of Tables

Acknowledgements

First I would like to thank deeply my supervisor, Professor Yahiko Kambayashi at Kyoto University. He gave me the opportunity to study databases and theories of logic synthesis, and continuously suggested many helpful ideas. He also extensively reviewed this thesis and suggested many improvements.

I would also express my thanks to Professor Toru Ishida, Associate Professor Hiroyuki Tarumi and Associate Professor Hiroki Takakura at Kyoto University. They gave me valuable comments and suggestions for my doctoral research.

I would like to thank Professor Shuzo Yajima at Kansai University. I had studied at his laboratory during the bachelor course of Kyoto University. He also introduced me to Professor Kambayashi for starting my research at the master course.

I would also like to thank Professor Saburo Muroga at University of Illinois. I stayed at his laboratory during the summer in 1992 and studied the theory of logic synthesis.

I would like to thank Dr. Tadatoshi Yamada, Dr. Akira Sugimoto, Dr. Hidekazu Arita, Dr. Satoshi Horiike, Dr. Ichiro Nakahori (now President at Sohatsu System Lab.), Dr. Shigeru Abe, Dr. Morikazu Takegaki, Dr. Hiromitsu Shimakawa and Dr. Koichi Munakata at Mitsubishi Electric Corporation (MELCO) for giving me the opportunity of the research and the chance of studying in the doctor course.

There are many other people I would like to thank. I discussed many ideas with Mr. George Ido, Mr. Ichiro Mizunuma and Mr. Takeshi Nishimura at Industrial Electronics and Systems Lab., MELCO. Mr. Yoshitomo Asano, Mr. Takayuki Mimura, Mr. Masao Ijiri and Mr. Seiji Shitaoka at Kobe Works, MELCO supported me for implementing the database middlewares. Ms. Masako Watanabe (at Kyoto University), Associate Professor Takeo Kunishima, Dr. Shintaro Meki (at Okayama Prefectural University), and Mr. Shigeru Yamashita (at NTT) have been helping me since I was in Kambayashi Laboratory as a master course student.

Finally, I would like to thank all people to help me, especially my family and my friends.

Chapter 1

Introduction

1.1 Recent Trends of Data Management in Manufacturing Systems

Today's manufacturing plants cannot be constructed without the help of computer technologies. The main purpose of introducing computers in manufacturing plants has been to automate the operation of machinery, increase the quality of control and reduce human burdens. In the times, connections between human operators and manufacturing computers were not so tight. However recent social requirements to products such as high quality, low cost and environmental consideration have begun to bring the traditional framework of manufacturing systems to the limit of evolution.

To meet the severe requirements of current society, it is needed to take human roles into the framework of the system. The system should provide manufacturing data to help making decisions by plant operators and enterprise managers. This means that the purpose of plant data changes from the traditional manufacturing systems. That is, although plant data has been utilized only for control and production records, today's systems are required to have the ability of more effective utilization of plant data.

The key issue for effective utilization of manufacturing data is how to retrieve useful data from production records. Production records include sensed process values such as temperature, humidity and density. These data can be acquired to computers through process I/O units periodically and stored to databases after some operations.

Traditional manufacturing management systems are built on proprietary platforms based on vendor's specific technologies. However, manufacturing systems are organized by some layers ranging from field equipment to enterprise business management and each vendor has both of good part and poor part. As a result, layers of a manufacturing system have been built by many proprietary platforms and this led to disconnectivity between different layers.

This situation began to change in early '90s. Small desktop workstations based on UNIX and even personal computers with Microsoft Windows began to be introduced to manufacturing systems. These platforms have standard interfaces to interact with other parts of a system. So the requirement of integrating system layers were increasing slowly.

This requirement blowed up in late '90s along with the spread of the Internet environment. Scalability of the Internet makes it possible to communicate between enterprise level systems and manufacturing field systems. In recent years, especially, ERP (Enterprise Resource Planning) systems[15] has begun to be integrated with manufacturing field systems. By this integration, the speed of communication from manufacturing fields to enterprise decision makers becomes rapid, compared with conventional batch reporting or manual inputs. It is considered that this integration improves the efficiency of production activities.

1.2 Requirements from Practical Usage

1.2.1 Difference of Data Utilization in System Layers

In recent years, many frameworks to integrate system components has been proposed. COM (Component Object Model) makes it possible to communicate among objects consisting of different applications. XML (eXtensible Markup Language) is a framework of flexible data exchange between system components, based on the concept of semi-structured data. These frameworks, however, provide only a mechanism for integrating system components. It is not enough to integrate system layers of a manufacturing line, because the way of data utilization is different at each level of the system. For example, raw sensor values for each of manufacturing processes or their aggregated data are meaningless for enterprise level systems. That is, it is required to provide a model and a mechanism to change the form of data from raw sensor values in manufacturing fields to useful information for business level systems.

For the purpose of effective management of product quality and cost, it is required to provide useful information which is tightly connected with object flow, such as "where this object was passed and how was its status" and "what object is (was) existing in which processes and how was its status." It is a very time consuming task for operators to retrieve this kind of information from raw sensor values.

1.2.2 Traceability of Production Records

In manufacturing systems, two kinds of data are acquired. One is sensor values which are acquired periodically. The other is trace histories of manufactured objects which are acquired eventually. Sensor values can be stored in a database as a tuple of values and an acquired time stamp. An object trace history can be represented by time stamps when the object enters and leaves each manufacturing process.

When a defect is found in a product, its cause needs to be detected for quality assurance. The cause may be abnormality of temperature caused by a problem of production parameter settings for a certain process. In this case, two types of data presented above should be provided in the integrated form. That is, sensor values for each of processes which the product passed through need to be retrieved. In order to realize this functionality, a database system must have the ability of handling temporal aspects of data, reforming stored data according to the requirement of data utilization and tracing manufactured objects.

1.2.3 Integration with Control Activity

Manufacturing management systems have to be connected with control systems of a real target plant. Short period feedback controls are performed by the controllers layer, but parameters for controls need to be issued from the manufacturing management system layer. This means that appropriate commands have to be transferred to controllers according to the plant status such that which lot of product is existing in which manufacturing process.

Product manufacturing is achieved according to production recipes which are planned based on production requirements such as order reception and demand estimation. Production recipes include production specifications such as the amount of raw materials, color and size of final products. Manufacturing management systems have to issue these specifications to controllers.

1.3 Conventional Database Approaches

In this section, several approaches on database technologies are discussed for realizing manufacturing management systems.

1.3.1 Temporal Databases

In order to deal with the change of the state of manufactured objects, the concept of time should be introduced to a database system. Basically, a temporal database is a kind of the relational database extended to have time attributes representing the valid time interval of each tuple. In traditional relational databases, a tuple is overwritten when updated. In temporal databases, a tuple is not overwritten when it is updated, but a new tuple is inserted after marking the valid time interval to the updated old tuple. Users can retrieve the *past* tuple by specifying a time clause in a query. Several languages like TSQL and TQuel extending the SQL for relational databases have been proposed[26][27][32]. For example, TSQL has the 'WHEN' clause which can be used as the following query.

```
SELECT salr
FROM S, M
WHERE S.eno = M.eno AND M.eno = 125
                  AND M.mgr = 'Smith'
WHEN S.INTERVAL OVERLAP M.INTERVAL
```

This query finds the salary of employee 125 when Smith was his manager.

These frameworks, however, are not enough for manufacturing information management, because of the following reasons.

- In a relational temporal model, each tuple represents a fact which is valid during the time interval of the tuple. An example of attributes of a tuple can be a person's name and the person's salary with a valid time interval. It is not appropriate to represent a history of a manufactured object by a sequence of facts, because the fact cannot handle the identity of objects.

- The identity of manufactured objects changes during a manufacturing process. One object may be split into multiple objects. The relational temporal model cannot handle the change of the identity.

Several object-oriented temporal databases are also proposed. But these models only handle the versions of objects. The change of identity of manufactured objects is different from the change of versions in the following points.

- Even if an object branches to two versions of the object, the class of the two objects is identical. On the contrary, if a manufactured object is split into two objects, the class of one object can be different from the class of the other object.

- When merging two versions of objects, the class of the two objects is also identical. On the other hand, the class of merged two manufactured objects can be the same or different from each other.

1.3.2 Views in Object-Oriented Databases

Views in databases are aimed to make the structure of a database independent from application usage. Data structure in relational databases is designed considering the elimination of data redundancy. Applications can re-structure the stored data for satisfying application specific usage by utilizing relational operations such as projection, selection, join and aggregation.

Views in object-oriented database[3][24] are mainly based on re-structuring class hierarchies. A new class can be defined by projecting and hiding some at-

tributes from some classes, or selecting some instances satisfying specified conditions. For example, a new class *Adult* can be defined by selecting instances such that their *Age* attribute value is over 21, as the following manner.

```
class Adult includes
    (select P from Person where P.Age >= 21)
```

For the purpose of tracking manufacturing histories of a product, these view mechanisms are not enough. Since a manufactured object changes the identity during a manufacturing process, a history of production is the combination of some objects which are linked each other. A mechanism for generating views for some objects which are linked each other should be provided.

In addition, the object-oriented data model cannot deal with the object life cycles of the real world. In object-oriented databases, objects are persistent, so all data in databases are regarded as *alive*. When applying to manufacturing systems, another aspect of object life should be considered. That is, the life of an object starts when it enters a manufacturing line and ends when it leaves, but the object including production records is not deleted from a database after leaving a manufacturing line.

For the above reasons, a new concept of view and life cycle needs to be considered for manufacturing data management.

1.3.3 Workflow Models

A manufacturing line involves a kind of workflow like business document processing. A pioneering work for workflow management of documents can be found in [12]. This model can handle branching and merging of documents. Other workflow models like [4][14][19][23] also aim to flexible management of processed work units.

These models, however, are focused on the business workflow, so there are some insufficient points for manufacturing management. The most decisive point is the ability of control to the real world. The workflow systems mainly contribute to monitoring the flow of work units, while it is required to control manufacturing machines in manufacturing lines. That is, control functions to issue appropriate control commands to a real world are required as well as monitoring functions to trace manufactured objects.

Another different point is the change of identity of work units. When a work unit branches, some different kinds of works are issued in workflow systems. But in a manufacturing line, some objects of the same kind may be produced. For example, a unit of liquid in a tank may be divided into some subdivided units for smaller tanks. That is, objects divided from an object may be processed in the same processes afterwards. This kind of branching may not happen and cannot be handled in the business workflow model.

1.3.4 Other Related Works

In the geographical information system field, a model to manage the change of object identity is proposed in [11]. This model defines various operations to represent the change of identity of geographical objects such as states, cities and countries. The operations defined in this model are very powerful, so change of the identity of objects in a manufacturing line can be represented. However, this model cannot handle the movement of objects, which is indispensable for manufacturing management. In geographical information management, the identity of objects may change but an object itself don't move. On the contrary, the change of object identity is involved with the movement of objects.

The multi-dimensional databases[9][10] and OLAP (OnLine Analytical Processing) systems[5][20] deal with various aspects of stored data and very useful

for analytical usage. The multi-dimensional operation is based on aggregating values focusing on one attribute. For the manufacturing management, however, a production history cannot be obtained by aggregating sensor values and object movements. Sensor values have to be retrieved according to the time interval when the object existed in the target process.

1.4 Approaches for Effective Manufacturing Information Management

Based on the above discussions, the author has been tried to build a data management model by the following approaches for effective manufacturing information management. The positioning of each approach is shown in Fig.1.1.

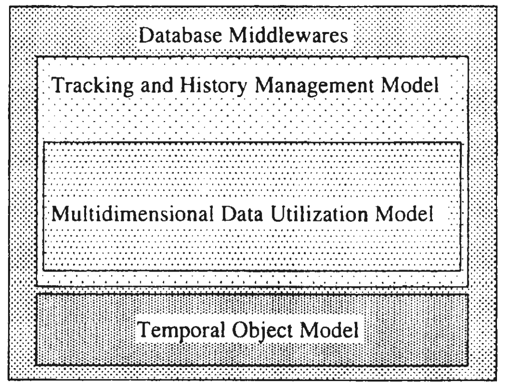

Database Middlewares
Tracking and History Management Model
Multidimensional Data Utilization Model
Temporal Object Model

Figure 1.1: Positioning of the Approaches

1.4.1 An Object Model for Handling the Temporal Property

In a plant monitoring and control system, an object should correspond to a physical target like a sensor rather than each sensed value. For example, it is useful for an application to be accessible to sensed values by reading a value field of a sensor object, including other static sensor information such as a photograph and a type number. This enables to integrate the process control information with facility information.

To provide such an object which integrates the control information with facility information, it is required to deal with the temporal aspects of the object. The value field of the object changes as time goes on. The status of the value field should be valid when an application reads the contents of it.

This requirement can be replaced with how to define the semantics of validity of an object with the temporal property. A temporal object model based on the temporal validity presented in Chapter 3 defines a valid interval to represent the semantics of validity. Each temporal object has its valid interval and the validity of a value of the temporal field is evaluated by the valid interval. Evaluation methods are different depending on the temporal properties. The model categorizes the temporal properties into three types: real-time objects, trend objects and event objects.

1.4.2 Multidimensional Data Utilization Model

Acquired data from a manufacturing line should be utilized in various aspects for different purposes. For example, the result of entire flow of products is useful for production scheduling. Sequence of sensed values of each production unit for a certain manufacturing unit can be utilized to analyze the cause of parameter setting and result of products. Sequence of sensed values for each of processes which a production unit passed through helps the detection of the cause of defect in a product.

To support the data utilization by these aspects, multidimensional data utilization model should be defined and a mechanism to realize this function has to be

designed. The multidimensional view model presented in Chapter 4 defines three aspects of data: the plant view, the unit view and the batch view.

1.4.3 A Tracking and History Management Model

In order to achieve manufacturing execution, each production unit needs to be traced in a plant and appropriate commands are required to be transferred to set the manufacturing process. For example, before a steel plate enters a mill stand, rolling thickness needs to be set, according to a production recipe.

To realize this function, the organization of a manufacturing line and the behavior of materials need to be represented and a system has to trace a manufactured object based on the definition and send appropriate commands to controllers.

The Time/Place/Object model presented in Chapter 5 defines a framework of representing the organization of a manufacturing line and the behavior of materials. In order to handle the change of object identity, five primitives of places are defined according to their functions. Three models of object forms are also defined to represent the difference of movement patterns.

In addition, the multidimensional view model is generalized based on the mapping operations among aspects of time, place and object.

1.4.4 Database Middlewares

The above models should be incorporated with basic database functions. For manufacturing management systems, both of solidity and high functionality are required. In particular, high functionality cannot interfere real-time processing capability.

In order to satisfy these two characteristics, database management systems with a reduced set of functions in commercial business database systems are expected. Furthermore, an appropriate architecture for manufacturing management databases should be designed.

Chapter 2

Backgrounds

2.1 Structure of Plant Monitoring and Control Systems

First of all, a physical structure of plant monitoring and control systems is described. Fig. 2.1 shows a typical example. In the bottom level, a lot of sensors and actuators are equipped in plant facilities. Signals sensed by sensors and driving actuators are handled by process I/O units equipped with controllers. Each controller issues drive signals to actuators according to sensed signals and operations taken by plant operators on operator stations. Controllers and operator stations are connected by a control network. A control network guarantees real-time communication among connected components by the token ring and distributed shared memory mechanism.

Industrial computers perform the tasks of database management for operator stations and business management systems. Operator stations need to provide not only the present status of the plant, but also trend histories. In order to provide the histories, the industrial computer acquires sensed values from the control network and stores them in a database.

Furthermore, the industrial computer needs to manage production recipes issued by business management systems. The business management system schedules the production for each lot according to several conditions such as market demand or available resource status. After the planned schedules are committed, production recipes consisting of production parameters such as the amount of raw materials and equipment settings are downloaded to the industrial computers. The industrial computer traces each lot in the plant according to events detected by controllers and issues appropriate parameters to controllers.

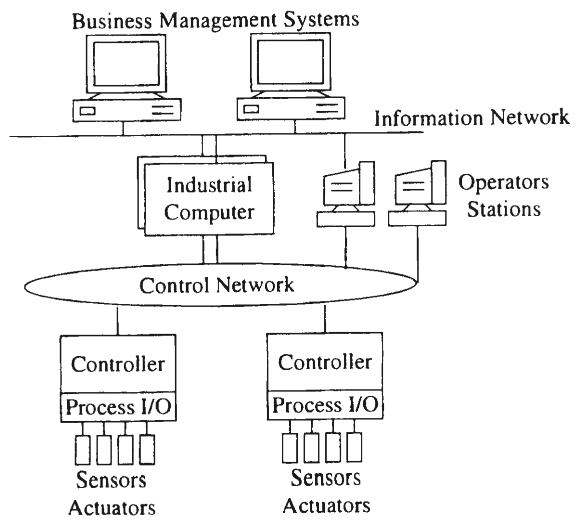


Figure 2.1: A Typical Structure of Plant Monitoring and Control Systems

As described above, the industrial computers have to provide database management functionalities with operator stations and business management systems. In order to build database management functions, an appropriate data model for plant monitoring and control applications is required. In the next section, the characteristics of manufacturing processes is described.

2.2 Characteristics of Manufacturing Processes

2.2.1 Classification of Manufacturing Processes

In the standard on batch control (ANSI/ISA-S88.01-1995)[1], industrial manufacturing processes are classified into the following three categories depending on how the output from the process appears.

Continuous processes Materials passes in a continuous flow through processing equipment.

Discrete parts manufacturing processes A specified quantity of product moves as a unit between workstations, and each part maintains its unique identity.

Batch processes Finite quantities of materials (batches) are produced by subjecting quantities of input materials to a defined order of processing actions using one or more pieces of equipment.

Continuous processes include gas, oil or water treatment plants, while discrete parts manufacturing processes include electronics assembling and car manufacturing lines. Batch processes have characteristics of both. For example, in beer plants, material flow from a tank to another tank is continuous, but a material moves as an identical lot. In other words, batch processes can be seen as continuous processes from micro view, but they can be seen as discrete processes from macro view.

2.2.2 Production Information Management of Manufacturing Processes

In continuous processes, produced materials have no identity since they flow continuously. Therefore, production information can be obtained from sensed values of sensors equipped with plant facilities. These values are acquired periodically. For example, the amount of liquid flow on a pipe is measured by a flow meter and acquired every 10 seconds. Other acquired values can be temperature, humidity, liquid level on a tank and so on.

In discrete parts manufacturing processes, all materials have an identity such as a lot number. Production information focuses on parts and their combinations rather than manufacturing processes. Identity of materials branches and merges as production goes on.

In batch processes, it is needed to handle both of produced materials and manufacturing processes. Produced materials have their identities like discrete parts manufacturing processes, while production information is obtained from sensed values. This means that periodically acquired sensed values have to be managed integratedly with identities of produced materials.

The most important issue in batch manufacturing information management is the traceability of batches and materials. Fig. 2.2 illustrates this issue. Materials and batches are processes by ordered pieces of equipment. Movement of batches are not straight forward. A batch may be divided into several batches or some batches may be mixed. If a defect is detected in a product, causes of the defect should be found and eliminated.

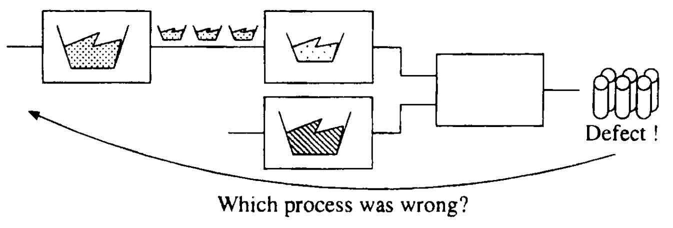


Figure 2.2: Traceability of Materials and Batches

In addition to the above issue, production recipes need to be managed. A

recipe includes the necessary information to execute the production, such as the amount of ingredients, process values such as temperature and density. Each recipe corresponds to each manufactured unit. As production goes on, control commands have to be transfered to controllers according to the contents of the corresponding recipe.

2.3 Integration of Control Systems with Enterprise Systems

The standard on enterprise and control integration model, ISA SP95[2], defines a system hierarchy of manufacturing systems as illustrated in Fig.2.3.

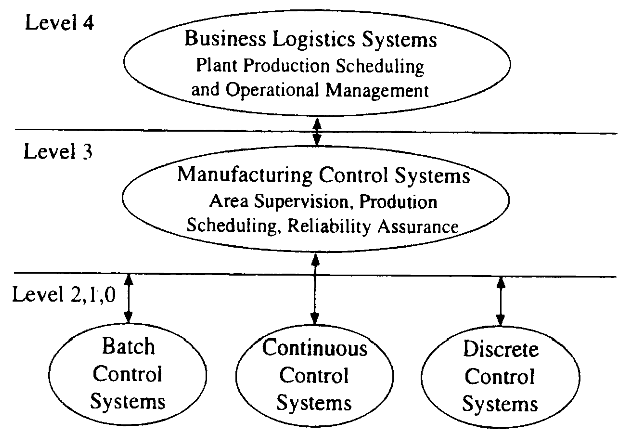

Level 4
Business Logistics Systems
Plant Production Scheduling
and Operational Management
Level 3
Manufacturing Control Systems
Area Supervision, Prodution
Scheduling, Reliability Assurance
Level 2,1,0
Batch Control Systems
Continuous Control Systems
Discrete Control Systems

Figure 2.3: System Hierarchy of a Manufacturing System

The standard defines the interface to interact between the business logistics systems (level 4) and manufacturing control systems (level 3), while the industrial computer takes a role of the level 3 and level 2. Each level of the layer performs the following activities.

- Level 4

- Production scheduling for a long span.
- Plant coordination and operational data reporting for a plant level.

- Level 3
  - Production scheduling for a short span according to the schedule established by the level 4 activities.
  - System coordination and operational data reporting for a line level.
- Level 2
  - Execution of manufacturing based on the schedule established by the level 3 activities.
  - Macro level (process level) tracking of materials.
  - Production data reporting for each production unit.
- Level 1
  - Feedback loop controls for sensors and actuators.
  - Micro level (sequence level) tracking of materials.
- Level 0
  - Control tasks taken by sensors and actuators with some intelligence.

In order to realize the level 3 and level 2 functions, database functions for reforming the data between enterprise level and control level as well as storing recipes and production results are required.

2.4 Basic Models for Handling Temporal Data

2.4.1 Temporal Consistency

In temporal databases, temporal consistency has to be satisfied in addition to the logical consistency. Logical consistency can be satisfied by scheduling the execution of concurrent transactions to satisfy the serializability. Temporal consistency requires the validity of data on the time axis. In [22], temporal consistency is defined as follows.

Absolute consistency Let CT be the current time. For data d whose time stamp is d_{vt} and absolute time interval is $avi(d)$, the data d satisfies absolute consistency if and only if:

$$(CT - d_{vt}) \leq avi(d)$$

Relative consistency For a data set R, the retrieved data from R satisfies relative consistency if and only if:

$$^{\forall}d, d' \in R, |d_{vd} - d'_{vd}| \leq rvi(R)$$

where d_{vd} is a time stamp of data d and $rvi(R)$ is relative valid interval of a data set R.

The absolute consistency means that the data should be *fresh* enough. The *freshness* for data d is represented by $avi(d)$. On the other hand, the relative consistency means that when data is derived from a set of data, deriving data in the set should be *similar age*. The similarity of a data set R is represented by $rvi(d)$.

2.4.2 Semantics of a Time Stamp

Semantics of a time stamp is generally discussed in the studies [21]. There are two types of meaning defined as follows.

Valid time: The time stamp is regarded as the time when the fact gets true in the real world.

Transaction time: The time stamp is regarded as the time when the fact is acquired in the system.

In general database applications, these two types have a significant difference. But in manufacturing management systems, the time stamp based on the valid time is difficult to realize, because sensors have no clock to associate values with a time stamp. Therefore transaction time is regarded as valid time by assuming that sensor values on a control network are current and acquired data is stored in a database in an enough short period.

2.4.3 Semantics of Now

In temporal databases, the variable "Now" is used to represent the current time, but this includes a lot of ambiguity. The semantics of "Now" is discussed in [28] in detail. The ambiguity originates in using the variable "Now" in the valid time interval of a tuple. On the contrary, the variable "Now" is not stored in manufacturing management databases. Instead, the value is evaluated by the absolute consistency described above.

Chapter 3

A Temporal Object Model Based on Temporal Validity

3.1 Temporal Properties of Objects in Plant Control Systems

In a plant control system, there are two types of data to be handled. One type of data is *static* data whose value doesn't change so frequently, such as equipment information. The other type of data is *dynamic* data whose value changes depending on the status of the plant. An example of the latter type is a sensed value of water temperature in a tank. Although there is such a difference, these two types of data are in connection with each other. For example, information about a tank such as a name and the location can be the *static* data, while temperature or level of water in the tank can be the *dynamic* data. In order to build a plant monitoring application which has the capability such that when an user clicks a tank in a plant map then the graph of temperature and level of water in the tank will be shown, these two types of data need to be managed integratedly.

Basically, dynamically changing data is acquired through a sensor and stored to a database periodically. A schema for storing the data is a relation including a time stamp attribute and values for data items, as shown in Fig. 3.1. Every time the data is acquired, a new tuple is inserted to the relation. In usual implementations, the maximum number of tuples are fixed and the oldest tuple is deleted before a new tuple is inserted, since a plant is operating all of the day and the size of the table monotonously increases.

Time stamp	item 1	item 2	item 3	......	item 8
2000/09/19 14:10:00	121	65	75		218
2000/09/19 14:10:05	125	63	78		220
2000/09/19 14:10:10	127	60	74		217
2000/09/19 14:10:15	129	62	72		220
:	:	:	:	......	:

Figure 3.1: A Schema Representing Dynamically Changing Data

As described above, this kind of dynamically changing data needs to be managed integratedly with the static data. To achieve this requirement, there are two technical issues to be solved. The first one is on the architecture to integrate the dynamic data handling with static data handling. Since dynamic data handling requires real-time capability, it should not be interfered by other heavy tasks in static data handling. The second one is on an object model which has the capability of representing dynamically changing data as an object.

In the succeeding sections, these two issues are discussed.

3.2 A Database Architecture to Integrate Control Systems and Management Systems

A plant system is organized by a control system and a management system. In a control system, computers, controllers, sensors and actuators are connected each other by a real-time network. A real-time network is realized by a kind of a distributed shared memory called the reflective memory in order to assure real-time communication. Each node connected to the reflective memory has a memory representing the network space and the write access to a memory of a node is reflected to the memories of all other nodes by a hardware mechanism. On the other hand, in a management system, business applications such as production management, quality management and operation management are realized in workstations, personal computers and mainframes. These systems are connected by a general purpose network like the Ethernet. In current plant systems, computers and controllers in a control system are also connected to the network in a management system to communicate each other.

A Database system in a plant system has two roles. One is data acquisition and the other is data provision. The function of data acquisition has to read the contents of the reflected memory and store them to disks with real-time constraints. Real-time constraints mean that tasks should be completed by a deadline. On the other hand, the function of data provision has to re-organize the acquired data in order to provide comprehensible representation with plant operators. It is difficult to realize these two functions in one database system because data provision tasks may interfere the real-time tasks of data acquisition. For this reason, data transfer between a control system and a management system has been achieved by a gateway which performs batch based communication. As a result, a management system cannot use 'fresh' data in a control system. Furthermore, data transferring mechanism should be realized independently in both systems for each application.

In order to solve the problems above, a database system architecture to integrate a control system and a management system should be considered. Data in both systems can be accessed by the same model in the integrated architecture. In addition, a management system can access 'fresh' data in a control system using this architecture.

3.2.1 Two Layered Architecture

In order to integrate these two systems, a two layered architecture is proposed as shown in Fig. 3.2. To prevent the business application tasks from interfering time-critical data acquisition, a kind of buffers is required. The two layered architecture is realized by 'Real-Time Data Server (RTDS)' which acquires plant data and 'Real-Time View Server (RTVS)' which provides the acquired data with applications. These two servers are interconnected by the ring buffer mechanism and temporal objects with temporal validity.

Structures and functions of these servers are described below.

3.2.2 Real-Time Data Server

Since the detail structure and functions of the RTDS[29][30][31] are out of scope of this thesis, only an outline is presented. The RTDS has the following data provision functions.

- Periodically transferring the latest acquired data. The data has a time stamp which represents the time when the data is acquired.
- Retrieving stored data according to a specified time interval.
- Reporting events detected from acquired data.

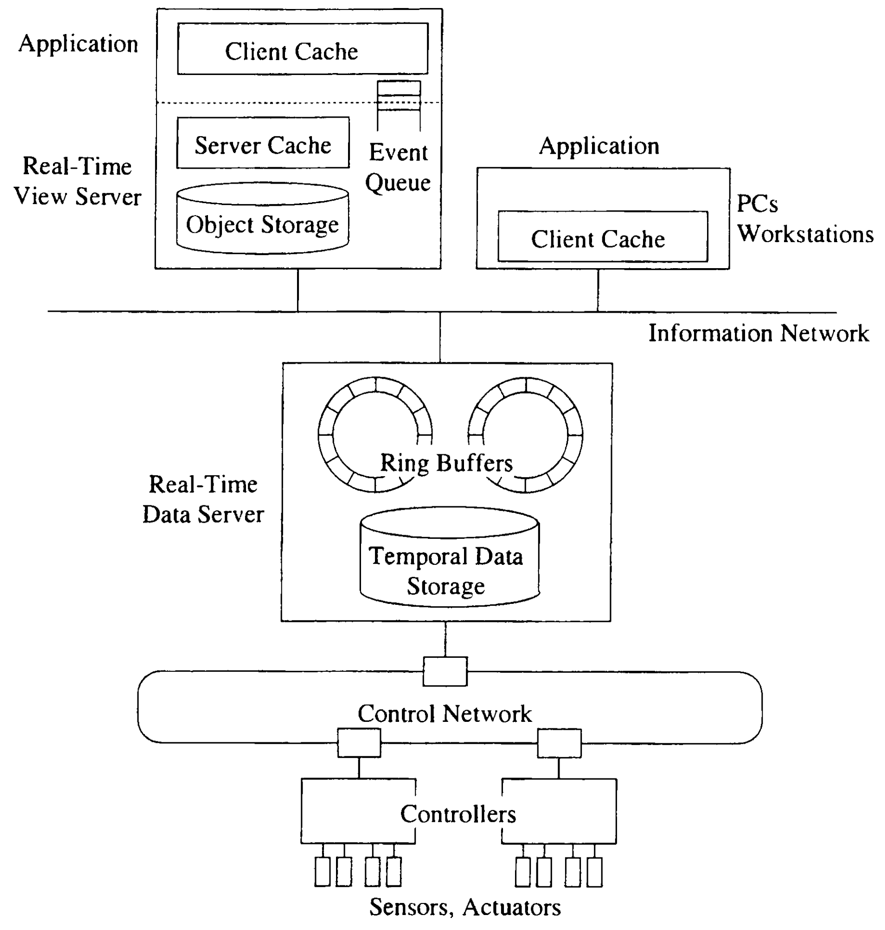


Figure 3.2: Two Layered Architecture for Plant Monitoring Database Systems

The RTDS is realized by the multi-processes and multi-threads architecture on a kind of UNIX with real-time extension. There are the following two major processes in the RTDS.

Acquisition Process One thread of the acquisition process periodically reads a set of data from the control network, gives a present time stamp and writes them to a ring buffer on a main memory. It also checks the contents of acquired data based on pre-defined conditions and detects the events. Another thread of the acquisition process stores data on the ring buffer to a temporal data storage on a disk. The acquisition process exists for each of data acquisition periods.

Provision Process There are two threads in the provision process. One transfers to clients the latest acquired data periodically or the detected events. The other receives requests of retrieving data from clients and provides the retrieval results.

Execution of these processes and threads are scheduled based on the "Rate Monotonic Scheduling Analysis"[18] which assigns higher priority to tasks of shorter execution periods. In addition, although the ring buffers are shared by the acquisition process and the provision process, the mutual exclusion among these processes are achieved by the priority inheritance protocol. By these mechanisms, the real-time processing of the RTDS is assured.

3.2.3 Real-Time View Server

The RTVS is a database server which has the capabilities of handling plant data with temporal aspects in the RTDS as objects virtually, in addition to the general persistent object management. In this thesis, this virtual object for handling plant data is called a 'temporal object.' The RTVS is realized by the multi-process architecture on a standard UNIX operating system. The RTVS has the following functions and mechanisms.

- The RTVS has a disk which stores persistent objects and meta-data such as class definitions. The contents of temporal objects are not stored in the disk of the RTVS and only the type definition is managed, because the contents are transfered from the RTDS. In the RTVS, temporal objects are held in the server cache and the client cache described later.

- The RTVS has the server cache where the recently accessed persistent objects and temporal objects are placed. The RTVS has also the client cache which improves the performance of multiple accesses to the same objects in one transaction. Applications access to objects through the client cache. Objects copied to the client cache from the server cache are locked to assure the serializability among multiple transactions. When a transaction commits, updated objects in the client cache are written to the server cache and all locks are released.
- Temporal validity of temporal objects in the client cache is maintained by the concept of the *valid interval*. Details are presented in 3.3.
- In addition to links among persistent objects, links to temporal objects are supported. A link to a temporal object is called a *temporal link* which has the parameters for temporal validation. Details are presented in 3.3.4.
- The RTVS has the event queue to pass the detected events to applications. Transactions which are invoked according to the event types can be registered. Details are presented in 3.3.5.

In this architecture, applications access to objects through the client cache. Therefore, objects in the client cache have to be in the valid status which applications expect. The status of objects in the client cache is managed by the concept of temporal validity proposed in the following section.

3.3 Definition of Temporal Objects and Their Temporal Validity

A temporal object is defined as an object which has the temporal property. A persistent object in conventional object-oriented database systems represents only the present status of the target defined as an object, therefore no temporal aspects exist. However, in case of representing an instrument in a plant as an object, the history of status changes of the object has significant meanings.

In this thesis, temporal objects are classified into the following three types according to the characteristics of them.

Real-time Object: The present status of a target.

Trend Object: History of status of a target.

Event Object: Event occurrence of a target.

Concepts of each temporal object are shown in Fig.3.3. Temporal objects have to be in valid status from the view point of temporal requirements of applications.

As described before, the RTVS obtains temporal data from the RTDS and provides them as encapsulated objects. Since the objects are placed on the client cache, the status of the cached objects has to be temporally valid. For example, if an object has an attribute representing a sensed temperature, the object has to hold the correct value of the present temperature. This correctness of an object is called *temporal validity* and the procedure for making the object valid status is called *temporal validation*. Definitions of temporal validity and descriptions of temporal validation for each type of temporal objects are presented below.

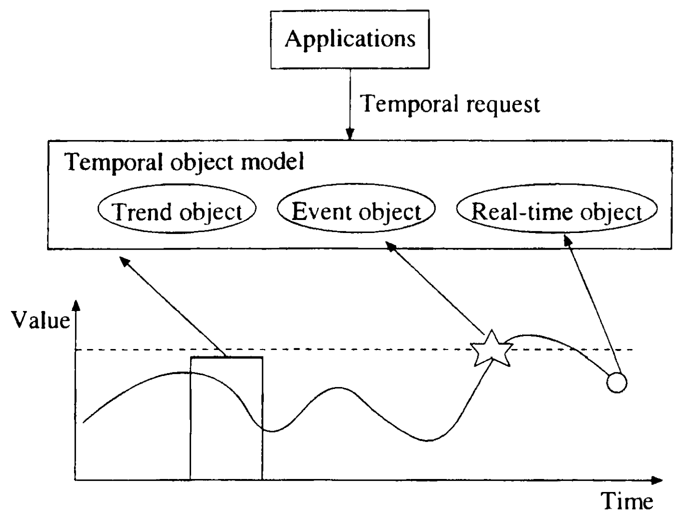


Figure 3.3: A Concept of the Temporal Objects

3.3.1 Real-time Objects

Real-time objects represent the *present* status of their targets such as temperature of a boiling tank. However, it is not realistic to assure that the state of the object strictly represents the present status of the target, because the state of the target changes continuously. In order to define the semantics of *present* state of the target, a concept of *valid interval* is introduced.

Definition 3.1 *Temporal validity of real-time objects*
A time stamp t and a valid interval t_v are associated with each of real-time objects stored on the client cache. If present time stamp t_p satisfies $t_p < t + t_v$, the real-time object is temporally valid.

The procedure for temporal validation is achieved when an application requests a value of a real-time object. Temporal validation is achieved in the following steps.

Step 1: Examine whether the real-time object is temporally valid or not, according to the definition described above.

Step 2: If the real-time object is temporally valid, exit the procedure.

Step 3: If the real-time object is not temporally valid, the newest value is requested to the server and the value of real-time object on the client cache is updated.

3.3.2 Trend Objects

A trend object represents a series of the status of the target for time points at even periods in a time interval, such as temperature values of a boiling tank from 12:00 to 13:00 for every 1 minutes.

Definition 3.2 *Temporal validity of trend objects*
Start time point t_s, end time point t_e, period p and valid interval t_v are associated with each of trend objects. A trend object stores values for time points $\{t_s, t_s + p, t_s + 2p, ..., t_e\}$. When start time point t_{s_v}, end time point t_{e_v} and p_v are specified as validation parameters, if each time point $t_i = t_{s_v} + i \cdot p_v (i = 0, 1, 2, \cdots, \lfloor (t_{e_v} - t_{s_v})/p_v) \rfloor$ satisfies $t_s + i \cdot p < t_i < (t_s + i \cdot p) + t_v$, the temporal object is temporally valid.

A temporal validation procedure for trend objects is achieved by the following steps when trend values are requested.

Step 1: Examine if the trend object is temporally valid based on the above definition.

Step 2: If valid, exit the procedure.

Step 3: If not valid, values for each time points specified by validation parameters are requested to the server.

Examples of temporal validation parameters which does and doesn't satisfy the temporal validity are shown in Fig.3.4. Utilizing this mechanism, when trend values from 10:00:00 to 10:10:00 for every 10 seconds with 5 seconds of valid interval are stored in the client cache, the trend object is temporally valid for a request with validation parameters such as start time point 10:00:02, end time point 10:10:02 and period 10 seconds. There is no need to communicate between a client and a server in this case.

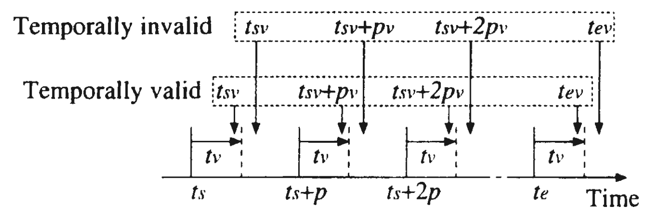


Figure 3.4: Temporal Validity of the Trend Object

3.3.3 Event Objects

An event object reports the change of status of the target with associated data items such as a lot number of a thrown raw material. Event objects are also accessed via a client cache from applications. When an application accesses an event object, the application should be informed whether the event object includes a new event occurrence or not. Therefore an execution control mechanism is required to decide whether the application can read the contents of the event object or not. When an application is not permitted to read the event object, it should be blocked or informed an error. This is also handled by the temporal validity for event objects.

Definition 3.3 *Temporal validity of event objects*

Let event object Oe handle a subset E_s of a set of events E which occurs in a target plant. When an event $e \in E_s$ exists in a event queue and e can be stored to an event object, the event object satisfies the temporal validity. If an event $e \in E_s$ doesn't exist in an event queue, the event object doesn't satisfy the temporal validity.

An event queue is a buffer to store events in the arrival order as shown in Fig.3.2. The temporal validation procedure for event objects is achieved by the following steps.

Step 1: Examine if the event object is temporally valid based on the above definition.

Step 2: If valid, exit the procedure.

Step 3: If not valid, the procedure is blocked until the event object becomes valid or an error is returned to an application. Which way chosen can be specified by a parameter of the validation procedure according to application's policy.

This mechanisms can be used to realize an event-driven application such as invocation of ECA rules.

3.3.4 Temporal Link

In general object-oriented database management systems, a link to an object is represented by an object identifier of the linked object. When a link is navigated, an object identifier associated with the linked object is evaluated and the linked object is retrieved. For temporal objects, a link to a temporal object can also be handled and the linked temporal object can be represented by an object identifier. However, when a link to a temporal object is navigated, the target temporal object

has to satisfy the temporal validity. Therefore the temporal validation procedure has to be included in a method for navigating links.

To support the navigation function to temporal objects, the *temporal link* is provided. When the temporal link is navigated, a linked temporal object is validated to satisfy the temporal validity.

The temporal link is classified into the following three types according to the type of a linked temporal object.

- Real-time link
  Temporal link to a real-time object.
- Trend link
  Temporal link to a trend object. The trend link has the following three parameters for the temporal validation procedure: the start time point t_{s_v} and the end time point t_{e_v} of the time interval and the time period p_v.
- Event link
  Temporal link to an event object.

Utilizing the temporal link, informations such as product data or facility data represented by persistent objects can be linked with control informations such as process values.

3.3.5 Active Mechanism

In addition to the passive function for applications to read the status change in a plant, the event object has the capability of realizing the active mechanism to invoke the pre-registered methods according to the type of the status change. The active mechanism is also called the ECA mechanism, that is, when an '*event*' occurs, if a '*condition*' is satisfied, then an '*action*' is taken[16].

The active mechanism supported by this temporal object model corresponds to the following sequences.

1. Interruption by the periodical timer for invoking data acquisition thread is regarded as an *event*.
2. The acquired data is checked whether it satisfies an pre-defined inequality or not. The inequality is regarded as a *condition*.
3. A transaction associated with an event is regarded as an *action*.

Note that the execution model is categorized to the separate mode[8] and these transactions are executed on the different CPUs, since the triggering transaction corresponds to the data acquisition thread in the RTDS while the triggered transaction corresponds to an application method running on the RTVS. Although a transaction on the RTVS may request to start the data provision thread, the data provision process never interferes the data acquisition process by the benefit of the RTDS architecture. In this way, the non real-time transactions running on the RTVS don't affect the real-time tasks in the RTDS. In addition, the event queue for transferring events from the RTDS to the RTVS takes a role of a buffer, considering that execution of transactions on the RTVS may be delayed.

3.4 Implementation of the Real-Time View Server

3.4.1 Architecture of the RTVS

In this section, implementation of the RTVS is described. The RTVS is implemented on a POSIX compliant operating system. It uses the shared memory, semaphores and message queues which are supported by the operating system for inter-process communication.

The architecture of the RTVS is shown in Fig.3.5. The RTVS can serve multiple clients simultaneously by multi-process architecture. A child process of the RTVS is assigned to each client. Child processes share the server cache which stores recently accessed objects. The server cache is mutually excluded by the semaphore. The server cache is also shared by RTDS connection child processes. The RTDS connection child process receives present data from the RTDS communication process and stores them to the server cache. The RTDS connection child process also receives event data and sends it to the message queue.

Applications utilize the functions of the RTVS by the RTVS library functions. The RTVS library is linked with application program and runs in the same context as the application. An application accesses to objects on the client cache. Temporal validity of objects on the client cache is maintained by the temporal object model described before.

3.4.2 Persistent Object Management

As well as ordinary object-oriented database management systems, the RTVS has functions for object persistence. Without object persistence, objects allocated by an application are discarded when the process of the application terminates. To make applications accessible to allocated objects across lifetimes of the application process, contents of the objects have to be stored to in-volatile memory before the application process terminates and restored to the main memory when the application accesses to the same objects on its next lifetime. An object persistency mechanism achieves these tasks.

Prior to creating a new persistent object, schema for objects need to be defined. In the RTVS, a schemata is defined by specifying a schemata name and an ordered set of attributes. Each attribute is represented by a type and a name. The size of an attribute can be derived from the attribute type. The sum of sizes of all attributes

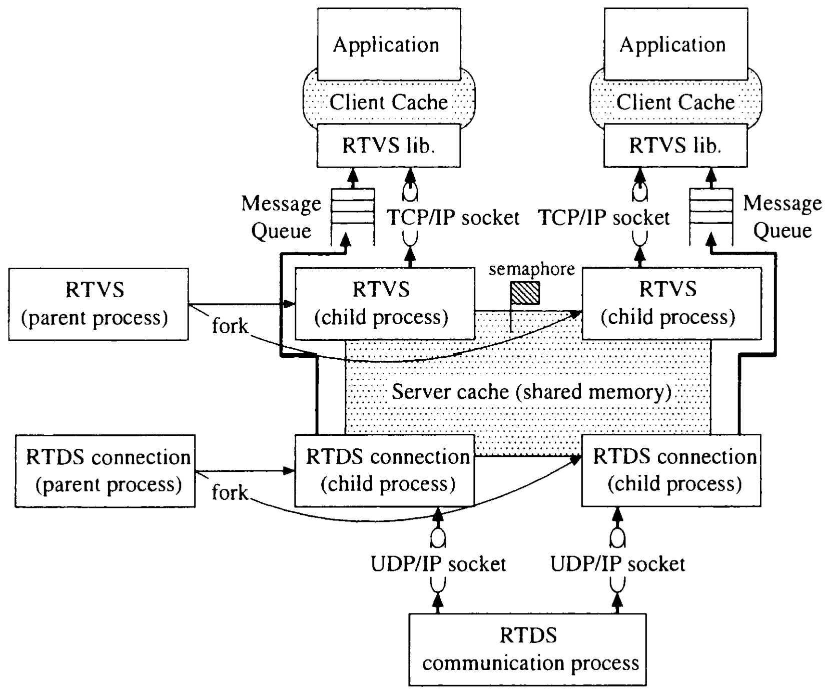


Figure 3.5: Architecture of the RTVS on a POSIX Compliant OS

makes an object size.

An application can create a persistent object by calling a library function with a schemata name. The RTVS allocates a required area in the server cache and the client cache according to the object size, assigns a new object identifier and returns a reference pointer to the application. Then the application can access to the object through the reference pointer.

When an application issues a library function call for the commit request, the contents of a persistent object is stored to a disk at the time. Updates on the persistent object is done through the reference pointer in the client cache, so newest value of the object only resides in the client cache. When an application commits a transaction, contents of updated objects are transferred to the server cache and also written to a disk.

At the next lifetime of the application, persistent objects can be accessed by their object identifier. Object identifiers can be obtained by conditional search and link navigation. When an application requests to access to an object by its object identifier, the RTVS determines the location of the object in a disk, reads it and stores to the server cache. Then the contents of the object on the server cache are transferred to the client cache and the reference pointer is returned to the application.

The RTVS also has an indexing mechanism by B-Tree. Conditional search for attributes with indices is achieved by B-Tree traversal.

3.4.3 Temporal Object Management

A temporal object in the RTVS is represented as an extension of a normal persistent object. However, there are some limitations. First, a schemata of a temporal object has to consist of a time stamp attribute and value attributes. Intuitively, a schemata of a temporal object is like an example shown in Fig.3.1. Practically, a schemata of a temporal object is defined by choosing one of schema on the RTDS. Schema on the RTDS can be defined only in the form satisfying this limitation.

Secondly, a temporal object has to be validated every time an application accesses, since an application accesses to an object on the client cache directly through a reference pointer. Validation is done by calling a library function.

Thirdly, a temporal object cannot be updated. Update requests to a temporal object by an application fails with an error.

Fig.3.6 shows a mechanism for temporal object management. An object has a header part and a body part. The header part includes internally used data for object handling, such as object identifier and cache control flags. A temporal object has an extensional header for temporal validity management in the client cache. A reference pointer passed to an application is an address of the body part.

The body part can be accessible based on a structure definition.

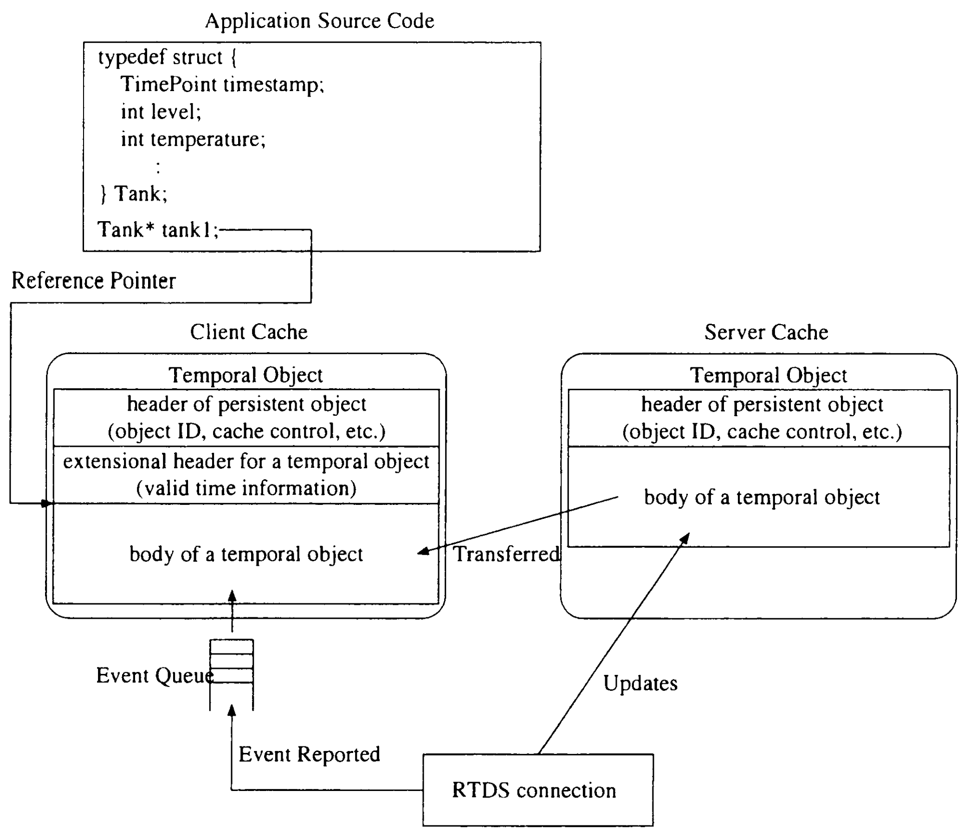


Figure 3.6: Temporal Object Management Structure

Management methods for each temporal object type are different from each other.

3.4.3.1 Real-Time Object

As described before, a real-time object represents the present status of a target. The present data is sent from the RTDS to the RTDS connection periodically. The RTDS connection updates the corresponding body part on the server cache when receives the present data. By this mechanism, the server cache always holds the

newest data.

On the client cache, the extensional header for a temporal object holds a valid interval. When a real-time object is validated, a time stamp t of the real-time object in Definition 3.1 is regarded as a value of a time stamp attribute. Valid interval t_v is held in the extensional header. When the evaluation result of a real-time object is temporally invalid, the newest data is requested to the server and the contents of the server cache is transferred to the client cache.

3.4.3.2 Trend Object

A trend object represents a series of the status for a certain interval. A schemata for a trend object represents a structure of a scene in a series. Therefore, the body part of a trend object is represented by an array of scenes composing a series.

The start time point t_s and the end time point t_e in Definition 3.2 corresponds to the value of the time stamp attribute of the first array element and the last array element, respectively. The period p corresponds to the difference of the value of the timestamp attribute of adjacent two scenes. Valid interval t_v corresponds to the valid interval held in the extensional header.

The start time point t_{s_v}, the end time point t_{e_v} and the period p_v for validation arguments are specified by parameters of the validation library function. If a trend object doesn't satisfy the condition of temporal validity, the corresponding scenes are requested to the RTDS and the client cache is updated. Since the contents of a trend object is organized by an array, the size of a trend object varies depending on the number of scenes.

3.4.3.3 Event Object

Management of an event object is rather different from those of real-time and trend objects. As well as a real-time object, a reference pointer to access to an event object points to the body part of an object in the client cache. When an application requests the temporal validation for an event object, the first element of the event queue is extracted and stored in the body part of an event object. If the event queue is empty, the execution of an application is controlled by application's specification. If blocking mode is specified, the validation procedure is blocked until an event is stored in the event queue. If non-blocking mode is specified, the validation procedure returns a notification. This mechanism is realized easily by utilizing the standard OS function for the message queue access.

3.4.4 Link Management

A link to another object can be realized by storing a reference of the target object. In normal programs, the reference of an object can be a pointer representing an address in the process memory space. For dealing with persistent objects, however, the pointer cannot be used because the address dynamically changes when the object is loaded from a disk to memory. Therefore, an object identifier is used to represent a reference of the object.

Fig. 3.7 shows a mechanism for realizing a temporal link. The RTVS provides a special type '*Link*' to store the object identifier of a linked object. The object identifier is represented by a pair of a type name and a serial number of created objects. In the figure, a normal persistent object with object identifier 'Equip, 11' has a link to a real-time object 'Sensor, 12'. To get a linked object, the RTVS provides a library function '*navigate_link()*'. In this function, an object identifier of '*Link*' attribute is evaluated and a target object is retrieved by the object identifier.

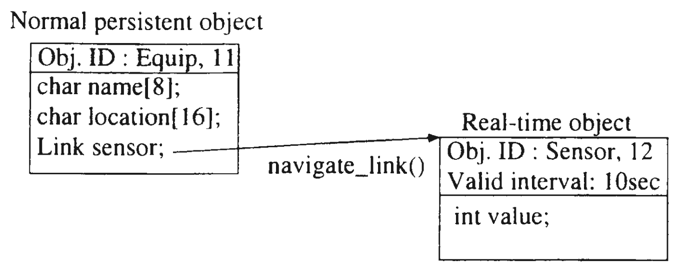


Figure 3.7: Link Management

3.5 Summary

In this chapter, a temporal object model is proposed to deal with temporal data in plant monitoring systems. In this model, temporal objects are classified into three types depending on the temporal characteristics. To define the correct status of a temporal object, the temporal validity is defined based on the valid interval.

A database management system with the proposed temporal object model is realized on a UNIX compliant OS.

Chapter 4

A Batch Manufacturing Information Management System

4.1 Manufacturing Information Management for Batch Processes

In the previous chapter, the temporal object model for dealing with temporal aspects of plant targets has been proposed. The model has the ability of representing historical data of sensed values as an object. This feature enables the integration of dynamically changing sensor data with other kinds of data such as equipment information and product information.

In this chapter, the application of the temporal object to batch manufacturing information management is presented with a novel model for effective utilization of production data.

At first, required functions of manufacturing information management for batch processes are analyzed based on the standard of batch control described in [1].

4.1.1 An Outline of Batch Processes

A manufacturing process consists of a set of units of manufacturing equipment and produced products. A manufacturing process can be classified into three types according to its properties, as described in 2.2.1. From this classification, batch processes have characteristics of both of continuous processes and discrete parts manufacturing processes.

In real batch processes, a batch is identified from another batch by a *batch identifier* such as a batch number. However, this batch identifier may change as the batch moves through a batch process. For example, the batch number is changed after mixing or dividing batches. Another aspect of batch processes is modifiable production procedures. Batch production sequence of units of equipment is subject to be changed according to batch kind or production results.

For these characteristics of batch processes, high functionality is required in batch manufacturing information management systems compared with continuous processes or discrete parts manufacturing processes.

4.1.2 Requirements for Batch Manufacturing Information Management

Based on the characteristics of batch processes described above, the following functions are required for batch manufacturing information management systems.

- Relationships between historical process values acquired from each unit of equipment and batches produced by some units of equipment should be retrieved easily.

- Batch history such as the sequence of used units of equipment and operations of mixture or division of batches should be traced easily.

- Modification of production procedure or addition of some new pieces of equipment should be performed easily.

To satisfy these requirements, a batch manufacturing information management model is proposed in the next section.

4.2 Multidimensional Views

As described before, relationships between historical process values periodically acquired from each unit of manufacturing equipment and batches produced by passing over some units of manufacturing equipment should be provided in comprehensible forms in batch manufacturing processes. The manufacturing information management system described here approaches this requirement by generating multidimensional views from acquired data from a manufacturing process. Batch processes can be regarded that products move through multiple units of equipment as production goes on. Fig. 4.1 shows a concept of the multidimensional views. Suppose that a batch process is in the cube. The batch process can be seen from the three sides. At first, when it is seen from the top side, overall batch movement through units of equipment can be observed. Secondly, when it is seen from the right side, production status of batches for a specific unit of equipment can be obtained. Thirdly, from the left side, production status for a specific batch moving through units of equipment can be traced.

These primitives are orthogonal because a real plant is three dimensional and a cube can be seen from three sides (the opposite side of each side can be regarded as the same). Therefore, the following three views are provided in this system.

- Plant View

  The overall status of a batch process is surveyed by representing *"when, where and what."*

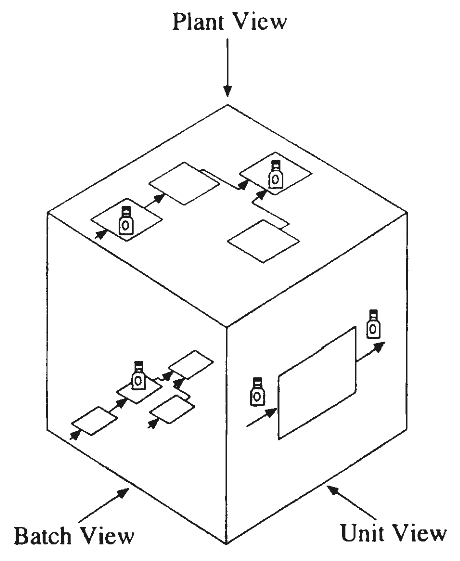


Figure 4.1: A Concept of the Multidimensional Views

- Unit View

  This view focuses on one specific unit of equipment and represents "*when, what and how.*"

- Batch View

  This view focuses on one specific batch and represents "*when, where and how.*"

Using the Plant View, operators can make a production plan which lets a batch process run efficiently. Using the Unit View, influence to batches can be analyzed in case of faults of a specific unit of equipment. Using the Batch View, which unit of equipment has a problem can be detected when a defect is found in a certain batch of products.

Hereafter, assuming that there are m units of equipment $U = \{u_1, u_2, \cdots, u_m\}$ and n batches $B = \{b_1, b_2, \cdots, b_n\}$ have been produced, formal definitions for each view are described.

4.2.1 Plant View

The Plant View represents batches which have been produced on each unit of equipment during a specified time interval. The Plant View is defined as follows.

- Specified condition parameters for the Plant View are
  - a time interval $[t_s, t_e)$
  - a period t_p
- Plant View PV is a set of *plant view elements* pv_{t_i} for the time t_i. pv_{t_i} is represented by $PV = \{pv_{t_i} | t_i = t_s + (i-1)t_p, 1 \leq i \leq \lceil (t_e - t_s)/t_p \rceil\}$.
- Plant view element pv_{t_i} for t_i includes the following attributes:
  - a time stamp t_i.
  - batches $b^{t_i}_{u_1}, b^{t_i}_{u_2}, \cdots, b^{t_i}_{u_m}$ which have been produced on equipment unit $u_1, u_2, \cdots, u_m$, respectively.

4.2.2 Unit View

The Unit View represents production status for each of batches which have been produced by a specific unit during a specified time interval. The Unit View is defined as follows.

- Specified condition parameters for the Unit View are
  - a unit of equipment $u \in U$
  - a time interval $[t_s, t_e)$
- Unit View UV is a set of k *unit view elements* $uv_{b_i^{(u,t_s,t_e)}}$ which are represented by $UV = \{uv_{b_i^{(u,t_s,t_e)}} | \forall b_i^{(u,t_s,t_e)} \in B^{(u,t_s,t_e)}\}$. $B^{(u,t_s,t_e)}$ is a subset of B

such that it includes k batches which have been produced during the time interval $[t_s, t_e)$ on the unit u.

- Unit view element $uv_{b_i^{(u,t_s,t_e)}} (1 \leq i \leq k)$ includes the following attributes:
  - a batch identifier b_{ID}.
  - the start time t_s^u and the end time t_e^u of production of batch $b_i^{(u,t_s,t_e)}$ on the unit u.
  - History of process values $H_u^{t_s^u, t_e^u}$ acquired from the unit u during time interval $[t_s^u, t_e^u)$. Time interval $[t_s^u, t_e^u)$ represents a period when batch $b_i^{(u,t_s,t_e)}$ existed in unit u.

4.2.3 Batch View

The Batch View represents production status for each of equipment units which a specified batch has moved through. The Batch View is defined as follows. In this definition, the term *derived batch* represents a batch which is recursively mixed or divided to the specified batch.

- Specified condition parameter for the Batch View is
  - a batch $b \in B$

  Generally, a batch specified here assumed to be a final product batch, not an intermediate batch before mixture or division.

- Let k derived batches from the batch b be $B^b = \{b_1^b, b_2^b, \cdots, b_k^b\}$.

- The Batch View BV is a set of l *batch view elements* $bv_{u_i^b}$ which is represented by $BV = \{bv_{u_i^b} | \forall u_i^b \in U^b\}$. U^b is a set of units of equipment in which at least one batch in B^b has been produced.

- The batch view element $bv_{u_i^b}$ includes the following attributes:
  - a unit identifier u_{ID}.
  - unit view elements $uv_{b_q^{(b,u_i^b)}}$ for each of h batches $b_1^{(b,u_i^b)}, b_2^{(b,u_i^b)}, \cdots, b_h^{(b,u_i^b)}$ which have been produced in u_i^b.

4.3 A Manufacturing Management Data Model

To generate the three types of views described above, batches need to be traced and histories of batch movement through several equipment units need to be stored by detecting the batch movement. Therefore, a flexible data model is required to represent the complex manufacturing process of batch processes. As described before, produced batches move through units of equipment as production goes on. In addition, batches may be mixed to one batch or a batch may be divided to multiple batches during the production process. To represent these special properties in a database, the object-oriented data model seems to be appropriate because it regards a target as an object and encapsulates the status and actions in the object.

4.3.1 Object Classes

The following object classes are defined for the batch tracing mechanism.

- Trend Object
  This object handles the sensed physical values from a unit of equipment and represents a history of physical values for the unit of equipment.

- Event Object
  This object handles status changes in a manufacturing process and represents batch movement in the manufacturing process.

- Batch Object

  An instance of this object class corresponds to one batch in a real manufacturing process. The following attributes are included in this object:

  - a batch identifier (such as a batch number).
  - the start time stamp and the end time for each of equipment units which represent the time when the batch enters and leaves, respectively.
  - links to other batch objects of batches preceding this batch in mixture or division.

- Unit Object

  An instance of this object class corresponds to one unit of equipment in a real manufacturing process. The following attributes are included in this object:

  - current batch

    A link to a batch object which is processed on this unit.
  - finished batches

    Link(s) to batch object(s) which have finished to be processed on this unit and are waiting for processing by the successive unit.
  - trend data

    A link to a trend object which represents a history of process values of this unit.

4.3.2 A Data Structure

Objects described above are organized as shown in Fig. 4.2. A unit object has a link to a batch object which is processed. An event object reports a batch movement event to a corresponding unit object. Links among batch objects are set when batch mixture or division occurs.

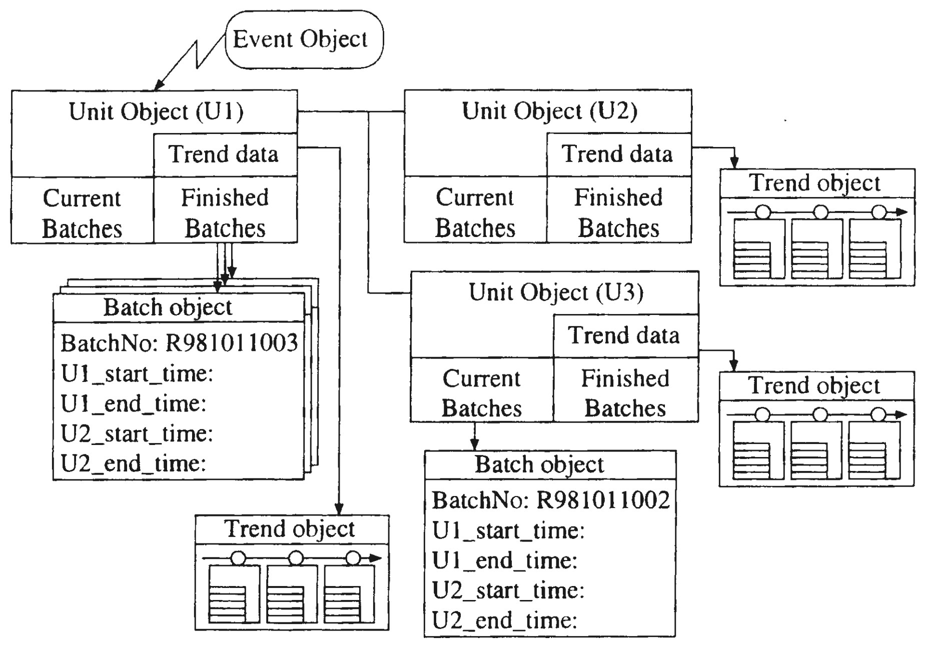


Figure 4.2: A Data Structure for the Batch Tracing Mechanism

This data structure enables to *virtually* represent the real manufacturing status and trace the batch movement which is subject to be changed at production time. Furthermore, adding a new unit of equipment can be reflected by inserting a new instance of the unit object. Adding a new type of produced products can also be realized by defining a new derived class of the batch object. Therefore, the complex production process of batch processes can be represented flexibly in this data model.

4.4 Implementation of the View Generation Library

4.4.1 Plant View

Fig.4.3 shows the structure of the Plant View. A batch which has been produced at each time point is associated for each of units of equipment.

Time	u_1	u_2		u_m	
t_s	$b^{t_1}_{u_1}$	$b^{t_1}_{u_2}$		$b^{t_1}_{u_m}$	
t_s+t_p	$b^{t_2}_{u_1}$	$b^{t_2}_{u_2}$		$b^{t_2}_{u_m}$	
$t_s+(i\text{-}1)\ t_p$	$b^{t_i}_{u_1}$	$b^{t_i}_{u_2}$		$b^{t_i}_{u_m}$	pv_{t_i}
$t_s+(l\text{-}1)\ t_p$	$b^{t_l}_{u_1}$	$b^{t_l}_{u_2}$		$b^{t_l}_{u_m}$	

Figure 4.3: The structure of the Plant View

For each time point $t_i = t_s + (i-1)t_p$ $(1 \leq i \leq \lceil (t_e - t_s)/t_p \rceil)$ which can be calculated based on specified parameters of the Plant View, a time interval $[t_s, t_e)$ and a period t_p and each unit of equipment u_j $(1 \leq j \leq m)$, a batch $b^{t_i}_{u_j}$ which has been being produced by unit u_j at the time t_i can be retrieved by the procedure shown in Fig.4.4.

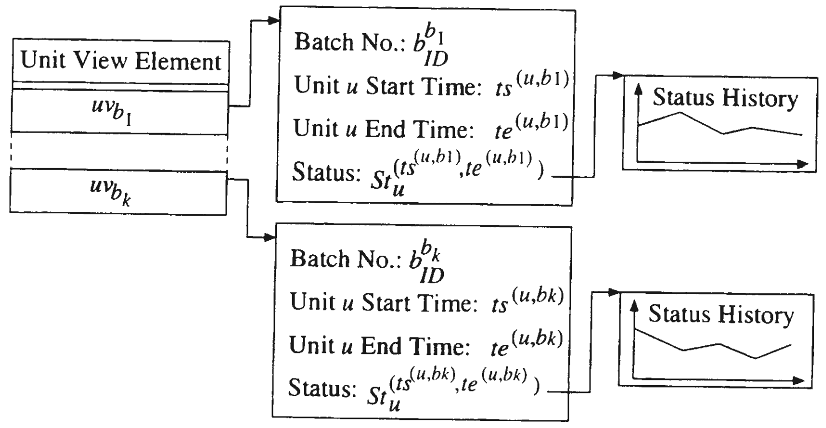
In the above procedure, $b.u_j\_start$ and $b.u_j\_end$ are the start time and the end time when the batch b has been produced by the unit u_j. The Plant View can be obtained by performing this procedure for all u_j and all t_i.

```
$b^{t_i}_{u_j} = NULL$
for $(b = b_1, b_2, \cdots, b_n)$ {
    if $(b.u_j\_start \leq t_i < b.u_j\_end)$ {
        $b^{t_i}_{u_j} = b$
    }
}
```

Figure 4.4: An Algorithm for Obtaining a Cell of the Plant View

4.4.2 Unit View

Fig. 4.5 shows the structure of the Unit View. In the Unit View, for each batch which has been produced by the specified unit u during the specified time interval $[t_s, t_e)$, the batch object and the status history of u during the production of the batch are provided.



Unit View Element
uv_{b_1}
uv_{b_k}
Batch No.: $b^{b_1}_{ID}$
Unit u Start Time: $ts^{(u,b1)}$
Unit u End Time: $te^{(u,b1)}$
Status: $St_u^{(ts^{(u,b1)},te^{(u,b1)})}$
Status History
Batch No.: $b^{b_k}_{ID}$
Unit u Start Time: $ts^{(u,bk)}$
Unit u End Time: $te^{(u,bk)}$
Status: $St_u^{(ts^{(u,bk)},te^{(u,bk)})}$
Status History

Figure 4.5: The Structure of the Unit View

A set of batches $B^{(u,t_s,t_e)}$ which have been produced by unit u during time interval $[t_s, t_e)$ can be obtained by the procedure shown in Fig.4.6.

```
$B^{(u_i,t_s,t_e)} = \phi$
for $(b = b_1, b_2, \cdots, b_n)$ {
    if $(t_s \leq b.u_i\_start \leq t_e \; \& \; t_s \leq b.u_i\_end \leq t_e)$
        add $b$ to $B^{(u_i,t_s,t_e)}$
    }
}
```

Figure 4.6: An Algorithm for Obtaining the Unit View

For each batch in $B^{(u,t_s,t_e)}$ obtained by the above procedure, the Unit View can be generated by assigning the batch identifier to b_{ID} of $uv_{b_j^{(u,t_s,t_e)}}$, the start time and the end time at the unit u to t_s^u and t_e^u of $uv_{b_j^{(u,t_s,t_e)}}$ respectively and the status history for the unit u during the time interval $[t_s^u, t_e^u)$ to $H_u^{t_s^u,t_e^u}$ of $uv_{b_j^{(u,t_s,t_e)}}$. The status history can be obtained from the Trend Object corresponding to the unit u.

4.4.3 Batch View

Fig. 4.7 shows the structure of the Batch View. The histories of process values for each of equipment units which a specified batch and its intermediate products have moved through are provided.

For generating the Batch View, derived batches from batch b which is specified as a view generation parameter need to be obtained. As described before, since the batch object has links which associate the mixed or divided batches with it, this derivation can be performed by traversing the links from the final product to the raw materials.

Another required processing for generating the Batch View is to obtain units of equipment which a specified batch has passed. Since a batch object has the attributes of time stamps which record the time when the batch has entered and

left each unit, whether a batch has passed a unit or not can be detected by checking whether the time stamp for the unit is recorded or not.

Since derived batches from batch b can be obtained for each of units by the above procedure, the Batch View can be generated by organizing the unit view element for each unit.

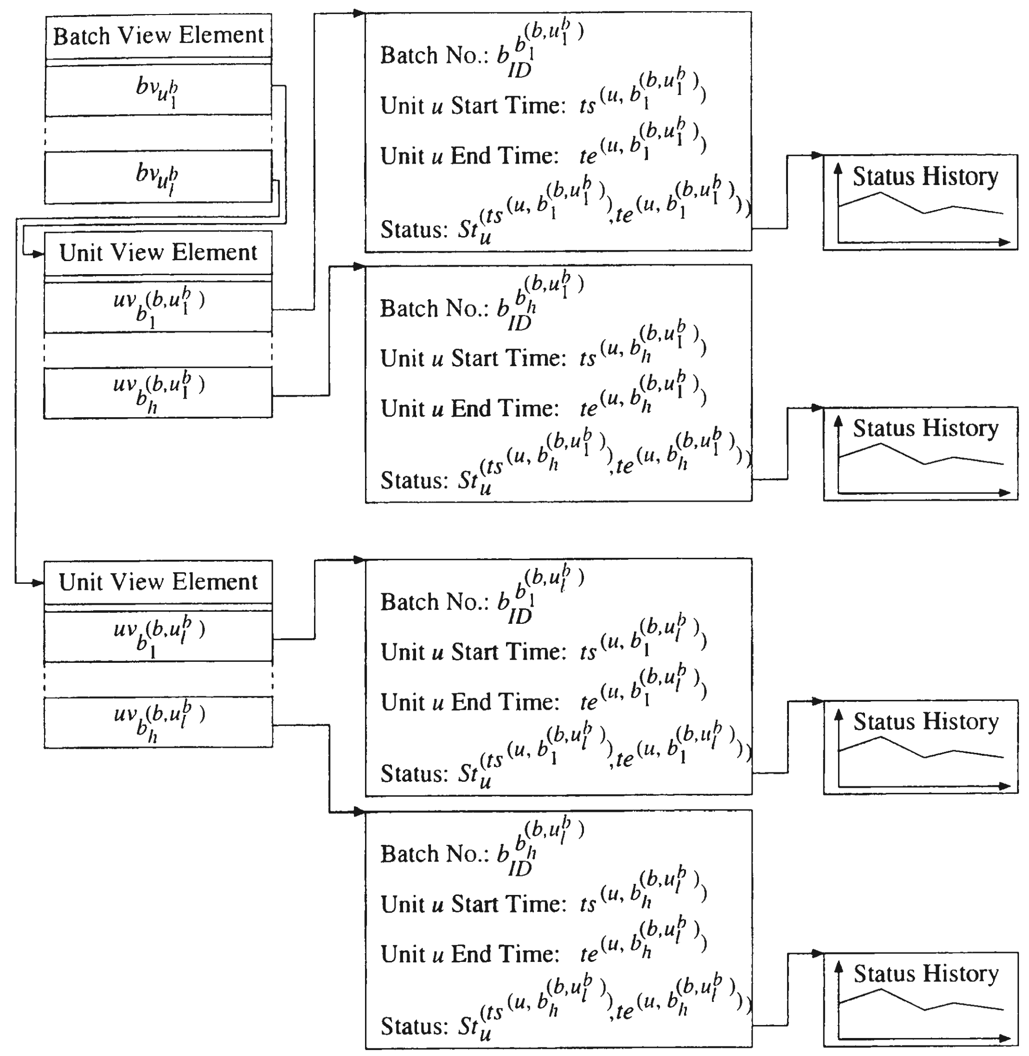


Figure 4.7: The Structure of the Batch View

4.5 Batch Tracing Based on the Active Mechanism

In the manufacturing management data model, batch objects need to be migrated through unit objects according to the production status of a real manufacturing process. Since links for generating the three types of views are organized at the time of batch production by realizing this mechanism, fast view generation can be performed.

As a framework for starting an action in response to an event occurred in databases or physical world, the active mechanism[8] is well known. The active mechanism is also called the ECA mechanism. The mechanism works as follows: when an event (E) occurred, a corresponding condition (C) is evaluated. If the condition holds, an action (A) is started. Utilizing this mechanism, migration of a batch object can be performed as an action.

Batches are traced by the active mechanism in the following manner.

- When starting of processing is detected in a unit of equipment, the unit object obtains the first batch object which is linked with finished batches of the preceding unit object, and links it to the current batch attribute.

- When ending of processing is detected in a unit of equipment, the unit object removes the link from the current batch attribute and adds to the finished batches.

In addition, units of equipment are classified into the following five types to manage the batch mixture and division as shown in Fig. 4.8.

- Processing Unit

  This type of units receives one batch and sends one batch.

- Mixing Unit

  This type of units receives batches of different types and sends one batch.

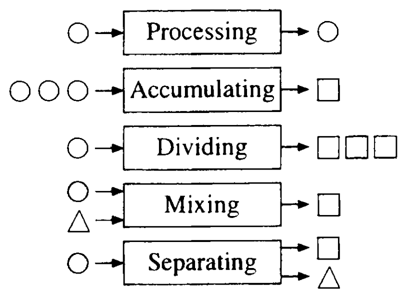


Figure 4.8: Process Primitives for the Batch Tracing Mechanism

- Separating Unit

  This type of units receives one batch and sends batches of different types.

- Accumulating Unit

  This type of units receives batches of the same type and sends one batch.

- Dividing Unit

  This type of units receives one batch and sends batches of the same type.

The unit object generates links among batch objects based on this classification. Since batches can be traced backward by navigating these links without searching all batch objects, fast view generation can be performed.

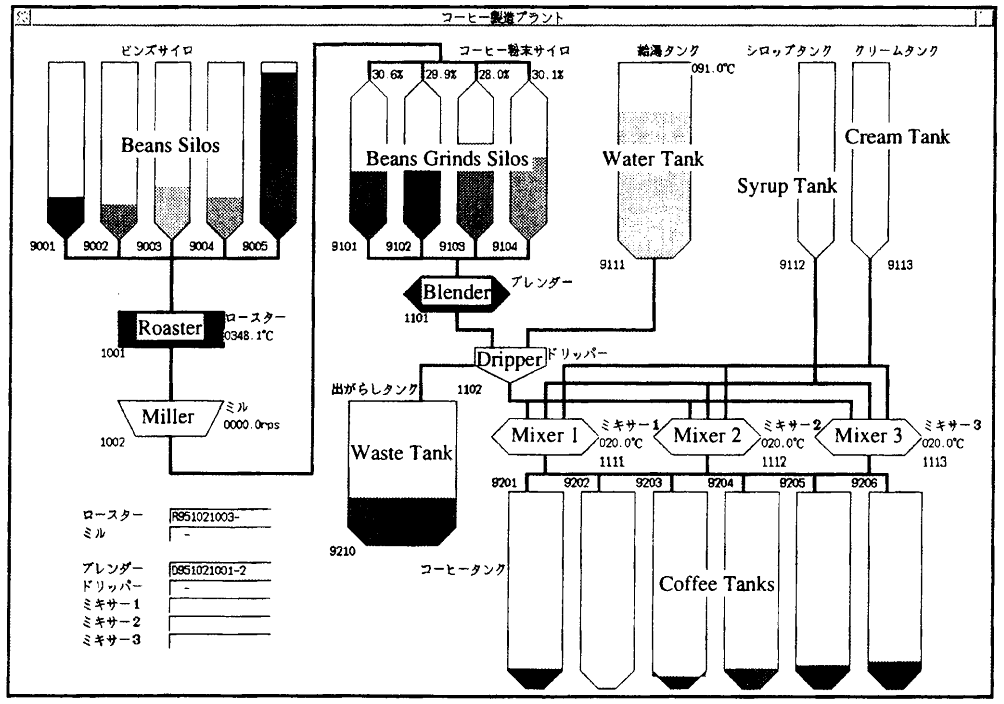
4.6 Example Application

In this section, an example application is introduced to show how the batch manufacturing information management model is useful in a practical system. A plant model for the example application is a coffee making plant.

In this application, a simulator produces a plant operational data. Fig.4.9 shows a monitoring screen of the simulated coffee plant. Raw coffee beans are stocked in the beans silos located in the upper left corner. One of five kinds of raw

beans selected, a batch of coffee beans is taken out and thrown to the roaster. In the roaster, a batch of raw coffee beans is roasted. After that, the batch is transferred to the miller which grinds the beans. The milled coffee beans are stocked in the beans grinds silo located in the upper center of the screen. At this point, a batch becomes to be indistinguishable from other batches.

From beans grinds silo, some kinds of beans are chosen and transferred to the blender which stirs beans grinds. After blended, hot water is poured into the dripper, cream and syrup are mixed in the mixer and stocked to coffee tanks. In the mixer, four batches are accumulated into one batch.



ロースター	R951021003-
ミル	-
ブレンダー	D951021001-2
ドリッパー	-
ミキサー1	
ミキサー2	
ミキサー3	

Figure 4.9: A Monitor Screen of a Coffee Plant Simulator

In this example plant, the following process values are acquired periodically and stored in the RTDS.

Roaster: temperature of heating plate

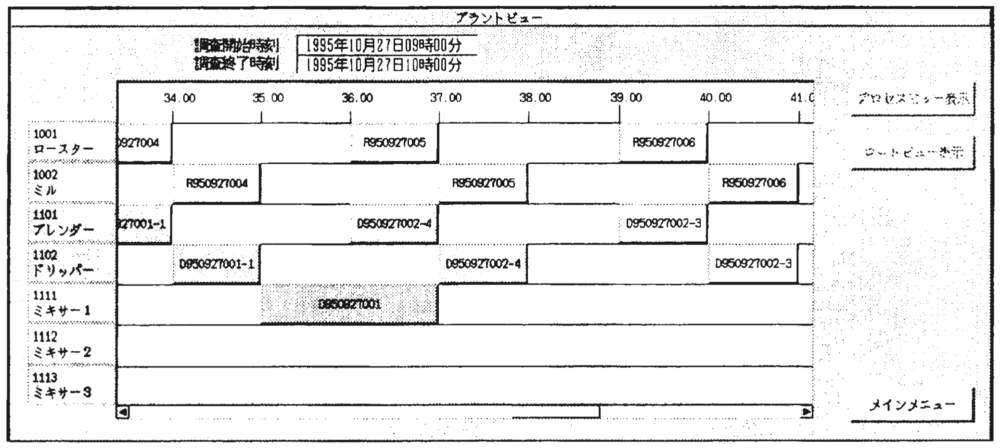
Miller: the number of rounds of revolution per minute

Blender: humidity of each of beans grinds silos

Dripper: temperature of poured water

Mixer: temperature of coffee liquid

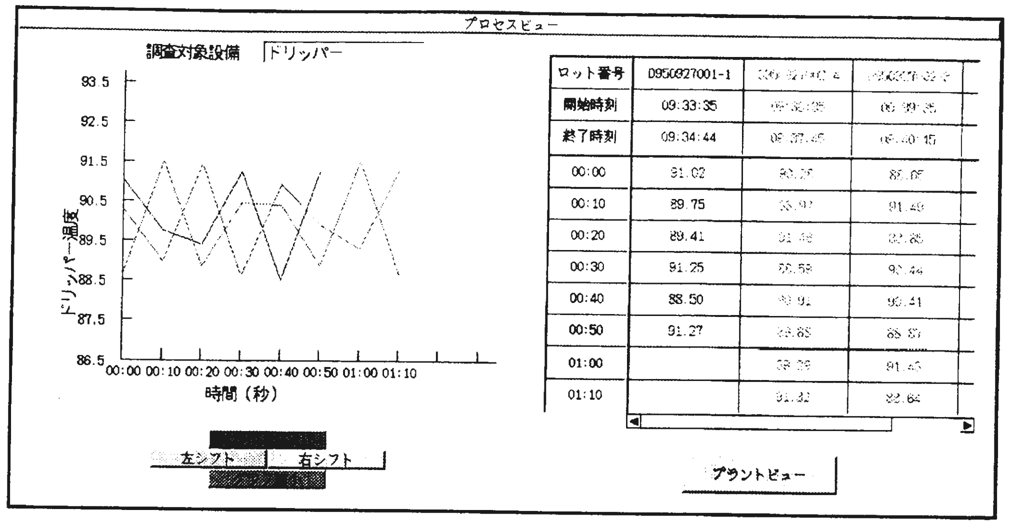
The Plant View is obtained by inputting a time interval to a dialog. An example is shown in Fig.4.10. In the chart, process units are arranged vertically and time axis is plotted horizontally. Each rectangle in the chart represents a batch. A batch identifier is shown in the center of the rectangle.



調査開始時刻 1995年10月27日09時00分
調査終了時刻 1995年10月27日10時00分

Figure 4.10: An Example of the Plant View

By clicking a process unit in the Plant View window, the Unit View for the clicked process unit is displayed. Fig.4.11 shows an example of the Unit View for the dripper. In this example, three batches were processed at the dripper during the specified time interval. Each line of the graph represents the status history of each batch.



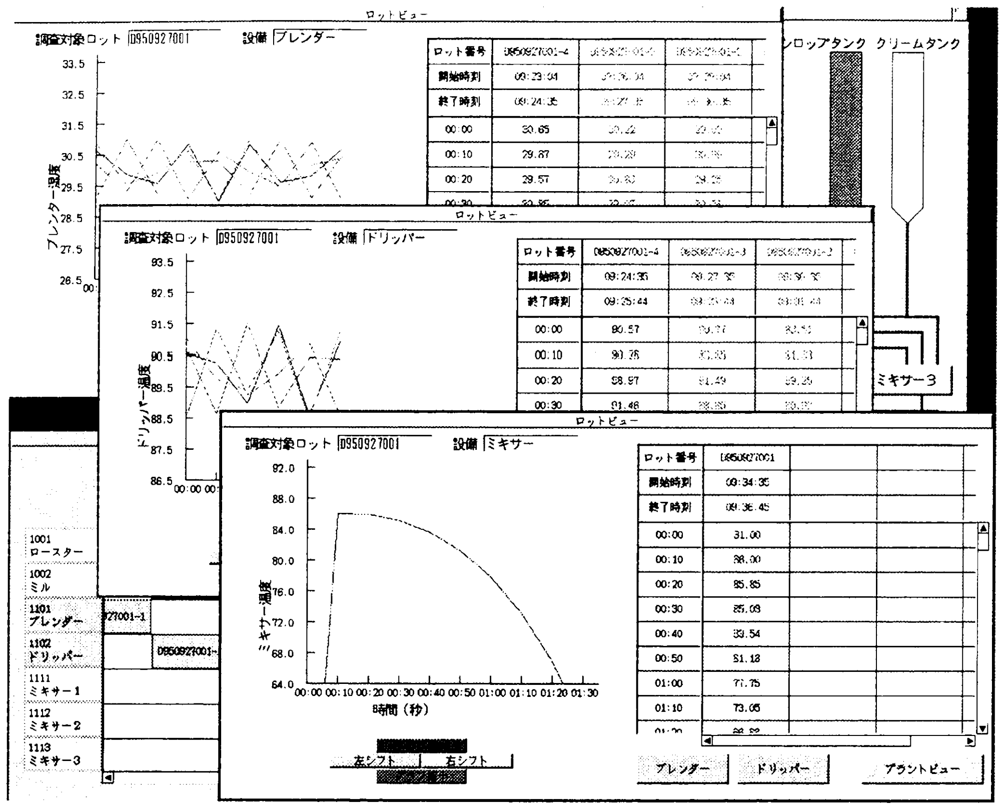
調査対象設備 ドリッパー

ロット番号	D950927001-1	[illegible]	[illegible]
開始時刻	09:33:35	[illegible]	[illegible]
終了時刻	09:34:44	[illegible]	[illegible]
00:00	91.02	[illegible]	[illegible]
00:10	89.75	[illegible]	91.49
00:20	89.41	[illegible]	[illegible]
00:30	91.25	[illegible]	[illegible]
00:40	88.50	[illegible]	90.41
00:50	91.27	[illegible]	[illegible]
01:00		[illegible]	[illegible]
01:10		[illegible]	[illegible]

Figure 4.11: An Example of the Unit View

By clicking a batch in the Plant View window, the batch view for the clicked batch is displayed. An example of the Batch View is shown in Fig.4.12. The selected batch is 'D950927001'. In the example, three windows corresponding to blender, dripper and mixer are shown. In the windows for blender and dripper, four lines are plotted in the graph because the specified batch are derived from four batches at the mixer.

Utilizing the Plant View, an operator can optimize the operation rate of each process unit. Utilizing the Unit View, an operator can tune process parameters or analyze the relation of cause and effect for products. Utilizing the Batch View, an operator can trace back and make use of a production history for a certain batch and detect a cause of defect.



調査対象ロット D950927001 設備 ミキサー

ロット番号	D950927001
開始時刻	09:34:35
終了時刻	09:36:45
00:00	31.00
00:10	88.00
00:20	85.85
00:30	85.03
00:40	83.54
00:50	81.13
01:00	77.75
01:10	73.05

Figure 4.12: An Example of the Batch View

4.7 Summary

In this chapter, a batch manufacturing information management system based on the multidimensional view has been presented. In order to deal with several aspects of manufacturing information, a data utilization model based on the multi-dimensional view, i.e, Plant View, Unit View and Batch View has been proposed.

In addition, a batch manufacturing information management system has been implemented on the RTVS based on the model. An example for a coffee manufacturing plant has also been illustrated.

Chapter 5

The Time/Place/Object Model

5.1 Backgrounds

In the previous chapter, it is shown that how the production data can be utilized effectively for quality management or defect analysis. However, the application is only one way, i.e. plant floor to business systems. On the other hand, the batch tracing mechanism presented in the implementation section of the previous chapter is powerful enough for tracking the manufactured products in a manufacturing line. For the next step, a model for supporting the opposite way, i.e business systems to plant floor is required.

In this chapter, a framework for supporting both ways, incorporating with control tasks is proposed to realize advanced manufacturing management systems.

5.1.1 System Organization

Fig. 5.1 shows a typical organization of a manufacturing line control system. The system consists of three layers.

Business system layer: Systems in this layer perform planning and management tasks such as production scheduling, inventory management and equipment maintenance management.

Process computer layer: This layer bridges between the business systems and control systems. This layer issues preset values to controllers according to production recipes from business systems and gathers production performance from controllers.

Controller layer: Controllers handle signals of connected actuators and sensors according to programmed sequence and control logic.

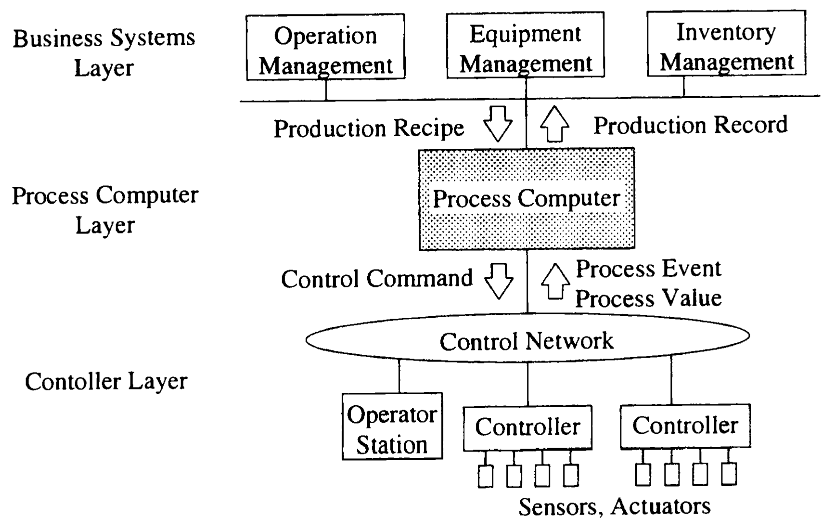


Figure 5.1: A Typical System Organization

5.1.2 Tracking and History Management Functions

The Time/Place/Object model is intended to provide a fundamental model to describe the structure and the behavior of a manufacturing line and realize tracking and history management functions performed in the process computer layer. An example of a control function is depicted in Fig. 5.2. When "Lot #1" is thrown to

"Process A" in a plant field, a process value of a sensor equipped with "Process A" changes and a program in a controller detects the change. The detection triggers a program in the process computer via the control network. The triggered program gets a pre-received corresponding production recipe which includes control parameters, then issues commands such as a preset value of the temperature to controllers. The program also updates the current tracking status in the process computer and reports it to the business systems.

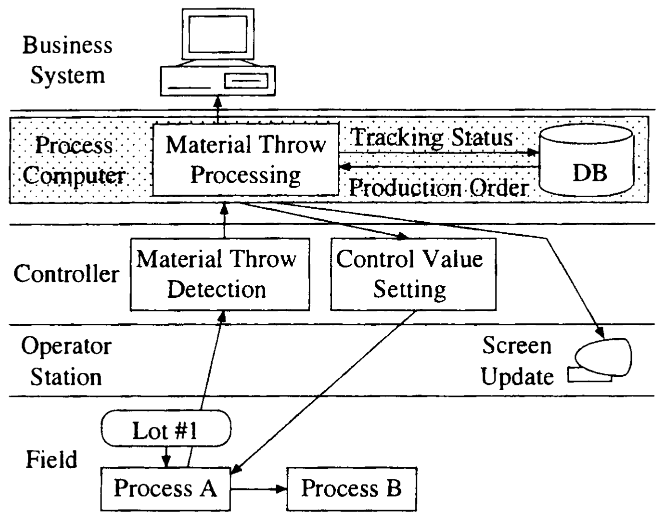


Figure 5.2: An Example of a Control Function

Main mechanisms to realize required functions in a process computer can be classified into two types: one is plant-wide material tracking and the other is history management of production performance data. The tracking mechanism gives a foundation for grasping which material exists in which production process and invoking appropriate actions such as issuing control commands to controllers. History management provides production performance data with business systems in appropriate viewpoints for each product or each production process.

5.2 The Time/Place/Object Model

5.2.1 Basic Concepts

Intuitively, a manufacturing line can be represented so that "manufactured materials move through manufacturing processes as production goes on." Based on this representation, the Time/Place/Object model provides three dimensions: time, place and object. A place corresponds to a physical unit of manufacturing equipment or a logical process of manufacturing steps. An object corresponds to a minimum production unit such as a *batch* in a batch manufacturing plant. An object is an instance of a certain class which defines the properties of the object.

A place, i.e. a manufacturing process, not only updates object properties, but also changes the identity of the production material by combining several objects or splitting an object. In a canned coffee plant, a batch of coffee liquid is mixed up with a batch of cream at one place, while a batch of mixed liquid is divided into small amount of batches to be canned. Five primitives for places are defined in order to represent the change of the production unit caused by object combination and splitting.

In some manufacturing lines, an object exists in only one place at a time. In this case, an object can be regarded as a point. But in others, objects may have length, that is, an object can be regarded as a line and may exist in more than one places at a time. The model provides three types of object forms to represent consistency of relationships among objects and places.

Object behavior in a manufacturing line can be represented as a collection of event-driven object movements between two places. The model represents the object movement such that "when an event occurs, an object is moved from one place to another with some involved actions." Actions to be performed corresponding to an object movements in a real manufacturing line are represented by

ECA rules.

Production performance data are gathered in several forms; some of them show sensor value acquired periodically, while others indicate results of on-demand inspections. These data should be materialized in some viewpoints when retrieved as production history. The model provides viewpoints of acquired data from the three dimensions, i.e. time, place and object.

5.2.2 Basic Definitions

5.2.2.1 Time/Place/Object

In this paper, L time points, M places, N object classes, and N_i objects of the ith object class are described in the following notations.

Time: A sequence of time points $t_1, t_2, \cdots, t_L$

Place: Places $p_1, p_2, \cdots, p_M$

Object: Object classes $c_1, c_2, \cdots, c_N$, objects of a class c_i $o_1^{c_i}, o_2^{c_i}, \cdots, o_{N_i}^{c_i}$

5.2.2.2 Primitives of places

Objects can be combined or split in a place. Places can be categorized into five primitives shown in Fig.5.3 to trace the combination or the split of objects. Each primitive has the following property.

Processing: An object at the entrance side is identical with an object at the exit side.

Accumulating: Some objects of a specific class are combined into one object.

Dividing: An object is divided into some objects of a specific class.

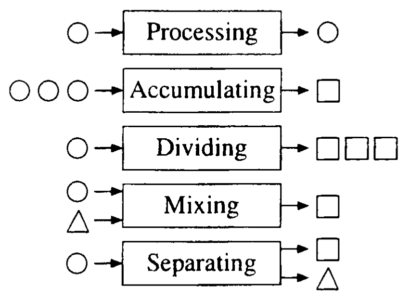


Figure 5.3: Primitives of Places

Mixing: Some objects of several classes enter a place and mixed into one object.

Separating: An object is separated into some objects of several classes.

These primitives are complete because

1. relationships of the entrance side and exit side objects can be categorized $1:1, 1:n$, and $n:1$,
2. if multiple objects enter or leave, these objects belong to either a specific class or several classes.

5.2.2.3 Object Forms

Objects are represented by the following object forms as shown in Fig. 5.4.

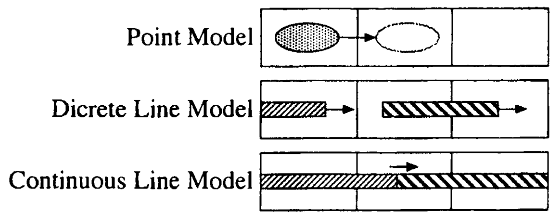


Figure 5.4: Object Forms

Point model: An object can exist in only one place.

Discrete line model: An object can exist in adjacent multiple places but isn't continuous with other objects.

Continuous line model: An object can exist in adjacent multiple places and be continuous with a preceding object and a succeeding object.

These forms are complete because

1. an object can exist in either one place or multiple places,
2. if an object can exist in multiple places, an object is either continuous with other objects or not.

5.2.3 Object Movement

5.2.3.1 Attributes Representing Object Movement

As described before, movement of an object between two places can be represented such that "when an event occurs, an object is moved from one place to another." In the model, an object movement is represented by a tuple of the following five attributes.

$$\{e_{move}, p_{from}, p_{to}, obj\_form, C_{next}\}$$

The meaning of each attribute is shown below.

Movement Event: e_{move} is an event which triggers an object movement.

Origin Place: p_{from} is an origin place.

Destination Place: p_{to} is a destination place.

As special cases, if p_{from} is $NULL$, this movement represents the creation of a new object at p_{to}, and if p_{to} is $NULL$, this movement represents the discharge of an object from p_{from}.

Object Form: obj_form is one of the object forms in Fig. 5.4.

Behavior of object movement differs according to the models of the object form as shown in Fig. 5.5. Suppose that an event where p_{from} is p_i and p_{to} is p_{i+1} occurs. For the point model, an object leaves p_i and enters p_{i+1}. For the discrete and continuous line model, an object enters p_{i+1}, but doesn't leave p_i. Instead, for the discrete line model, an object leaves p_i when an event where p_{from} is p_i and p_{to} is $NULL$ occurs. For the continuous line model, an object existing in p_i leaves when a succeeding object enters p_i, that is, when an event where p_{from} is p_{i-1} and p_{to} is p_i occurs.

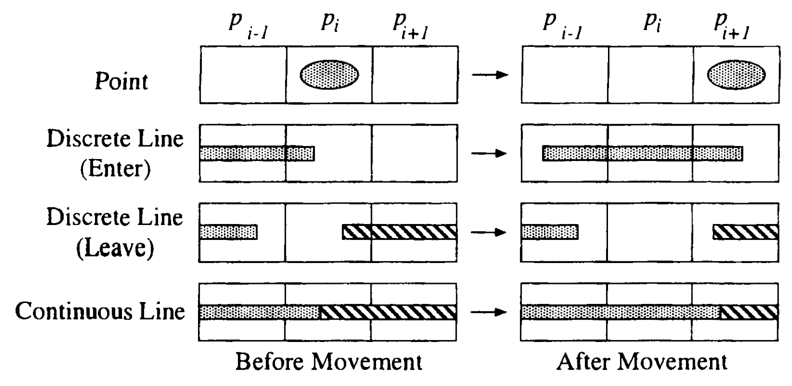


Figure 5.5: Object Movement

Next Object Class: C_{next} specifies a class of an object after accumulating, splitting, mixing or separating.

If p_{from} is not the *processing* place, an object moved to p_{to} is not identical with an object existing at p_{from}. An example in case of separating place is shown in

Fig. 5.6. Suppose that an object of class C_1 is existing in separating place p_1. For movement $(e_1, p_1, p_2,$ point model$, C_2)$, an object of class C_2 is created in place p_2 when event e_1 occurs. For movement $(e_2, p_1, p_3,$ point model$, C_3)$, an object of class C_3 is created in place p_3 when event e_2 occurs. After separation, the object of class C_1 in p_1 leaves p_1 when event e_3 which is associated with movement $(e_3, p_1, NULL,$ point model$, NULL)$ occurs.

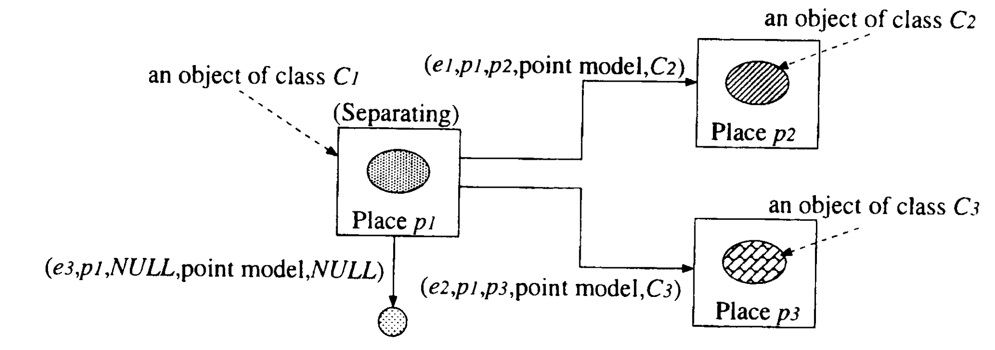


Figure 5.6: Next Object Class Specification

5.2.3.2 ECA rules

When an object moves, some involved actions such as issuing control commands need to be performed to automate production. In the model, the ECA mechanism is used to describe a logic of an object movement and involved actions.

In order to provide a general framework for invoking actions according to an object movement, four categories of ECA rules for the object movement event are defined.

Pre-movement ECA rules: $\{e_{move}, pre\_C_1, pre\_A_1\}, \{e_{move}, pre\_C_2, pre\_A_2\}, \cdots$

An on-movement ECA rule: $\{e_{move}, on\_C, on\_A\}$

Post-movement ECA rules: $\{e_{move}, post\_C_1, post\_A_1\}, \{e_{move}, post\_C_2, post\_A_2\}, \cdots$

Invalid movement ECA rules: $\{e_{move}, inv\_C_1, inv\_A_1\}, \{e_{move}, inv\_C_2, inv\_A_2\}, \cdots$

These ECA rules are invoked by the following logic when a corresponding event occurs.

```
if($pre\_C_1()$ == true) $pre\_A_1()$;
if($pre\_C_2()$ == true) $pre\_A_2()$;
  ...
if($on\_C()$ == true) {
  if($on\_A(p_{from}, p_{to}, obj\_form, C_{next})$ == true) {
    if($post\_C_1()$ == true) $post\_A_1()$;
    if($post\_C_2()$ == true) $post\_A_2()$;
      ...
  } else {
    if($inv\_C_1()$ == true) $inv\_A_1()$;
    if($inv\_C_2()$ == true) $inv\_A_2()$;
      ...
  }
}
```

Each condition function, e.g $pre\_C_1()$, is expected to return a true value if a corresponding condition satisfies, otherwise, return a false value. Object movement function $on\_A()$ is expected to return a true value if the movement is successfully performed. If consistency violation among places and objects, which is described later, is detected, $on\_A()$ returns a false value.

In actual situations, this invocation logic works as follows.

- Pre-movement ECA rules such as collecting production performance data at origin place p_{from} are processed prior to the object movement.
- On-movement condition $on\_C()$, such as confirming whether a reported event is reliable by re-examining a sensor value, is evaluated. If this satisfies, object movement action $on\_A()$ is executed.
- The object movement action $on\_A()$ moves an object from p_{from} to p_{to} according to obj_form and C_{next}.
- If the object movement is successfully performed, post-movement ECA rules such as issuing control commands to p_{to} are processed.
- If the object movement fails, invalid movement ECA rules such as modifying tracking status or alarming to operators are executed.

5.2.3.3 Handling of Object Movement

Handling of object movement is achieved by leaving the origin place and entering the destination place. The handling method depends on the types of place primitives and the object forms.

Handling of leaving place p_i is achieved by the following methods.

1. If the origin place is the processing place
   (a) If the object form is the point model
   an object leaves place p_i when event e_{move} associated with the origin place p_i occurs.
   (b) If the object form is the discrete line model
   an object leaves place p_i when event e_{move} associated with the origin place p_i and the destination place $NULL$ occurs.

(c) If the object form is the continuous line model

an object leaves place p_i when event e_{move} associated with the destination place p_i occurs.

2. If the origin place is the accumulating, mixing or separating place
an object leaves place p_i when event e_{move} associated with the origin place p_i occurs.

3. If the origin place is the dividing place
an object leaves place p_i when event e_{move} associated with the origin place p_i and the destination place $NULL$ occurs.

Handling of entering place p_i is achieved when event e_{move} associated with the destination place p_i occurs.

5.2.3.4 Detection of Inconsistency

If an object movement event cannot be reported to a process computer for some reasons, the status of a modeled manufacturing line in the database of the process computer becomes different from the actual status of a real manufacturing line. If this occurs, the object movement action cannot be performed successfully. If one of the following cases is detected, consistency among places and objects is regarded as violated and invalid movement ECA rules are triggered.

- There is no object at origin place p_{from}.
- There is an object at destination place p_{to} when p_{to} is *processing* place and obj_form is not the continuous line model.
- There is an object at destination place p_{to} when p_{to} is *dividing* or *separating* place.

5.2.3.5 Extension of Primitive Places

In order to represent real manufacturing lines intuitively, some extensions of the processing place are defined. One extension is to define a place which can hold multiple objects. This type of place can be represented as a queue. Queues can be categorized into two types.

- First In First Out (FIFO) Queue
- Last In First Out (LIFO) Queue

Another extension is to define a place where multiple objects move by one event. This type of place can be represented by a conveyor, as shown in Fig. 5.7.

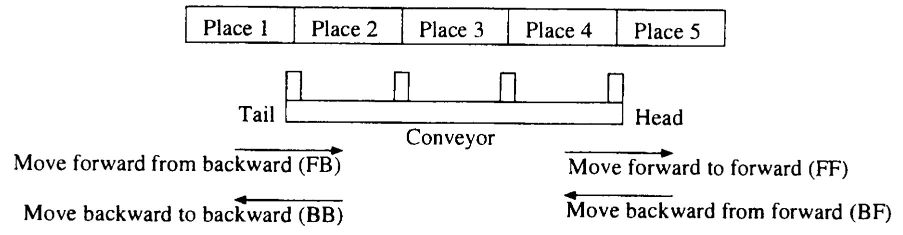

Place 1
Place 2
Place 3
Place 4
Place 5
Tail
Head
Conveyor
Move forward from backward (FB)
Move backward to backward (BB)
Move forward to forward (FF)
Move backward from forward (BF)

Figure 5.7: Conveyor

A conveyor is a sequence of processing places. Objects on a conveyor moves simultaneously by one movement event. Movements of a conveyor can be defined as follows.

Move forward to forward: Objects on Place 2, 3 and 4 move to Place 3, 4 and 5, respectively.

Move backward from forward: Objects on Place 3, 4 and 5 move to Place 2, 3 and 4, respectively.

Move forward from backward: Objects on Place 1, 2 and 3 move to Place 2, 3 and 4, respectively.

Move backward to backward: Objects on Place 2, 3 and 4 move to Place 1, 2 and 3, respectively.

5.2.4 History Management Model

The history management model described below generalizes the multidimensional views presented in the previous chapter, based on the mapping operations among time, places and objects.

5.2.4.1 Status of Places and Objects

Places and objects can be associated with time-varying status values acquired by sensors, such as temperature, humidity and length. Status of places and objects can be represented by time series of data and is denoted by the following notations.

Status of a place: Status of place p_i at the time t_l is denoted by $\sigma_{p_i}(t_l)$.

Status of an object: Status of object o_k^{cj} at the time t_l is denoted by $\sigma_{o_k^{cj}}(t_l)$

5.2.4.2 Mapping Operations

To extract the relationships among time, place, and object, the following mapping operations can be defined.

Place/Object mapping $Time(p_i, o_k^{cj})$

Place/Object mapping obtains a set of time points when object o_k^{cj} existed in place p_i.

Place/Time mapping $Object(p_i, t_l)$

Place/Time mapping obtains a set of objects existing in place p_i at the time t_l.

Object/Time mapping $Place(o_k^{cj}, t_l)$

Object/Time mapping obtains a set of places where object o_k^{cj} existed at the time t_l.

5.2.4.3 Views

In order to provide retrieval functions of historical data, views for time, place and object are defined based on the mapping operations. t_s and t_e denote the starting point and the ending point of a time interval, respectively. t_p is a period for obtaining each tuple of the views.

Time View: $V_{time}(t_s, t_e, t_p)$ is a set of tuples

$$(t_l, p_i, o_k^{cj})$$

where $^\forall t_l \in \{t_s, t_s + t_p, t_s + 2t_p, \cdots, t_s + (\lfloor \frac{t_e - t_s}{t_p} \rfloor) t_p\}, {}^\forall p_i \in \{p_1, \cdots, p_M\}$, $^\forall o_k^{cj} \in Object(p_i, t_l)$.

Place View: $V_{place}(p_i, t_s, t_e, t_p)$ is a set of tuples

$$(t_l, o_k^{cj}, \sigma_{p_i}(t_l), \sigma_{o_k^{cj}}(t_l))$$

where $^\forall t_l \in \{t_s, t_s + t_p, t_s + 2t_p, \cdots, t_s + (\lfloor \frac{t_e - t_s}{t_p} \rfloor) t_p\}, {}^\forall o_k^{cj} \in Object(p_i, t_l)$.

Object View: $V_{object}(o_k^{cj})$ is a set of tuples

$$(t_l, p_i, \sigma_{p_i}(t_l), \sigma_{o_k^{cj}}(t_l))$$

where $^\forall t_l \in Time(p_i, o_k^{cj}), {}^\forall p_i \in \{p_1, \cdots, p_M\}$.

The time view provides relationships among places and objects for each time point $t_s, t_s + t_p, \cdots, t_e$. The place view provides the status of the specified place and the status of objects passing through the place during the time interval from t_s to t_e. The object view provides the status of the specified object and the status of places where the object passed through.

5.2.4.4 Lineage of objects

When an object passes through places other than processing, an object is *reborn* as new objects of other classes. For example, an object $o_1^{c_1}$ of class c_1 may be split into other objects $o_1^{c_2}, o_2^{c_2}, o_3^{c_2}$ of class c_2 at a dividing place. When object $o_k^{c_j}$ enters a place and object $o_{k'}^{c_{j'}}$ is delivered from the place, it is called that object $o_{k'}^{c_{j'}}$ is *derived* from object $o_k^{c_j}$. A set of objects from which $o_{k'}^{c_{j'}}$ are derived is denoted by $parent(o_{k'}^{c_{j'}})$ and a set of objects which are derived from $o_k^{c_j}$ is denoted by $child(o_k^{c_j})$.

By applying the parent and child relationship to object $o_k^{c_j}$ recursively, all ancestors and descendants of the object can be obtained. A set of objects which result in being derived from $o_k^{c_j}$ and objects from which $o_k^{c_j}$ is derived can be obtained by the following procedure.

1. Set an initial object $o_k^{c_j}$ to a set O.
2. For every object $o_{k'}^{c_{j'}} \in O$, add $child(o_{k'}^{c_{j'}})$ to O. Repeat this until no more object is added.
3. For every object $o_{k'}^{c_{j'}} \in O$, add $parent(o_{k'}^{c_{j'}})$ to O. Repeat this until no more object is added.

A set of objects obtained by the above procedure is called *a set of lineage objects* for object $o_k^{c_j}$ and denoted by $lineage(o_k^{c_j})$.

In the flow from raw materials to final products, classes of objects vary at places other than processing. By obtaining the object view for all objects in $lineage(o_k^{c_j})$, production history from raw materials to final products can be retrieved.

5.2.5 An Example

An example of representing a manufacturing line is described here, utilizing a sample manufacturing line shown in Fig.5.8.

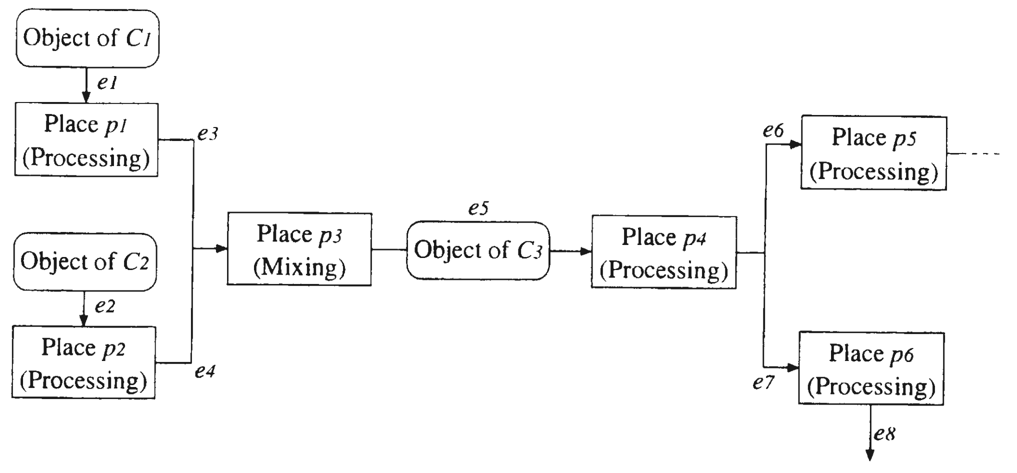


Figure 5.8: An Example of a Manufacturing Line

The sample manufacturing line consists of six places $p_1, p_2, p_3, p_4, p_5, p_6$. Places p_1, p_2, p_4, p_5, p_6 are processing places and place p_3 is a mixing place which mixes objects from p_1 and p_2. An object existing in p_4 has two candidates of destination places, p_5 and p_6, for its movement.

Three object classes C_1, C_2, C_3 are defined in the example. Mixing place p_3 generates an object of C_3 from an object of C_1 and an object of C_2. The object form is defined as the point model for all object movements.

Each object movement $\{e_{move}, p_{from}, p_{to}, obj\_form, C_{next}\}$ for the object classes can be defined as follows. Note that two movements from p_4 are defined because the example line has a branch in p_4.

- C_1

  $\{e_1, NULL, p_1, \text{point model}, NULL\}$

$\{e_3, p_1, p_3, \text{point model}, NULL\}$

$\{e_5, p_3, p_4, \text{point model}, C_3\}$

- C_2

  $\{e_2, NULL, p_2, \text{point model}, NULL\}$

  $\{e_4, p_2, p_3, \text{point model}, NULL\}$

  $\{e_5, p_3, p_4, \text{point model}, C_3\}$

- C_3

  $\{e_6, p_4, p_5, \text{point model}, NULL\}$

  $\{e_7, p_4, p_6, \text{point model}, NULL\}$

  $\{e_8, p_6, NULL, \text{point model}, NULL\}$

Next, suppose that object movements occurs in the following sequence.

1. Object o_1^{c1} enters place p_1 at the time t_{l_1}.
2. Object o_1^{c2} enters place p_2 at the time t_{l_2}.
3. Object o_1^{c2} moves from place p_2 to place p_3 at the time t_{l_3}.
4. Object o_1^{c1} moves from place p_1 to place p_3 at the time t_{l_4}.
5. Objects o_1^{c1} and o_1^{c2} becomes object o_1^{c3} and o_1^{c3} moves from place p_3 to place p_4 at the time t_{l_5}.

Examples of the mapping operations for the object movement history described above are shown below.

Place/Object mapping $Time(p_2, o_1^{c2}) \rightarrow [t_{l_2}, t_{l_3})$

Place/Time mapping $Object(p_3, t_{l_4}) \rightarrow \{o_1^{c1}, o_1^{c2}\}$

Object/Time mapping $Place(o_1^{c2}, t_{l_2}) \rightarrow \{p_2\}$

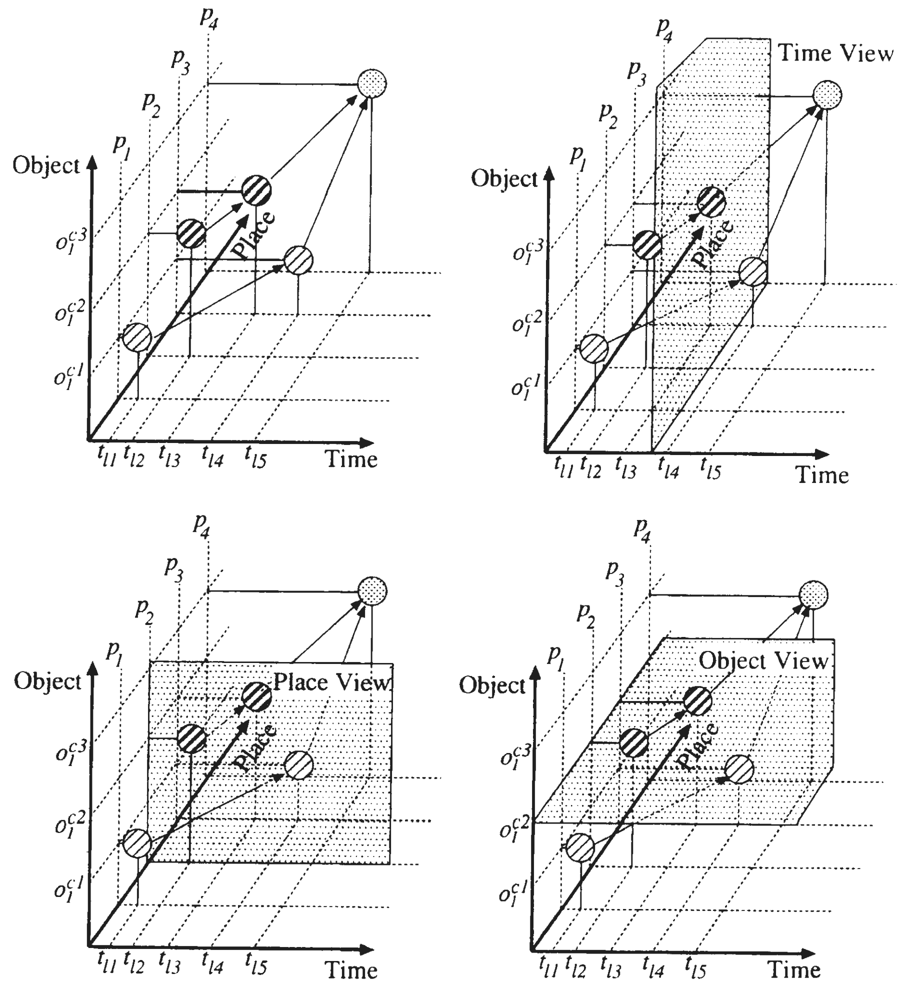


Figure 5.9: An Example of Time, Place and Object Views

Examples of views for the object movement history are shown in Fig.5.9 by the three dimensional representation (place p_5 and p_6 are not included because the inclusion makes the figure so complex). Intuitively, each view can be regarded as a cross plane cut by one of time, place and object axes.

$lineage(o_1^{c_3})$ is $\{o_1^{c_1}, o_1^{c_2}, o_1^{c_3}\}$. Production history for raw materials $o_1^{c_1}$ and $o_1^{c_2}$ to final product $o_1^{c_3}$ can be traced by obtaining object views for objects in $lineage(o_1^{c_3})$.

5.3 Implementation

Based on the Time/Place/Object model described in the previous section, a family of middlewares has been implemented as a development environment of tracking and history management systems. The middlewares are organized by the following components as shown in Fig.5.10.

1. Development tools

   - A builder
     Using this tool, a system designer defines address assignments of acquired data, schema of data acquisition, conditions of event detections, schema for event transmission, organization of a manufacturing line and movement of manufactured objects.
   - A source code generator
     According to the information defined by the builder, source codes specialized for a target plant are automatically generated.

2. Run-time environments

   - An active real-time temporal database (RTDS)
     This component achieves the data acquisition from control networks, storage of them to disks, detection of events and data provision to the RTVS.

   - A temporal object-oriented database (RTVS)
     This component achieves object persistence, provision of temporal objects and transaction management.
   - A tracking manager
     This component achieves tracing of objects and invocations of ECA rules. A part of source codes of this component is generated by the source code generator.
   - An application library
     This component provides library functions which can be used in applications for a target plant such as ECA rules, operator station control and communication with business systems.

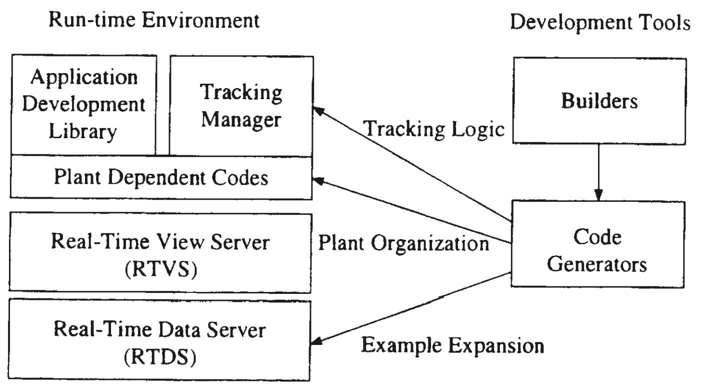


Figure 5.10: Organization of the Middlewares

5.3.1 Development Tools

Development tools run on PCs with Microsoft Windows. As described before, development tools include a builder by which a system designer defines a target plant and a source code generator which generates source codes according to the builder definition.

Among the components in the run-time environment, the RTDS and the tracking manager are customized for a target plant property. Builders and source code generators for the RTDS and the tracking manager have different concepts from each other, considering the difference of defined informations.

For the RTDS, system designers need to input the following informations about the target plant.

Definition of data acquisition Addresses of data on a control network and labels for the data are defined. Types of data acquisition from the control network are periodical reading from a distributed shared memory and eventual reception of messages.

Definition of event detection Conditions for detecting events from acquired data are defined. Conditions can be the edge detection of a bit signal, linear inequality and so on.

Definition of schema for provided data Data structures for periodical transmission of acquired data and eventual report of detected events are defined.

On the other hand, informations for the tracking manager are as follows.

Definitions of the target plant organization Places in a target plant are defined. Places can be defined in hierarchy and a plant can be divided into some sections.

Definitions of production units Classes of objects in a target plant are defined. A class can be associated with schema for production recipes and results.

Definition of the target plant behavior Movements of objects are defined. Trigger of movements used in this definition can be chosen from events defined for the RTDS. ECA rules associated with movements are also defined.

An example screen of the builder is shown in Fig. 5.11.

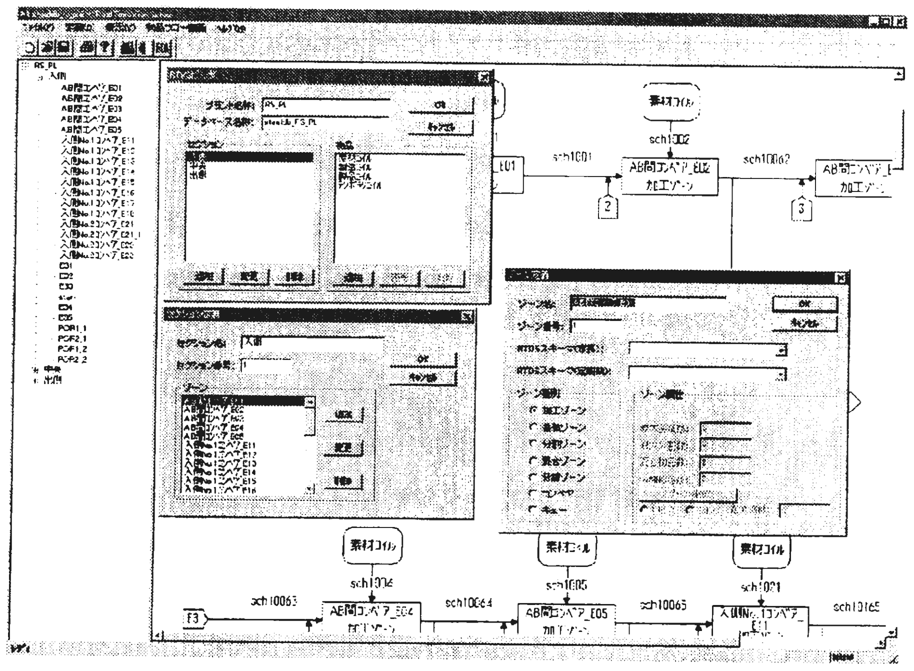


Figure 5.11: An Example Screen of the Builder

The generators generate source codes customized for a target plant according to these definitions. A mechanism of the generator for the RTDS is different from that for the tracking manager, due to the difference of characteristics.

The generator for the RTDS is based on the method of the *example expansion*[31] which expands minimum but functionally complete examples according to the definitions. Source codes for data acquisition, event detection and data provision are automatically generated by expanding pre-registered example codes.

The generator for the tracking manager has a different mechanism from that for the RTDS. Source codes for the tracking manager have two parts: a target dependent part and a target independent part. Codes for the target independent part are fixed, while those for the target dependent part are generated by the generator. The target dependent part includes invocations of object movements and ECA rules, access functions to the defined plant organization and object properties.

5.3.2 Run-time Environments

Run-time environments work on a kind of UNIX which has a real-time capability. Run-time servers are based on the RTDS and the RTVS which have been already implemented before.

Time points in the Time/Place/Object model are represented by the `timeval` structure defined as a standard of UNIX. Places and objects are represented as persistent objects in the RTVS. The data structures for places and objects are shown in Fig. 5.12. An instance of a place has a link attribute storing a reference to an object existing in the place. For places in which multiple objects can exist, the link attribute is able to store the multiple references. An instance for an object has link attributes storing references to objects from which the object has been derived and/or which the object has derived, in order to deal with the lineage of an object. An instance of an object has also attributes storing time stamps when the object has entered and left each places. The time stamp attributes associate time, place and object and make it possible to realize the mapping operations and view generations.

The tracking manager receives events utilizing the event object and invokes the object movement procedure and defined ECA rules. The tracking manager associates instances of places with instances of objects and achieves tracing objects. When an object passes through places other than the processing place, it

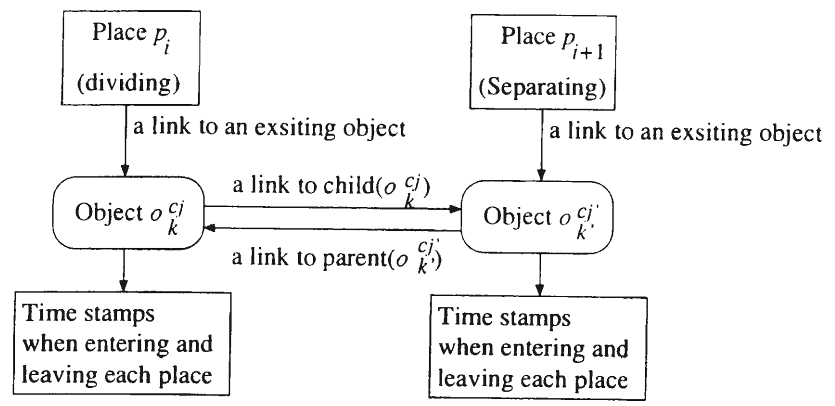

Place p_i
(dividing)
Place p_{i+1}
(Separating)
a link to an exsiting object
a link to an exsiting object
a link to child($o_{k'}^{cj'}$)
a link to parent(o_{k}^{cj})
Object o_k^{cj}
Object $o_{k'}^{cj'}$
Time stamps when entering and leaving each place
Time stamps when entering and leaving each place

Figure 5.12: A Data Structure for Places and Objects

associates an instance of an object with those of derived objects.

The application library includes about 100 library functions by which applications can utilize to get and change the status of the tracking manager, store result data, and obtain the time, place and object views.

5.3.3 Application to a Real System

The middlewares have been applied to a tracking system for a steel process line. This system realizes core functions for steel plant management such as managing production recipes, providing display data for operator stations, issuing preset commands to controllers, gathering result data from controllers and reporting production performance to business systems.

As shown in Fig.5.13, a typical steel process line consists of three sections: 1) entry section in which steel coils are carried by conveyors and mounted to pay-off reels, 2) center section in which steel plates are rewinded from pay-off reels, welded to a proceeding plate and a succeeding one, processed by chemicals and wound to tension reels and 3) delivery section in which steel coils are carried by conveyers and housed to warehouses. The Time/Place/Object model applied to this steel process plant, a pay-off reel can be defined as a dividing place because one coil has multiple production change points when moving from the entry section to the center section. In the center section, a steel plate is cut by a shear which can be defined as a dividing place and a tension reel which rolls up multiple production change points can be defined as an accumulating section. The rest places such as chemical processors can be defined as processing places. In the entry and delivery section, since a steel plate is wound up in the shape of a coil, its object form can be represented by the point model. In center section, since steel coils are rewound and welded to a proceeding coil and a succeeding one, its object form can be represented as the continuous line model.

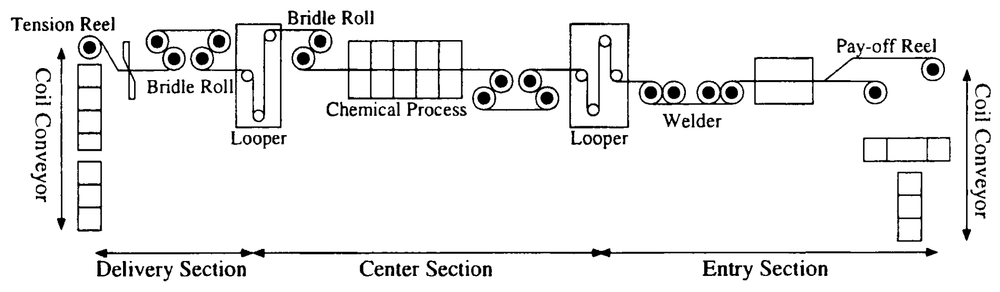


Figure 5.13: A Typical Steel Process Line

ECA rules are utilized to realize the following functions: issuing preset commands to controllers, acquiring and storing result data from controllers, communicating with business computers, comparing the state of the plant model in a computer and that of the real plant by coil sensors and so on.

The scale of this application is shown in Table 5.1. In this application, it took about two days to input the plant information on the builders. Time to automatically generate source codes is only a few seconds. The number of generated source codes is about 31 thousands lines which corresponds to 10 persons/month volume when converted by using widely known production volume 3 thousand

Table 5.1: An Application Result of the Middleware

Num. of places	Num. of classes of objects	Num. of ECA rules	Num. of lines of generated codes
68	4	563	31352

lines/person.month. In addition, specification modifications and design errors could be easily compensated by the builders and the source code generators.

5.4 Summary

In this chapter, in order to represent the organization of a manufacturing line and the behavior of objects, the Time/Place/Object model has been defined. The model defines five primitives of places and three forms of objects. The behavior of objects is defined as a set of object movements through places. Multidimensional views for batch manufacturing information management are defined generally based on the mapping operations and the views.

Chapter 6

Conclusion

Recent break-through of information technologies have brought small but powerful computers to manufacturing management systems. This fact requires that layers of systems should be integrated and have the ability of communicating each other. This thesis discussed about the following issues for the frameworks to achieve this integration.

- Temporal object model based on temporal validity

  In order to represent a monitoring target as an object, a new paradigm for dealing with temporal aspects of objects has been introduced. The model defines three categories of temporal objects and their temporal validity.

  The Real-Time View Server (RTVS) has been implemented based on the temporal object model. This can be utilized as a framework of a database system which has the capability of integrating control information and facility information by the temporal link mechanism.

- A batch manufacturing information management system based on multidimensional view

  In order to deal with several aspects of manufacturing information, a data utilization model based on multidimensional view has been proposed. This model defines three views: Plant View, Unit View and Batch View.

A batch manufacturing information management system has been implemented based on the model. The system is built on the RTVS. An example for a coffee manufacturing plant has been illustrated.

- The Time/Place/Object model

  In order to represent the organization of a manufacturing line and the behavior of objects, the Time/Place/Object model has been defined. The model defines five primitives of places and three forms of objects. The behavior of objects is defined as a set of object movements through places. Multidimensional views for batch manufacturing information management are defined generally based on the mapping operations and the views.

  The family of middlewares including the system development tools have been implemented and these tools can generate runtime source codes for the RTVS. The middlewares have been applied to a real steel mill tracking system and contributed to improving the software productivity. The plant is currently operating successfully.

Of course, the models have rooms to be improved. At this phase, the models are focusing only on industrial computers, but the behavior of systems should be defined from the view point getting all over the system layers. For example, the definition of movement events can be shared with controllers which actually detect the events. At the ultimate stage, all programs for the system layers seem to be able to be generated by defining the behavior of a manufacturing line.

At this time, the Time/Place/Object model is focusing only on manufacturing management, but this model is general enough to apply to other domains. For example, it seems to be applicable to car transportation monitoring and controlling

on a high-way. Extensions to other application domains can be valuable future works.

For the latest trend, even mobile phones begin to join to the components of manufacturing management systems. In this situation, the integration issues will become more significant meanings.

Bibliography

[1] Batch Control, Part I: Models and Terminology; ANSI/ISA-S88.01-1995, The International Society of Measurement and Control, Oct. 1995.

[2] Enterprise - Control System Integration, Part I: Models and Terminology; ISA standard, 1998.

[3] S.Abiteboul, A.Bonner: Objects and Views, ACM SIGMOD Conference, pp.238-247, 1991.

[4] N.R. Adam, A. Gangopadhyay: Integrating Functional and Data Modeling in a Computer Integrated Manufacturing System, Proceedings of International Conference on Data Engineering, pp302-309, 1993.

[5] E.F.Codd, S.B.Codd, C.T.Salley: Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate. E.F. Codd and Associates, 1993.

[6] C.J.Date: An Introduction to Database Systems, *Addison-Wesley*, 1990.

[7] U.Dayal, et al.: The HiPAC Project: Combining Active Databases and Timing Constraints, SIGMOD Record, Vol.17, No.1, Mar. 1988.

[8] K.R.Dittrich, S.Gatziu, A.Geppert: The Active Database Management System Manifesto: A Rulebase of ADBMS Features, International Workshop of Rules in Database Systems, Athens, Greece, pp.3-17, Sept. 1995.

[9] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, Proceedings of International Conference on Data Engineering, pp.152-159, 1996.

[10] M. Gyssens and L.V.S. Lakshmanan: A Foundation for Multi-Dimensional Databases, Proceedings of Very Large Data Bases, pp.106-115, 1997.

[11] K.Hornsby, M.J.Egenhofer: Qualitative Representation of Change, Proceedings of the International Conference on Spatial Information Theory, pp.15-33, 1997.

[12] Y.Kambayashi: A Document Management System, Application Development System, Springer-Verlag, pp.65-77, 1986.

[13] W.Kim: Introduction to Object-Oriented Databases, The MIT Press, 1990.

[14] P. Koksal, S.N. Arpinar, A. Dogac: Workflow History Management, SIGMOD Record, Vol.27. No.1, pp.67-75, Mar 1998.

[15] K.Kumar, J.V.Hillegersberg: ERP Experiences and Evolution, Communications of the ACM, Vol.43, No. 4, Apr. 2000.

[16] D.R.McCarthy, U.Dayal: The Architecture of an Active Data Base Management System, Proceedings of International Conference on Management of Data, pp.215-224, 1989.

[17] J.Y.Lee and R.A.Elmasri: An EER-Based Conceptual Model and Query Language for Time-Series Data, International Conference on Conceptual Modeling, pp.21-34, 1998.

[18] C.L.Liu, J.W.Layland: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, Journal of ACM, Vol.20, No.1, pp.46-61, 1973.

[19] D. Lomet, et al.: Special Issue on Workflow Systems, IEEE Data Engineering Bulletin, Vol.18, No.1, 1995.

[20] D. Lomet, D.Barbará, el.al: Special Issue on Supporting On-Line Analytical Processing, IEEE Data Engineering Bulletin, Vol.20, No.1, 1997.

[21] G.Özsoyoğlu and R.T.Snodgrass: Temporal and Real-Time Database: A Survey, IEEE Transaction on Knowledge and Data Engineering, Vol.7, No.4, pp.513-532, 1995.

[22] K.Ramamritham: Real-Time Database, International Journal of Distributed and Parallel Database, Vol.1, No.2, 1993.

[23] A. Reuter, F. Schwenkreis: ConTracts - A Low-Level Mechanism for Building General-Purpose Workflow Management Systems, IEEE Data Engineering Bulletin, Vol.18, No.1, pp.4-10, Mar. 1995.

[24] E.A.Rundensteiner, H.A.Kuno, Y.Ra, V.C.Taube, M.C.Jones, P.J.Marron: The Multiview Project: Object-Oriented View Technology and Applications, ACM SIGMOD Conference. pp.555, 1996.

[25] R.M.Sivasankaran, J.A.Stankovic, D.Towsley, B.Purimetla, K.Ramamritham: Priority Assignment in real-time active databases, The VLDB Journal, pp.19-34, 1996.

[26] R.T.Snodgrass: The Temporal Query Language TQuel, ACM Transactions on Database Systems, Vol.12, No.2, pp.247-248, 1987.

[27] R.T.Snodgrass: The TSQL2 Temporal Query Language, Kluwer Academic Publishers, 1995.

[28] J.Clifford, C.Dyreson, T.Isakowitz, C.S.Jensen, R.T.Snodgrass: On the Semantics of "Now" in Databases, ACM Transactions on Database Systems, Vol.22, No.2, pp.171-214, 1997.

[29] H.Shimakawa, et al.: Acquisition and Service of Temporal Data for Real-Time Plant Monitoring, Proceedings of the Real-Time Systems Symposium, pp.112-119, 1993.

[30] H.Shimakawa, G.Ido, H.Takada, Y.Asano, M.Takegaki: Active Transactions Integrated with Real-Time Transactions, Proceedings of Third IEEE Real-time Technology and Applications Symposium, pp.49-59, 1997.

[31] H.Shimakawa, G.Ido, H.Takada, S.Horiike: Expanding Small Example into Large Scale Real-Time Control System, Transactions of IPSJ, Vol.40, No.5, pp.2468-2477, 1999.

[32] A.U. Tansel, et al.: Temporal Databases: Theory, Design and Implementation, Benjamin/Cummings Publishing, 1993.

List of Publications by the Author

Major Publications

1. H.Takada, H.Shimakawa: Software Technology in Industrial Systems III, Journal of Systems, Control and Information, Vol.44, No.11, pp.45-55, Nov. 2000. (in Japanese)

2. H.Takada, H.Shimakawa, G.Ido, S.Horiike: The Time/Place/Object Model for Tracking and History Management in Manufacturing Lines, Transactions of IPSJ, Vol.41, No.6, pp.1799-1810, Jun. 2000. (in Japanese)

3. H.Takada, H.Shimakawa, S.Horiike: The Time/Place/Object Model for Tracking and History Management in Manufacturing Line Control, Proceedings of International Symposium on Databases for Non-Traditional Environments, pp.385-394, Kyoto, Nov. 1999.

4. H.Shimakawa, G.Ido, H.Takada, S.Horiike: Expanding Small Example into Large Scale Real-Time Control System, Proceedings of Asia-Pacific Software Engineering Conference, pp.486-493, Takamatsu, Dec. 1999.

5. H.Shimakawa, G.Ido, H.Takada, S.Horiike: Expanding Small Example into Large Scale Real-Time Control System, Transactions of IPSJ, Vol.40, No.5, pp.2468-2477, May 1999. (in Japanese)

6. H.Takada, H.Shimakawa, Y.Asano, M.Takegaki: A Batch Manufacturing Information Management System Based on Active Mechanism and Multi-dimensional View Generation, Transactions of System, Control and Information, Vol.10, No.11, pp.575-584, Nov. 1997. (in Japanese)

7. H.Shimakawa, G.Ido, H.Takada, M.Takegaki: Active Transactions Integrated with Real-Time Transactions According to Data Freshness, Proceedings of the Third IEEE Real-Time Technology and Applications Symposium, pp.49-59, Montreal, June, 1997.

8. H.Takada, H.Shimakawa, Y.Asano, M.Takegaki: A Batch Manufacturing Information Management System Based on Multidimensional View Generation and Active Mechanism, Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications, pp.70-77, Kyoto, Dec. 1996.

9. H.Takada, H.Shimakawa, Y.Asano, M.Takegaki: Production Information Management for Batch Manufacturing Plants Based on ECA Mechanism and View Generation, Proceedings of the Workshop on Databases: Active and Real-Time, pp.77-81, Rockville, Nov. 1996.

10. H.Takada, H.Shimakawa, Y.Asano, G.Ido, M.Takegaki: A Database Management System Based on an Object Model with Temporal Validity for Plant Monitoring Applications, Transactions of IECIE, D-I, pp.853-862, Oct. 1996. (in Japanese)

11. H.Takada, K.Munakata, M.Takegaki: A Real-Time Replicated Data Management Method on Distributed Databases for Industrial Applications, Proceedings of Advanced Database System Symposium, pp.27-34, Tokyo, Dec. 1994. (in Japanese)

12. H.Takada, Y.Kambayashi: An Object-Oriented Office Description Model and an Office View Management Mechanism for Distributed Office Environment, Proceedings of the Forth International Conference on Foundations of Data Organization and Algorithms, pp.362-377, Chicago, Oct. 1993.

Books

1. Y.Kambayashi, M.Arikawa, T.Kunishima, S.Konomi, H.Takada and Y.Miyauchi: Distributed and Cooperative Media Series 4 – Hypermedia and Objectbase, published from Kyoritsu Pub. Co. Ltd., Nov. 1995 (in Japanese).

Technical Reports

1. H.Takada, H.Shimakawa, S.Horiike: Realization of a Hot Standby System for Monitoring and Controlling Manufacturing Lines, Technical Report of IEICE, DE98-44, pp.49-56, Dec. 1998. (in Japanese)

2. H.Takada, M.Takegaki: A Time/Place/Object Model for Production and Flow Management, Technical Report of IEICE, DE96-87, pp.79-84, Jan. 1997. (in Japanese)

3. H.Takada, H.Shimakawa, M.Takegaki: A Temporal Object Model with Valid Interval Based Temporal Validity and Its Applications, Technical Report of IPSJ, 96-DBS-109-25, pp.147-152, Jul. 1996. (in Japanese)

4. H.Takada, H.Shimakawa, Y.Asano, M.Takegaki: Utilization of the Active Mechanism and View Support Functions for Manufacturing Management Databases, Technical Report of IPSJ, 95-DBS-104-32, pp.249-256, Jul. 1995. (in Japanese)

5. An Implementation of the Real-Time Replication Management Middleware for Industrial Distributed Database Systems, Technical Report of IEEJ, IP-94-46, pp.89-98, Dec. 1994.