Towards Ubiquitous Maintenance - Defining Invocation of Plant Maintenance Agents in Real Workspace by Spatial Programming

Hiroki Takahashi and Osamu Yoshie

Waseda University
Hibikino 2-7
Wakamatsu-ku
Kitakyushu-city, Fukuoka
Japan

taka@aoni.waseda.jp, yoshie@waseda.jp

Abstract

Recent progress of VR (Virtual Reality) technologies makes it possible to realize the VR space that is synchronized with the real space. We can hereby integrate virtual workspace in which a worker in real workspace can automatically acquire and invoke appropriate plant maintenance agents. We propose spatial programming which is a manner of VR programming technique, locating various place-dependent agents and web information in VR space, and also describe the interface between agent world and real workspace as an application of spatial programming, towards ubiquitous maintenance.

1 Introduction

In manufacturing systems, to acquire maintenance information and to diagnose quickly and appropriately are important. However, most facilities of manufacturer's factory have some secrecy and its maintenance data is prohibited from sending externally. So, remote maintenance system has large difficulty there, especially it is served and shared with various manufacturers. We have presented *virtual community for plant engineers*[1] or *virtual plant* to prevent this problem, by using plant maintenance agents which are served to each factory from virtual plant. The advantage of this mechanism is that user does not need to send diagnosis data to the external and the agent makes diagnosis process on the computer which is in the factory.

Recently, by the progress of VR technologies, we can now realize the VR space on a computer/network that is synchronized with the real space. Then we can build virtual workspace which contains various programs such as agents and which can work with real world. Basically, diagnosis procedures in the plant depend on each target machine, that is, depend on the place. It is very efficient if it becomes available that a worker can automatically get appropriate plant maintenance agent which is required for one's work of its place from our virtual plant. This also means that the automatic agent service also indicates the workers mission to do there, that is to say, he can automatically get appropriate instruction there. We can regard those place-dependent works as declaratively located processes, so introducing Prolog, which is one of the most popular declarative languages, to our system is useful because of Prolog's declarative programming ability.

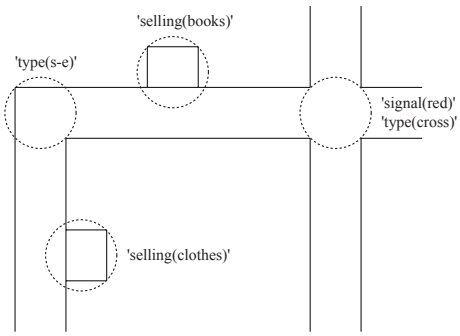
We have also presented VR space description language[2, 3], which is a manner of VR programming technique, locating various place-dependent programs to VR space. This paper enhances the ability of linking VR space to real space and tries to apply spatial programming to define plant maintenance agents' world which works with virtual plant on the web, towards ubiquitous maintenance.

2 Spatial Programming

Spatial programming, the main idea of this paper, is the place-dependent, declarative programming in VR space.

In the real world, it is often the case that our action is basically place-dependent. For example, if we are at the traffic intersection, we acquire the signal information and decide 'stop' or 'go' (Figure 1). But this process is only at traffic intersection, so it is place-dependent. In manufacturing systems, of course, there are many place-dependent information and its processing action such as acquiring diagnosis data and its processing where various agents can flourish, associated with the facility. So, it is natural that which plant maintenance agents are required is place-dependent and user's status-dependent. We noticed this place-

dependency and try to use VR space as the environment for building such an agent world linked to real world. Construction of this agent world begins in plain VR space initially, and constructor locates agent's information (strictly speaking, this is the conditions to summon agents) at appropriate position. We call this manner *spatial programming*.




```
Each information means the following:
  type(s-e)       : Intersection of south and east street
  type(cross)     : Intersection of crossing street
  signal(red)     : Signal is now red
  selling(books)  : Books are sold here
  selling(clothes) : Clothes are sold here
```

Figure 1: Place-dependent Information in Real World

Spatially programmed VR space is presented as 3D virtual space of course, and it can be linked to real space using generic position sensors. This is one of the advantages of spatial programming. Once this link is established, a worker's movement in real space is reflected to VR space. If any place-dependent agent-summoner is located at his position, he can get agents from virtual plant through some devices such as portable PC or PDA (Personal Digital Assistant). Besides it, constructor can also go around in 3D VR space and locate agent-summoner in the space. So it is very intuitive.

Another merit of spatial programming is that it does not need to stop VR space even while agent world's constructor modifies the VR space, because place-dependent description assures independency of descriptions for each object (worker, facility, etc.) in the VR space, so modification of one object does not give influence to other objects' description.

In the following sections, we describe how to realize the idea described above and the implementation of place-dependent agent world in VR space.

3 Model of Place-dependent Processing

In the spatial programming, the place-dependent agent-summoner is embedded in the virtual space as virtual object. Agent invocation process is the interaction between embedded summoner object and worker. We named this embedded summoner object *junction*. And we regard worker and facility as kind of objects, which

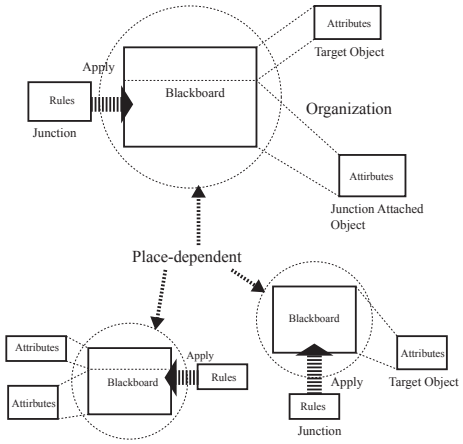


Figure 2: Local Blackboard Model

have entities in real space. Our VR system can grasp the position of objects (especially workers) in the real space via position sensors and establish link between object in VR space and entity in real space. When a worker in real space approaches the place where some junction is located in VR space, the mission of agent happens. This corresponds to the interaction between object and junction which contact each other in VR space. Then some mechanism of interaction is required.

Basically, a facility or worker in the manufacturing system has some status. In this paper, we express these things as objects' attributes (e.g. state of 'it is vibrating', state of 'he has a thermometer' and so on). A request for agent also depends on its and his attributes. Agent summoning process has two phases. At first, interaction process must refer to objects' (both worker's and target facility's, but in some case one of them) attributes and must decide which agent he requires, and next, delivers it to the worker.

At the former phase, we adopt blackboard model[4] for this attribute information exchange. Each object has peculiar attributes. Object's attributes are organized to a blackboard, and junction's decision rules that decide which agent should be retrieved will be applied to it. An interaction occurs when an object comes within the effective range of a junction, and is place-dependent and object-attribute-dependent. So, if interactions occur here and there at the same time, different blackboards should be required (Figure 2). If the organization of blackboard is finished, decision process starts and rules in the junction are applied to the organized blackboard. Each of satisfied rules in decision rules simply enumerates the required agents. The result of decision which is a set of indicators to required agents is written to the object's peculiar attributes temporarily, and then process goes to the next phase.

The latter phase, how to use agent, is object-attribute-dependent, that is, agents' actual invocation

process varies due to the worker's situation. For example, if a worker has a sensor processing data is passed from the device, and if a worker should get processing data via file on computer it should be read and passed from file. So, object also has rules how to use agents which are acquired. And in this phase, target object's attributes which contain the result of interaction as temporary attributes are written to the blackboard, and the rules are applied to it. As the result of this, instructions for worker are composed and sent to the worker's device, and agents are appropriately summoned and do expected process on the worker's device. This invocation process is executed for each object.

The advantage of local blackboard mechanism is that description of object and junction has high-level independency because object/junction-dependent description is apart from each object's and junction's description.

We can say place-dependent work is like declaratively located process. In fact, description of attributes on blackboard and decision rules in junction are written declaratively. We adopt Prolog[6] to describe them. Prolog is one of the most famous declarative programming languages, and is also used for artificial intelligence, so it is very useful for our system. Of course, Prolog's whole ability is too great and our demand is not so high, but in the future, its potential is very attractive.

Most important ability of Prolog in this system is pattern matching. It is not necessary to consider how to solve variable's actual value. Furthermore, we only need to declare facts and rules. The order of declaration is not important, and Prolog automatically solves matching value of unknown variable. This is very useful because we only need to declare place-dependent conditions and other conditions, but need not program how to judge if the conditions are satisfied or not.

We also adopt Java language to process actual invocation of agents. Since Java is very strong in network programming, agents are written in Java on the virtual plant. So, it is proper to use Java in our system.

4 Description of Object and Junction

In this system, description of objects, junctions and the space is basically written in XML (eXtensible Markup Language)[8].

XML is the meta-markup language for text documents. It can be used for various fields with user defined tags, and XML documents can contain various kinds of descriptions freely, such as Prolog programs. So, we can use XML as description language of VR (which is also real) space and extend it also to be containing description of interaction rules in Prolog and description of actual agent's control in Java.

Description of spatially programmed agent world

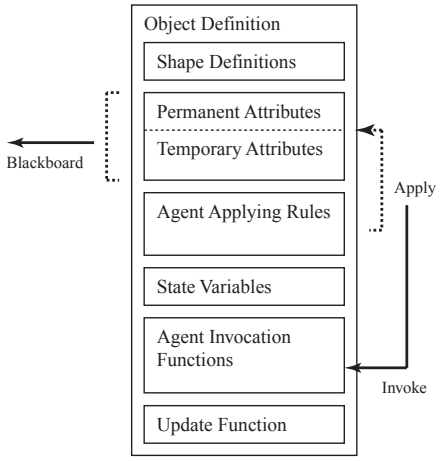


Figure 3: Description of Object

contains two kinds of definitions. One is of objects in wide sense, such as object and junction located in the space. The other is of VR space itself in which objects and junctions are located. This section describes about definition of object and junction. Definition of VR space is described in the following section.

4.1 Description of Object

Object has the entity in the real space. So, some basic attributes are naturally possessed by the object such as shape, position, velocity and so on.

Object also has the attributes which are described declaratively and can be referred from junction. Attributes are classified into two types: permanent attribute, and temporary attribute. Permanent attribute is never cleared and always has some kind of value. Temporary attribute can be removed if it is no longer necessary yet, so this is mainly used to pass the pointer of required agents to the object.

Description of object has the following blocks (Figure 3):

1. Shape Definitions
2. State Variables
3. Permanent Attributes
4. Temporary Attributes
5. Agent Applying Rules
6. Agent Invocation Functions
7. Update Function

'Shape Definitions' block develops object's shape. It is described by combination of primitives.

'State Variables' block consists of state variables such as position, velocity, temperature and so on. These are

used in agent invocation functions or by the whole VR system.

'Permanent Attributes' and 'Temporary Attributes' block contain declaratively described attributes of the object. Temporary attributes are writable from the outer, and permanent attributes are only changed from agent invocation functions. Description language of those blocks is Prolog.

'Agent Applying Rules' block is to decide how to use the agents, and 'Agent Invocation Functions' block does preprocess of applying agents and compose and send instructions to the worker's device or facility's controller in the real world. The device receives those instructions and actually summons required agents and invokes them obeying the instructions. The functions are written in Java[7]. Retrieved maintenance data should be often processed by in the facility's and worker specific way, so it is proper that this part of composition of instructions is implemented in object, not in junction.

'Update Function' is a function which updates object's state in VR space corresponding to the object in the real world every moment. Detail of this function's role is described in the section 5.

4.2 Description of Junction

Description of junction has following blocks (Figure 4):

1. Type and Range Definition
2. Agent Summon Rules
3. Interaction Functions

Junction can be classified into two location types and has two types of effective range.

One is 'Fixed' location type junction. This type of junction is fixed in the space, and is interested in only one object which comes into effective range. Another location type is 'Attached' type. This type of junction is attached to specific object, and move with the object. This type of junction is interested in two objects. One is attached object and another is the object which comes into effective range. So, this type can be used as interaction between objects. Usually diagnosis is done by a worker with target facility, so the latter type of junction will be often used.

Effective range is basically defined by radius. If an object comes into range, junction is activated and indicates the agent summoning instructions to target object. Effective range can be defined as 'contact'. Only attached type junction can have this type of range. If an object contacts another object which has an attached contact type junction, then the junction is activated and indicates the agent summoning instructions to target object. In this case, junction's targets are both objects.

Junction has decision rules that decide which agents should be retrieved and used by the object (strictly

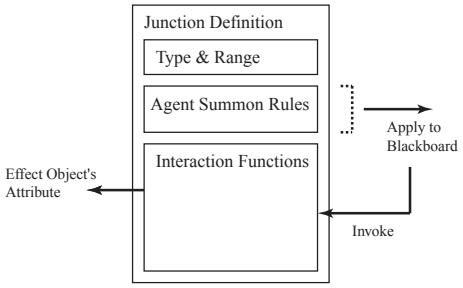


Figure 4: Description of Junction

speaking, by the worker's device in the real world). The decision rules are described in the 'Agent Summon Rules' block. These rules are applied to organized blackboard which consists of the target objects' attributes (if target is only one, organized blackboard has same contents as the object's attributes) when the interaction is occurred.

After the decision process is completed, appropriate function which composes indicator of the required agents and basic instruction (which should be arranged later by object specific way) of using them according to the result of decision. The functions are defined in 'Interaction Functions' block.

It is allowed that no agent will be summoned. This is used to give massage such as warning to target object.

5 Interaction Process between Object and Junction

The time of VR space is synchronized with real space and follows real time. To synchronize, objects in the VR space must update their state every moment. Basically in this system, system activates each object's update function in every time interval. So each object can update its own state (e.g. position, equipment etc.) and can communicate with entity in real space (e.g. worker with portable PC) every moment.

System also monitors physical relationship of objects and junctions and can detect that an object is within the range of some junction or contacts another object with junction. Process of system obeys following flow:

```
Do loop
{
  1. While any object exists in junction's
     range (that is, the junction is
     activated).
  {
    1.1. Target object's attributes are
         written to blackboard and junction's
         rules are applied to it.
    1.2. Junction invokes appropriate function
         corresponding to the result of inquiry,
         which gives required agents and its
```

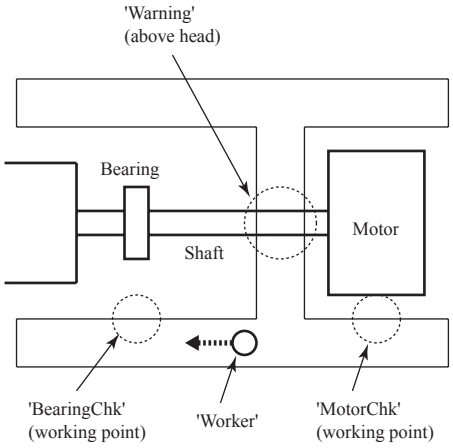


Figure 5: Sample World

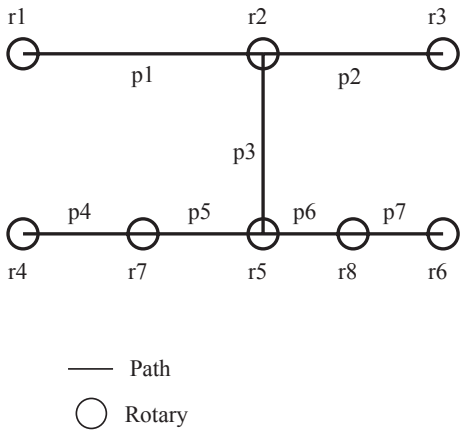


Figure 6: Sample World's Space Description

```
        instructions to temporary attributes
        of objects.
    1.3. Then, the object applies its rules to
        its attributes. Appropriate function
        in the object is invoked and composes
        final instruction of agents and send it
        to the device in the real world.
    1.6. Object updates its own state and
        removes temporary attributes.
  }
  2. do loop until all objects' actions
     are completed
  {
    2.1. Object updates its own state and
        does other required action.
  }
  3. Clock of VR space is progressed.
}
```

6 Defining and Constructing VR Space

6.1 Description of VR Space

Now, we can define objects and junctions in the space for spatial programming. In this section we describe the structure of the space appearing in manufacturing systems where we and objects can go around, and how to define it.

Our VR space fundamentally consists of path and rotary (Initially, no object and junction is located). Path is a straight way that an object corresponding to a worker or other movable object can move along. Rotary is a spot where an object can turn around. Rotaries are connected with path. Rotary is also used as a spot where a worker can access to a facility, because a worker who wants to operate it will turn around there. Each rotary and path has a unique name for system's identification.

For example, consider a plant which has a motor. There are bearing, motor, shaft and so on in this sample factory. Significant facility is motor, bearing and shaft. Motor and bearing require periodic and unexpected maintenance, and shaft is danger when a worker across below it. So, motor and bearing has working point of diagnosis as junction which is named 'MotorChk' and 'BearingChk'. Shaft has 'Warning' junction which warns to worker across below. The facilities' location is indicated in Figure 5.

At first, user should prepare the structure of the space such as path and rotary. Then, the VR space will be defined as Figure 5.User can navigate in this space, and can locate object and junction. Next, user should locate objects and junctions in the space to construct VR information world.

6.2 Construction of VR space

If defining space is completed, we can navigate in it. The next step is locating objects and junctions in it. User should locate objects of facilities in this factory such as motor, bearing, shaft and so on. Locating objects of facilities are completed, then user defines and attaches junctions to appropriate objects. In this space, junction 'MotorChk' is attached to 'Motor', junction 'BearingChk' is attached to 'Bearing'. Next, user should defines fixed junction 'Warning' and locate it to appropriate position, crossing the shaft and path (Figure 7). Last, user should locate worker in the space, which is avatar of worker in the real space.

The interaction process should be designed the following, for example. We consider an interaction between worker and bearing. The interaction is diagnosis of bearing. If the bearing has trouble and the worker has enough equipment to maintain it, the interaction offers plant maintenance agents and its procedure to him. Corresponding to the bearing's vibration trouble, we define the attributes and rules of objects and junction as below.

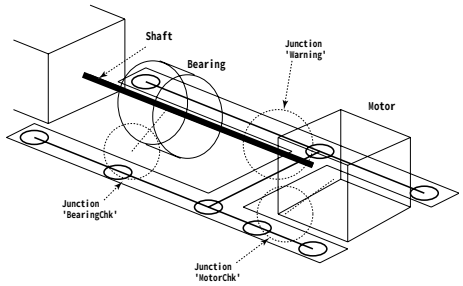


Figure 7: Composed Sample Information World

Object 'Worker': Attributes:

```
equip(thermometer).
```

Object 'Bearing': Attributes:

```
sensor(vibration).
```

Junction 'BearingChk': Rules:

```
exec(diag) :- obj1(equip(thermometer)),
              target(sensor(vibration)).
exec(diag) :- obj1(equip(thermometer)),
              obj1(equip(vibration)).
```

The bearing makes self-diagnosis in the update phase, but if the diagnosis cannot be completed because of lack of any information (e.g. sensor data which is not equipped on the bearing usually), the bearing calls worker who can observe it. (Of course, if the diagnosis is completed, the bearing does not need to call anyone.) Then, if a worker approaches bearing and becomes in range of 'BearingChk', interaction occurs. Inquiry 'exec(X).' answers X=exec(diag). Function diag() is invoked and it composes instructions of bearing diagnosis as:

```
proc([temperature,[vibration, normal,
                   fft, peaksearch], diag]).
diag(bearing).
```

This instruction is written to the worker's temporary attributes. Next, worker does its action. The worker's agent summon rules are applied to its attributes, and inquiry 'exec(X).' answers X=diagBearing. If the worker is only casual passing there, the inquiry is failed and nothing happens. Then worker invokes function diagBearing() and compose and send actual instructions of invocation sequence of agents to the worker's device in the real world, and diagnosis process begins. This is a brief introduction of interaction design.

Thus, constructing VR space is completed. In this way, the VR space is work as spatial programming environment and now, it can also work as plant maintenance agents' world.

In the next section, we describe how to present the VR space to the user.

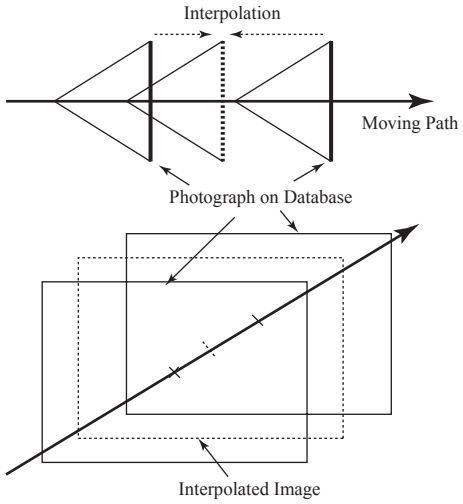


Figure 8: Image Based Rendering

6.3 Spatial Programming Environment with VR

Our VR space uses images each of which is associated with specific path or spot to present realistic 3D space. The feature of our VR system is that the real space and the VR space are linked and synchronized. So navigating in this system should be made in realistic environment. Traditional ways to display VR space use 3D modeling, texture mapping and so on. But those methods cost very much. This also causes that rendering process requires very strong machine power.

We adopt the image based rendering[9]. Image based rendering is developed to reduce modeling cost for 3D objects in the space. This method uses a set of real images alternatively and displays image interpolated from those images (Figure 8). So, image database is very large but this can save human's labor efficiently. This method is especially effective for our VR system with spatial programming, because the number of critical spots or ways is not so big and our movable ways are limited. There is no need to prepare such a large database of images.

User can construct VR space while walking around in it. If he wants to locate an object or junction, what is necessary is to click the button on the browser and input the description of it. Of course, the description can be also loaded from specified text file. After putting it on the VR space, it becomes available to interact immediately without stopping the space.

A user can also interact or receive instructions or message, if user's avatar has appropriate attributes. User can add attribute freely to any object in sight. This function also works for giving instructions to object.

7 Conclusions

This paper describes defining invocation of plant maintenance agents using VR techniques. The basic idea is spatial programming for place-dependent works, one of declarative programming paradigm. This is elegantly achieved by adopt Prolog to implement significant mechanism of this. This idea is realized by interaction between worker object and junction which is located in VR space as interface of giving place-dependent maintenance procedure with agents to the worker in the real world. In this system, the VR space should be linked to real space, and worker in real space can get various required agents and instructions at real time. Spatial programming also makes it possible to achieve this, because user can modify the VR space without stopping or restarting it.

Thus, the VR space which is linked to real space can work as plant maintenance agents' world, and now, we are aiming to realize true ubiquitous maintenance.

References

[1] Osamu Yoshie, Kyoko Iino, Tatsuya Fukunaga, Nobuyoshi Sato "Supplying High-Quality Knowledge of Machine Diagnosis in Virtual Community," Journal of the Society of Plant Engineers Japan, Vol.12, (2001).

[2] Osamu Yoshie, Hiroki Takahashi "VSL-Virtual Space Description Language, and its application to spatial programming," The International Conference on Electrical Engineering, (1999).

[3] Tstsuya Inaba, Hiroki Takahashi, Kyoko Iino, Natsuko Hayashi, Osamu Yoshie, "VSL-Trial for Describing Virtual Space in Logic and Its Application to Remote Robot Operation," International Conference on Applications of Prolog, (1999) pp. 50-53.

[4] Stuart Russel, Peter Norvig, *Artificial Intelligence*, (Prentice Hall, 1995)

[5] Osamu Yoshie, Hiroki Takahashi, Kinjiro Ito, Kageo Akizuki "Building Integrated Homepage by Illustration from Web- and XML- Centric Information Systems," The International Conference on Electrical Engineering, (2002).

[6] Leon Sterling, Ehud Shapiro, *The Art of Prolog. SECOND EDITION*, (The MIT Press, 1994)

[7] *The Source for JavaTM Technology.* Available at http://java.sun.com/.

[8] *Extensible Markup Language (XML) 1.0 (Second Edition).* Available at http://www.w3.org/TR/REC-xml.

[9] E.S.Chen, " QuickTimeVR - An Image-Based Approach to Visual Environment Navigation," Computer Graphics, Proc. of ACM SIGGRAPH95, (1995) pp. 29-38.