AGENT-ORIENTED KNOWLEDGE ENGINES FOR
INTELLIGENT VIRTUAL REALITY-BASED
INDUSTRIAL TRAINING SYSTEMS

By
LAN YE

A Thesis Presented to
The Hong Kong University of Science and Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Philosophy in
Industrial Engineering and Engineering Management Department

Hong Kong, July 1998

Copyright © by Lan Ye 1998
AGENT-ORIENTED KNOWLEDGE ENGINES FOR INTELLIGENT VIRTUAL REALITY-BASED INDUSTRIAL TRAINING SYSTEMS

By

LAN YE

APPROVED:

Dr. Chuan-Jun SU
(Supervisor)

Prof. Mitchell Tseng
(Head of Department, Chair of Committee)

Dr. Vincent George Duffy
(Member)

May, 1998

Department of Industrial Engineering and Engineering Management
AGENT-ORIENTED KNOWLEDGE ENGINES FOR INTELLIGENT
VIRTUAL REALITY-BASED INDUSTRIAL TRAINING SYSTEMS

ABSTRACT

Virtual Environment (VE) has great potential for providing training to users in industry. It has real value for improving the training effectiveness, supporting collaborative training in shared VE, lowering the "transfer barrier" to the real world, and guaranteeing the safety of both human and systems during training. A practical Virtual Reality-based Industrial Training System (VR-ITS) should take advantage of both Virtual Reality (VR) technology and Intelligent Computer-Aided Training (ICAT). While VR technology permits interactive, immersive experience to the trainee, Artificial Intelligent (AI) technology can provide him/her with instructions, by monitoring his/her actions, detecting the errors made by him/her, and evaluating his/her performance. To reach this goal, it is quite important to effectively embed knowledge into an intelligent VR-ITS.

The difficulty of developing usable Knowledge Engines (KE) in VR-ITS is not only that the training domain knowledge involved is quite various and complex, but also that the interaction style between the virtual training systems and trainees is quite different from other traditional training approaches. Although researchers have developed some VR-based training systems, there is no effective approach that incorporates the knowledge and experience of human experts into VR-ITS. In this thesis, by analyzing and classifying the specific characteristics of knowledge in the field, a generic architecture of Recursive and Extended Petri Net (REPN)-based and agent-oriented KE for an intelligent VR-ITS was proposed. Furthermore, according
to the architecture proposed, a task/action-oriented knowledge elicitation skeleton was depicted, and an hybrid knowledge representation pattern was presented. The reasoning mechanism for the proposed KE was also based on REPN. To demonstrate the proposed KE design and illustrate its effectiveness, KE were integrated into the VR-based CNC milling operation prototyping system by following this methodology. In the end, some experiments were performed in order to examine the effectiveness of the developed system.
ACKNOWLEDGMENTS

I extend my sincere gratitude to my supervisor, Dr. Chuan-Jun Su and the head of the department, Professor Mitchell M. Tseng, for their direction and support. Their continuous advice refined my ideas throughout the research. Without their encouragement and support, my thesis could not possibly have been completed.

My earnest thanks are offered to Dr. Vincent G. Duffy, for accepting my request to be in my thesis committee. His valuable suggestions helped me to enrich the content of my thesis. I am also thankful to Dr. Benjamin P. Yen, Dr. Hanqiu Sun at the Chinese University of Hong Kong, and Dr. Fugee Tsung.

I specially give my thanks to all of my friends in the Design and Modeling seminar group and Virtual Reality seminar group, for their comments on my research. Especially, consecutive discussions with Dr. Fuhua Lin helped me a lot for this research work. I would also express my thanks to Ms. Finney Colleen, Mr. Zhixin Yang, and Mr. Lo Wun Tak. They provided me many useful assistance and advice during different periods of this work.

I would like to thank technicians Mr. Chan Wai Keung (Denil) and Tin Kwok Chu for helping me to set up equipment and experiments in the Virtual Reality Application Lab and Manufacturing System Design Lab, and undergraduate students Mr. Kan and Pong for their great help in programming and object modeling. Finally, I would like to thank Ms. Chuqian Zhang for her kindly caring in these two years.
# TABLE OF CONTENTS

Title Page ........................................................................................................................................... i
Signature Page ........................................................................................................................................ ii
Abstract ............................................................................................................................................... iii
Acknowledgements ............................................................................................................................ v
Table of Content .................................................................................................................................. vi
List of Tables ......................................................................................................................................... xi
List of Figures ......................................................................................................................................... x

Chapter

1 Introduction ........................................................................................................................................ 1
   1.1 Virtual Reality Technology and Application ............................................................................ 1
   1.2 Training in Industry ..................................................................................................................... 1
   1.3 Virtual Training System ............................................................................................................. 2
   1.4 Benefits of Virtual Training ....................................................................................................... 2
   1.5 Knowledge Engine - the Indispensable Component for VR-ITS ............................................. 3

2 Objective and Literature Survey ....................................................................................................... 5
   2.1 Research Objective .................................................................................................................... 5
   2.2 Literature Review ....................................................................................................................... 5
      2.2.1 ICAT in Industry ................................................................................................................... 5
      2.2.2 Knowledge Bases in VR-based Training System ............................................................... 6
      2.2.3 Petri Nets in Knowledge Base ............................................................................................. 7
   2.3 Problem Statement ...................................................................................................................... 8
   2.4 Strategies of the Study ................................................................................................................. 9
   2.5 Organization of the Thesis ......................................................................................................... 9

3 Overall Architecture of Knowledge Engine for VR-ITS ............................................................ 10
   3.1 Introduction ............................................................................................................................... 10
   3.2 Overall Architecture of Knowledge Engine for VR-ITS ....................................................... 11
   3.3 Virtual Training Environment .................................................................................................. 14
   3.4 PetriNet-based Engine ............................................................................................................. 15
   3.5 Knowledge for Intelligent Training Task Planning Agent .................................................... 17
      3.5.1 Knowledge on Training Goal Definition .......................................................................... 17
6.1 Conclusion ..................................................................................................................... 63
6.2 Future Work .................................................................................................................. 64

Appendix I: Generic Knowledge Class Definitions in VR-ITS ............................................ 65
Appendix II: Knowledge Acquisition Tables ....................................................................... 68
Appendix III: Usability Analysis ......................................................................................... 79
Appendix IV: Questionnaires for VR-CNC Training Experiments ................................. 93
References ....................................................................................................................... 94
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Usage of knowledge for agents</td>
<td>12</td>
</tr>
<tr>
<td>4.1 Decision table for decomposing goals</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Decision table for refining sub-goals</td>
<td>33</td>
</tr>
<tr>
<td>4.3 Decision table for mapping sub-goals to tasks</td>
<td>34</td>
</tr>
<tr>
<td>5.1 Knowledge acquisition form</td>
<td>51</td>
</tr>
<tr>
<td>5.2 Error detection</td>
<td>52</td>
</tr>
<tr>
<td>5.3 On-line operation function guidance</td>
<td>52</td>
</tr>
<tr>
<td>5.4 CNC machine function tutorial</td>
<td>53</td>
</tr>
<tr>
<td>5.5 An example of decision table for mapping knowledge</td>
<td>55</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3.1 Overall architecture of KE in Intelligent VR-ITS</td>
<td>13</td>
</tr>
<tr>
<td>3.2 A fundamental structure of Petri nets</td>
<td>16</td>
</tr>
<tr>
<td>3.3 Flowchart for training goal definition</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Knowledge categories and trigger mechanism in the ITPA</td>
<td>22</td>
</tr>
<tr>
<td>3.5 Knowledge categories and trigger mechanism in the IPA</td>
<td>24</td>
</tr>
<tr>
<td>3.6 Knowledge representation and inference mechanism for the TEBA</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Knowledge class diagram</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Interaction between an instance of class AbnormalMachineBehavior and instance of class graphical objects and the IPA</td>
<td>43</td>
</tr>
<tr>
<td>5.1 Hardware environment of VR-CNC</td>
<td>45</td>
</tr>
<tr>
<td>5.2 Overall architecture of the VR-based CNC prototyping training system</td>
<td>48</td>
</tr>
<tr>
<td>5.3 The iterative development process of the VR-CNC milling operation training system</td>
<td>49</td>
</tr>
<tr>
<td>5.4 The components of the actual CNC milling machine</td>
<td>50</td>
</tr>
<tr>
<td>5.5 Graphical model of the three-axis CNC milling machine model</td>
<td>50</td>
</tr>
<tr>
<td>5.6 Goal decomposition</td>
<td>54</td>
</tr>
<tr>
<td>5.7 Workpiece</td>
<td>56</td>
</tr>
<tr>
<td>5.8 Final product</td>
<td>56</td>
</tr>
<tr>
<td>5.9 Knowledge in manual operation setting up procedure</td>
<td>58</td>
</tr>
<tr>
<td>5.10 Knowledge involved in task model</td>
<td>59</td>
</tr>
<tr>
<td>5.11 The snapshot of 3D-text instruction in the VR-CNC training system</td>
<td>60</td>
</tr>
<tr>
<td>5.12 Implementation of knowledge in IPA</td>
<td>61</td>
</tr>
<tr>
<td>5.13 Implementation of knowledge for PR</td>
<td>62</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Virtual Reality Technology and Application

Virtual reality (VR) is the use of various computer graphic systems in combination with respective displays and interface devices to provide the effect of immersion in an interactive three-dimensional (3-D) computer-generated environment [Bryson 1995]. VR systems offer reliability, speed, ease of access, compactness, security, and are easily transmitted to other computers located in distant parts of the world. VR technology can create new environments that could only be reached with difficulty in the real world, which allows users to perceive information through synthetic experience.

VR is not so much a technology, but more a mode of computer interaction. Computers are now firmly integrated into every facet of modern society [Vince, 1995], including engineering & architecture, education & training, military simulations, scientific visualization, financial analysis, medical applications.

1.2 Training in Industry

The general objectives of industrial training is to provide trainees with a practical understanding of engineering activities and practices, materials, tools, machines and equipment, appropriate process controls, and the organization of production, testing, installation, commissioning, maintenance and servicing facilities.

The number of people engaged in the manufacturing industry is declining due to the development of automation. Those who remain in the industry need to demonstrate higher skills than the “machine minders” of the past [Warwick, et al., 1993]. Training has become an indispensable routine part of industry. How to provide efficient and cost-effective training is becoming a critical issue. Although intelligent computer-aided training (ICAT) has been used as a training tool for a few years, some apparent shortfalls in ICAT systems are still existing. For instance, trainees often become passive during training sessions; the systems can not give trainees the sense of immersion; there are
fewer interactions between trainees and systems. The trainee has therefore no possibilities to experience a broad range of practices. Transferring the training from ICAT system to reality remains difficult. Obviously, to develop fresh and innovative approaches to meet the requirements of industrial training is an urgent mission.

1.3 Virtual Training System

By definition, the strength of VR technology is to allow users to freely walkthrough and experience situations with a sense of reality, by direct manipulation, visualization, audio, and feeling. VRTS have been subject to a wide range of research: VR technology has a great potential to be applied to training/education systems [Stansfield, et al., 1995 and Brown, et al., 1995]. It is creating new forms of simulations that may lead to fundamental improvements in simulation-based training [Thurman and Mattoon, 1994]. Even in its present early stage of development with its evident limitations, VR is being recognized as a valuable training tool [Cydata, 1996]. Though published results on the efficiency of VR-training technology are rare, existing commercial implementations give convincing evidence of the economical and efficient aspect of the approach comparing to conventional methods [Adams and Lang, 1995; Adam, 1993].

1.4 Benefits of virtual training

In VRTS, trainees will have the same experience as in a real world environment. They are able to participate in the virtual training environment, to operate systems, hear noise, feel force-feedback, and watch the results of their operations. Hence, their motivation will be increased. In the mean time, VRTS enable trainees to spend more time on understanding the training system rather than working out how to use the software [Caird, 1996]. As one kind of VRTS, VR-based industrial training systems (VR-ITS) has the following advantages over the traditional training environment:

1. VR-ITS has real value in safety training [Blotzer, 1995]. Workers could experience industrial accidents without getting hurt as part of their training.

2. Applying VR technology to industrial training enables workers to practice operations before the equipment is installed.

3. VR-ITS are especially valuable in domains where real practice is very expensive.
4. VR-ITS are reconfigurable and portable. It will be easy to extend the systems to different training domain and requirements as well as different hardware equipment.

5. VR-ITS provide collaborative training in networked VEs. Multiple trainees can work at different work sites.

6. To be trained in VR-ITS can lower the "transfer barrier" to the real world [Jackson, 1993].

1.5 Knowledge Engine - the Indispensable Component for VR-ITS

The VR-ITS we proposed are the integration of ICAT and VR technology. First, from ICAT standpoint, artificial intelligence (AI) is the heart of systems. By incorporating ICAT into the systems, VR-ITS can possess domain expertise with knowledge of appropriate training methods, and then act as a domain expert and an experienced trainer who is always energetically paying full attention to the trainees. According to a special training goal, with the support of AI technology, the system can propose tasks suited to the individual trainee. During training, it will watch and evaluate the actions of trainees, provide meaningful comments corresponding to their errors, respond to trainees' requests for information, give hints at any appropriate time, and remember the strengths and weaknesses displayed by trainees so that suitable future training can be designed [Loftin and Savely, 1996].

Second, the construction of a VE involves two kinds of issues. For fully developing the potential of VR to duplicate on-the-job training, and to construct a virtual training environment needs a large amount of intelligence.

Firstly, a sufficient and complete scenario analysis for industrial training is fundamental for building an effective VR-ITS. One of the difficulties to construct useful simulations is that critical aspects of relationships are not necessarily known a priori, nor is the proper sequence of skill components to be taught necessarily known without extensive testing [Gopher et al. 1994, and Fredriksen and White, 1989]. By analyzing and refining training goals, the priori and the proper sequence can be acquired. However, defining training goals, decomposing training goals into tasks, sub-tasks and simple trainee's actions involve a huge amount of developer's intelligence.
Secondly, in the field of VE, modeling is currently one of the most important areas in VE research [Green and Sun, 1995]. Modeling techniques can be divided into two groups, *geometrical* and *behavioral*. Geometric modeling describes the shape of virtual objects as well as their appearance. Behavior modeling deals with how the motion or behavior of these objects can be described, which involves changes of the object in a virtual world in terms of position, collision, grasping, scaling, surface deformation, and also in terms of its physical properties [Burdea and Coiffet, 1994]. Furthermore, traditional simulation techniques deal with the motion of a single object in a controlled environment. In VEs multiple moving objects are represented and the model does not have complete control over the environment. Many special control issues and physical behavior have not been involved in the motion of a traditional simulation, but are important for motion in a VE. To some extent, the machine behaviors are finite, but they are still complicated and sometimes accidents may happen due to the unexpected misoperation of trainees. To build a desired VR-ITS, knowledge about object behavior modeling is a critical issue that must be carefully taken into account. AI technology can be used as one of solutions to dynamically control and change the virtual training environment.

Therefore, in practical VR-ITS, AI technology must be integrated into the systems. Being one of the AI technologies, knowledge engine (KE) or knowledge base systems (KBS) is the most practical approach to satisfy the thirst of VR-ITS for intelligence. KE is the very heart of VR-ITS.
CHAPTER 2

OBJECTIVE AND LITERATURE SURVEY

2.1 Research Objective

Since it is a difficult undertaking to develop practical VR-ITS, and the knowledge in the field of industrial training is various and complex, this study concentrates on proposing how to develop knowledge engines for constructing effective VR-ITS.

2.2 Literature review

KE has been widely applied in the field of training. In this section, the projects for KE (or knowledge base systems (KBS)) application on industrial training and on VR-based training and the related literature are reviewed. The application of Petri nets on KE is also analyzed.

2.2.1 ICAT in Industry

KBSs have been put to practical use in industry for a long time, although in the beginning stage, VR was not involved. Manivannan and Banks (1992) proposed the design principle of a knowledge-based on-line simulation system architecture to control a manufacturing shop floor. A KBS for automated assembly planning was proposed by Tonshoff et al. (1992). Calzada et al. (1993) applying expert systems to safety in chemical engineering plants for consequence analysis, training and planning of operation procedures. Knowledge-based safety training systems (KBSTS) [Shankararaman and Lee, 1994] use model-based reasoning, which limited the lesson to only certain plant models. An interactive graphic environment for numerical control lathe training was developed by Zhang & Chen (1994 and 1992). This system embedded rules to assist instructors and students in teaching and learning. An environment for creating tutoring systems in industrial training, SAFARI, was developed by Gecsei and Frasson (1994). This system is based on four instructional modes: demonstration, exploration, coaching and critiquing, and represents knowledge at two levels - at the physical level
corresponding to a simulated device, and the plan level. Computer aided instruction systems for plant operators [Ujita et al., 1996] provided CAI for novices and experts.

2.2.2 KE in VR-based Training Systems

VR-based training

VRTS have already begun to emerge in various areas, especially in military and medicine: to treat the fear of heights [Hodges et al., 1995], to simulate eye surgery [Sagar, et al., 1994], and to train shipboard firefighting [Tate and Sibert, 1997] and the officer of the deck [Zelter, et al., 1995].

In recent years, some virtual training systems for industrial training came up. For instance, Motorola provided training courses via VR [Adams and Lang, 1995]. For assembling and disassembling various sorts of machines in a hazardous environment, a machine-maintenance training system in VE was developed by Tezuka et al. (1995). The aim of a German project entitled “Multimediale System-Entwicklung (MUSE)” [Burgues and Burgues, 1995] was to give tools to developers of a system that allow them to specify parts of the system and then test the behavior of parts interactively on the screen. Manufacturing Operations in VEs (MOVE) [Wilson, et al., 1995] provided training for maintenance and rapid prototyping for product design. A VR-based training system for CNC milling machine operations was proposed by Lin et al. (1996). Mourant et al. (1996) exhibited a VE for training overhead crane operator. However, all these systems did not adopt KE or KBS yet.

KE in VR-based training

To date, knowledge has been imbedded in many VR-based training systems. Examples include a hybrid expert system prototype for scheduling the U.S. army’s Close Combat Tactical Trainer (CCTT) exploited by Meginis and Phelan (1996); a VE was evolved using KBS to train the Hubble Space Telescope Repair and Maintenance Mission flight team [Loftin and Kenney, 1995]; virtual ECHO aimed at training specialized technical staff in using radar techniques [Meloni, et al., 1996]. In medicine, Thorn et al. (1996) proposed two knowledge bases for anesthetists in surgery in VR, to make inferences from the instrument readings and to prompt and offer advice on possible problems; Dinsmore et al. (1997) used VR for palpation training and provided training quiz to users. VR
technology has also been applied to provide surgical training and certification [Higgins, et al., 1997].

**KE in Industrial VRTS**

Nevertheless, knowledge-based virtual training systems have already been introduced in industrial area. An intelligent tutoring system, which focused on training of technical personnel in the maintenance of electrical devices, was proposed by Bertin et al. (1993). It provided a simulator to evaluate the degree of danger, a didactician to match the trainee’s needs, and the diagnosis of trainee’s errors. A training system using virtual machines for teaching assembling/disassembling operations to novices was evolved by Abe et al. (1996) where the knowledge base was used to instruct a novice to operate a part. A VR operator-training simulator Esope-VR was developed, to train the operation of power transmission and distribution [Okapuu-von Veh, et al., 1996; Garant, et al., 1995; and Garant, et al., 1995]. The KE in this system was to validate the trainee’s operations at all stages of the process and to provide verbal context-sensitive advice when errors were made. A virtual learning environment has been built by Matsubara et al. (1997) to train operator for control activities of an electric power plant. The concept of intelligent training systems adopted in this system was used as a student module that diagnoses the knowledge of the student, and a tutoring module that identifies which knowledge the student should focus on. A four mapping approach for developing VR-based training system has been introduced by [Su et al., 1997]. However, research involving an effective knowledge modeling approach has so far not yet been presented.

**2.2.3 Petri Nets in KE**

Petri nets (PN) theory is a graphical and modeling tool applicable to many fields [Berthomieu and Diaz, 1991; Freedman, 1991; Oeterson, 1981; Rozenberg, 1991; Zhou, 1995]. In AI applications, an original knowledge representation scheme based on Petri net theory was proposed [Ribaric, 1988]. High level nets have been used to model production rules [Giordana and Saitta, 1985; Valette and Bako, 1991]. Deng and Chang (1990) used Petri nets to model both semantic and control knowledge. Petri nets have also been used to verify rule-based systems [Nazareth, 1993], to infer about plans [Murata, et al., 1991; Zhang, 1991], to model logic programs [Murata and Zhang, 1988],
to diagnose knowledge representation and reasoning [Portinale, 1997] and to reason with fuzzy or uncertain knowledge [Cardoso, et al., 1990; Looney, 1988]. Yao (1994) proposed a Petri net model for temporal knowledge representation and reasoning. Castelain and Gentina (1989) used Petri nets to model the scheduling of manufacturing tasks devoted to the low-level control system. Kaddouri et al. (1995) described the method to use object oriented Petri nets to specify the interfaces concerned with different working situations that the operator could deal with. Tezuka et al. (1995) proposed a method for representing assembly and disassembly procedures by using Petri nets. However, no explicit correspondence with VR-ITS has been proposed.

2.3 Research Challenges

From the above survey of past literature and related projects, we can point out that although enormous effort has been done to integrate AI with industrial training systems, little attention has been paid to how AI technology can be effectively applied into VR-ITS. The following subjects that have not been sufficiently covered:

- As it is well known, during the development of KE, the process of extracting the necessary knowledge from a human expert for problem solving remains an extremely tedious, time-consuming and difficult task. Human experts have difficulties to verbalize their problem solving activities [Wenger, 1987]. Knowledge acquisition has proved to be the major bottleneck in KE development. Especially, in VR-ITS, there is no an effective approach to acquire the knowledge and experience of human experts.

- As one of the most important components in KE, inference process will effect the system usability. Exceptionally, there are highly real-time requirements for multi-model and dynamic human-computer interaction in VR-ITS. Inefficiency at speed in searching the knowledge due to large-scale knowledge space increases the interaction time lag. There is no effective inference mechanism for real-time response in VR-ITS.

- As one of the two important concepts in KE, knowledge representation patterns will directly influence the real-time response in complex and dynamic human-computer interaction and the reusability of the system. There is no formal model to represent and coordinate training knowledge for intelligent VR-ITS.
2.4 Strategies of the Study

To fulfill the objective of the study, the following strategies were taken up:

- Recursive and Extended Petri nets (REPN) is used as task modeling to facilitate knowledge acquisition.

- At the same time, REPN also acts as an inference engine to achieve real-time response.

- According to the variety and complexity of knowledge in VR-ITS, a hybrid knowledge representation scheme is developed. Further the KE is constructed by agent-oriented technology.

2.5 Organization of the thesis

In the next chapter, based on multi-agent systems (MAS), the overall KE architecture for VR-ITS will be introduced. The characteristics and use of knowledge in each agent will also be described. Chapter 3 states the task/action-based knowledge extraction first, and then proposes a hybrid knowledge representation scheme for the most optimal organization of different types of knowledge in the system. In Chapter 4, by following the proposed approach, the integration of knowledge into a VR-based CNC milling operation training system is used as a case study. Conclusions are drawn in the final chapter, and at the same time, some recommendations for the future work are suggested.
CHAPTER 3

OVERALL ARCHITECTURE OF KNOWLEDGE ENGINE FOR VR-ITS

3.1 Introduction

The power of knowledge-based systems evolves from the clear separation of the knowledge from how it is used [Laird, et al., 1987]. To develop KE for VR-ITS, we should first understand the problems and major concepts involved. As a consequence, we need to propose the overall architecture for VR-ITS first in order to identify the problem and the specific system goals.

VR-ITS design entails the process of knowledge acquisition, including elicitation of knowledge, modeling the domain, fitting the domain specific knowledge in the model, expressing the knowledge structures in an appropriate knowledge representation form. Knowledge must be extracted from the domain-specific expert through the process of knowledge elicitation. The representation format of the knowledge refers to its internal application within the KE so that it can be used in problem solving.

The elicitation approach and representation format of the knowledge in VR-ITS will be described in Chapter four. However, since there is a large variety of knowledge involved in industrial training, including all of the relevant, domain-specific, problem-solving knowledge in related industrial training, possessed by experts and gleaned by knowledge engineers [Nikolopoulos, 1997], we need to analyze and classify the knowledge for VR-ITS first in order to carry out different representation schemes. The knowledge engineer should not determine which representational mechanism is to be used before analyzing the problem type and domain knowledge. If this step is missed, the building of the conceptual model can be influenced, as the knowledge engineer may try to fit knowledge according to a pre-chosen representation scheme, regardless of suitability, and be tempted to ignore whole sub-areas of knowledge which cannot efficiently be represented in that
scheme. An effective knowledge category scheme will make the development of efficient VR-ITS become possible. In general, for a system, knowledge can be catalogued from different standpoints.

According to the types of expertise in KE, knowledge can be classified into three distinct categories. KE have had varying degree of success when representing knowledge from each of these categories:

- associational (black box);
- motor skill;
- theoretical (deep) knowledge.

The first two types of expertise in industrial training are basic requirements for developing VR-ITS to enable novices to become expert workers. The third type of knowledge should be taken into account when high-level training is provided to professional engineers or scientists. In our study of VR-ITS, we divided all the three types of knowledge according to the features of VR systems and the responsibility of knowledge in the system.

3.2 Overall Architecture of Knowledge Engine for VR-ITS

VR-ITS are simulation in which computer graphics are used to create a realistic looking industrial training world. Moreover, the synthesized world is not static but responds to user inputs (operations, gestures, verbal command, etc.). A key feature of VR system is real-time interactivity. Here real-time means that the VR-ITS are able to detect user inputs and modify the virtual training environment instantaneously. Users should be able to promptly see production line (or system etc.) changes on the screen in response to their commands or operations and hence become captivated by the simulation.

For assisting VR-ITS to provide more effective training, KE will help the system to define training goals according to the training requirements and performance of trainees; assist in proposing challenging and dynamic training scenarios to trainees; monitor the behavior of trainees; respond to the requests of trainees for information or demonstration; give hints if appropriate; and evaluate the performance of trainees. As one of the
supporting mechanism, KE models the abnormal behavior of the complex systems in VE, which can be a solution to overcome the complexity problem in Petri nets representation and enhance the realism of the VE.

The VR-ITS proposed is a multiple agent system (MAS). Three main agents in the system are an intelligent pedagogical agent (IPA), an intelligent task plan agent (ITPA), a trainee behavior evaluation agent (TBEA). The responsibility of each agent is different. They perform individually, but cooperate through intercommunication. Figure 3.1 illustrates the overall architecture of KE for VR-ITS.

The detailed information about the overall architecture of VR-ITS and how to develop it has been proposed by Lin (1998). This research focuses on KE, the back-up mechanism of VR-ITS. Knowledge in agents has different appearance characteristics and functions as illustrated in Table 3.1. The following sections will detail them separately.

Table 3.1 Usage of knowledge for agents

<table>
<thead>
<tr>
<th>Knowledge Title</th>
<th>Usage</th>
<th>Knowledge Agent</th>
<th>Support Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGMB</td>
<td>Provide instruction for next step in machine behavior model</td>
<td>Instruction Provider</td>
<td>IPA</td>
</tr>
<tr>
<td>OGTP</td>
<td>Provide instruction for next step in task plan model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPMB</td>
<td>Provide introduction or demonstration of system function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPTP</td>
<td>Provide introduction or demonstration for tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSMB</td>
<td>Detect the wrong operation in machine behavior model</td>
<td>Troubleshooting</td>
<td></td>
</tr>
<tr>
<td>TSTP</td>
<td>Detect the wrong operation in task plan model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMB</td>
<td>Model the abnormal machine behavior caused by trainees' misoperations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD</td>
<td>Help define individual training goal</td>
<td>Training Task Provider</td>
<td>ITPA</td>
</tr>
<tr>
<td>Mapping</td>
<td>Help decompose goal to a set of tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>Help dynamically judge which routine a trainee will take</td>
<td>Dynamic Plan Recognition</td>
<td></td>
</tr>
<tr>
<td>PE</td>
<td>Help evaluate the performance of trainees</td>
<td>Trainee Model</td>
<td>TBEA</td>
</tr>
</tbody>
</table>
Figure 3.1 Overall Architecture of KE in VR-ITS
3.3 Virtual Training Environment

Inside this computer-based virtual training environment, trainees can experience the same kinds of sensations as in real world training [Wittenberg, 1995]. As it is well known, humans intensely rely on visual perception [Mitchell, et al., 1997]. The visualization component has thus become the most important part of VR. In VR application, visualization is achieved through computer graphics [Stuart, 1996]. Graphic objects that allow trainees to directly interact within VR-ITS will comply with the physical principle. The trainees' current information will be captured through some special devices, such as Dataglove and Polhemus, and then be taken as the input of the VE. In the opposite direction, the trainees can also monitor the systems' situation through the VE's output by head-mounted displays (HMD), 3-D audio systems. The other three agents can exchange the corresponding information with the environmental agent during the whole training procedure. Sound or audio effects provided by the environment allow trainees to hear the special effects of different situations. In addition, the sound also represents one of the instruction providing tools.

3.4 Petri Nets-based Engine

Real-time reaction is one of the main requirements in providing a virtual training environment. Assisting VR-ITS to react promptly and appropriately, and to foresee trainee's actions in the environment is one of the basic requirements for knowledge. The pattern of knowledge representation and the inference mechanism, which are able to manipulate the knowledge stored in KE, will influence the reactive speed of systems. In the proposed VR-ITS, we use Petri nets as the inference engine to retrieve the stored knowledge in run time.

However, why did we select Petri nets as an inference engine in VR-ITS? As it is well known, Petri nets are a useful and powerful modeling tool. C. A. Petri (1962) proposed the methodology as a general purpose-modeling tool for asynchronous systems. The main characteristics of Petri nets are the following:

- Petri nets can definitely represent the causality and independence of the event set in the systems.
• Petri nets are suitable to describe nonsequential systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic.

• Petri nets give the same treatment to active and passive components and, above all, provide the possibility of progressing, at a high level and with any degree of precision.

The approach is especially useful when individuals at work often perform tasks in an asynchronous manner with stochastic properties. The ability to construct models with these properties makes Petri nets an attractive tool for modeling an individual’s behavior. Furthermore, because this methodology provides the capability of modeling parallel activities (and conflicts), aggregates of individual activities (sub-teams and teams) can also be modeled.

However, what are Petri nets? Petri nets are a graphical and mathematical modeling tool. Figure 3.2 summerizes a simple example of Petri nets. It consists of places, transitions, and arcs that connect them. Input arcs connect places with transitions, while output arcs start at a transition and end at a place. Places can contain tokens; the current state of a modeled system (the marking) is given by the number (and type if the tokens are distinguishable) of tokens in each place. Transitions are active components. They model activities which can occur (the transition fires), thus changing the state of the system (the marking of the Petri net). Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled (they are enough tokens available in input places). When the transition fires, it removes tokens from its input places and adds some to all of its output places. The number of token removed/added depends on the cardinality of arcs.

Petri nets are notated in the following way:

\[ PN = ( P, T, I, O, u ) , \]

\[ P = \{ p_1, p_2, ..., p_n \} \text{ is a finite set of places} \]

\[ T = \{ t_1, t_2, ..., t_n \} \text{ is a finite set of transitions} \]

\[ P \cap T = \emptyset \]
\[ I : P \rightarrow T \] is an input function, a mapping from bags of places to transitions
\[ O : T \rightarrow P \] is an output function, a mapping from transitions to bags of places
\[ u : P \rightarrow N \] is a marking, a mapping from places to non-negative integers \( N \)

Figure 3.2 A fundamental structure of Petri net

Moreover, why do we use Petri nets to represent training tasks in VR-ITS? Since in an industrial training system, scenarios are very complicated, how tasks are represented will determine the effectiveness of system representation and will also directly affect the design of KE. To answer this question, let us take into account the features of industrial training systems first:

In industrial training systems,
- There is definitely causality between tasks and the nature of each task is distinct. For example, the action of drilling a workpiece involves two tasks; the tool colliding with the workpiece and material removal. The task collision and material removal are indistinct, but have identified causality.
- The relationship among tasks is nonsequential. Sometimes, it is nondeterministic, and/or stochastic. Sometimes, some tasks can be activated concurrently and some actions are repeatable.
- The operation tasks and their relationship can be described at various levels of detail.
Therefore, using Petri nets to represent industrial training system will be an effective method. At the same time, in VR-ITS, it will also be promising to manage objects in virtual training and knowledge by Petri nets.

Since the characteristics in industrial fields are various, the Petri nets may be different to one another. However, when applied to VR-ITS, they will possess some basic common features. The common character of places in VR-ITS means some objects (or attached objects) in some state and a token indicates that the object in the state, which corresponds to the place, exists. The general character of transitions is defined by an action of trainees or one time interaction between trainees and systems. All types of knowledge are related to tasks, trainees' actions or decisions are linked to the corresponding transition or control place in task planning. From KE standpoint, Petri nets act as an engine. An appropriate rule will be activated instantaneously, when the corresponding transition is fired and all of the aspects of the rule are satisfied. With Petri nets representation, the searching space and time will be dramatically decreased. Hence trainees can get real time feedback.

3.5 Knowledge for Intelligent Training Task Planning Agent

The responsibility of an ITPA is to propose the most suitable training plan to individual trainees in accordance with the training goals and trainees' performance records. A training plan in VR-ITS is a set of dynamic training tasks to be performed. To form a training plan, first a training goal must be defined in response to the requirements of trainees and their performance history, taking into consideration the trainee's current skill. Then the training goal is refined by mapping procedure from goals to sub-goals and then to tasks. The final training plan will be adjusted dynamically in response to the trainees' operations during the training procedure.

3.5.1 Knowledge on Training Goal Definition

Helped by training goal definition knowledge, VR-ITS can automatically propose a training goal for individual trainees. At the beginning of the training session, trainees are required to input their individual profiles (name, sex, age, post etc.), their training objectives and some special requirements if existing. At first, the virtual system will
check if the trainee has used the system before. If not, the system will proceed to ask the
trainee if he/she is a novice or an expert. According to the status and the skill level of a
trainee, VR-ITS will check if the training goal that the trainee required is acceptable. For
instance, if a novice wants to learn high level operation or maintenance skills, it will be
unacceptable.Besides, if a trainee does not claim his/her training objective, system will
provide a few alternatives so that the trainee can select one.

If a trainee was trained in the virtual system before, the VR-ITS will load the personal
file of the trainee that holds his/her profile and performance history, and will judge
his/her skill level to recommend a suitable training goal. The system may prompt the
trainee for his/her performance history if he/she wants. In addition, for any training
tasks, if the performance of a trainee does not exceed a requisite standard, he/she must be
trained again. Figure 3.3 demonstrates us the flowchart of goal definition process.

3.5.2 Mapping Knowledge

Mapping knowledge for decomposing goals to sub-goals, sub-goals to lower sub-goals
and then to task models, and lastly, task models to action models, will facilitate the
procedure of training task planning and make the training task more suitable to the
individual's capability.

Decomposition is necessary because systems in industry are usually very complex. A training goal is
usually a general description of what trainees desire to accomplish. Decomposition allows suitable
training tasks to be singled out. If trainees can satisfyingly complete these training tasks,
they are considered to have achieved the requirement of the training goal. The process of
decomposition allows a high level training goal to be breakdowned step by step, into
tasks that are executable by trainees. We call the tasks at this level actions.

*Mapping knowledge from goal to sub-goals and from sub-goals to lower sub-goals*

These two types of knowledge will help systems to refine training goals. The first step is
to decompose a training goal into several independent sub-goals, and then the sub-goals
into lower level sub-goals, until reaching the extent where each sub-goal can be directly
supported by several training tasks.
Figure 3.3 Flowchart for Training Goal Definition
Mapping knowledge from sub-goals to tasks

Task models will be directly related to a dedicated Petri net representation in VR-ITS. Task models ought to be well defined and constructed during the system development period. The definite relationship between training task models and Petri nets representations should continue to exist in VR-ITS after it is developed.

When a training goal is fixed, the mapping knowledge can help the system to select a sufficient number of suitable task models to in turn satisfy the requirements of the training goal. VR-ITS are able to provide the most suitable training scenario to trainees of different level. Even though different level users may require the same training goal, VR-ITS are able to propose individualised training scenarios according to their respective levels. For instance, suppose there are two trainees, one a senior manager and the other a novice operator, with both of them willing to learn the workshop operations. If the manager does not specify that he/she wants to learn the operation in detail, the virtual system may provide the senior manager with a high level training environment with key operation information only, or the virtual system will only provide him/her a with demonstration. However, a novice operator will automatically be presented with more detailed basic training tasks.

Mapping knowledge from task model to action model

In proposed VR-ITS, we use Petri nets to represent training tasks. Conceptually, each definite task model is related to an existing Petri nets representation. The connection between task models and their corresponding Petri nets representation is fixed for the developed VR-ITS. The ‘action’ in VR-ITS refers to a trainee’s action or his/her interaction with the virtual system. Each ‘action’ corresponds to a transition in the Petri nets representation. The type of knowledge regarding how to map task model to action model, is only taken into account during the design and construction of VR-ITS. Once the system has been developed, we do not need it anymore.

3.5.3 Plan Recognition Knowledge

After a training goal is specified, a skeleton task plan will be automatically generated by the system. A training task plan is a sequence of actions which when executed will result

- 20 -
in the trainee reaching his/her objective. In the proposed VR-ITS, by following a series rule, some process plans can be proposed to achieve the corresponding training goal at varying levels of abstraction. In other words, the proposed task’s Petri nets representation will entail a few task process plan models. When the trainee executes tasks along one of the models, he/she will be able to complete the training task and achieve his/her goal in the end. The detailed plan contains the route, actions, machining parameters, machines, tools required for production and other operation parameters. For instance, in generic training for part machining, process planing is the act of preparing detailed machining operation instructions to transform the design to a final functional workpiece. For each design entity one or more candidate operations may exist. However, during training tasks performing, planing problems include: where the trainee wants to be, his/her goal, and reasoning about how he/she is going to achieve those goals and his/her actions.

The knowledge about plan selection is linked to control places, which takes charge of monitoring the execution of plans. The trainee may discover that part of the plan is prevented and he/she must therefore seek an alternative solution. Trainee’s actions are much more unpredictable. The system will not only judge if the action taken by the trainee may cause errors, but will also dynamically judge which routine the trainee is taking. Knowledge about inferring intentions from actions linked to control place supports the dynamic process planning generation. Plan recognition should be explored in a range of domains. According to the analysis of trainee’s behavior as intentional activity, appropriate advice or help and related error detection may be given. In the proposed VR-ITS, the IPA is very much concerned with what its goals should be. There is a degree of correspondence between the TroubleShooting model, InstructProvision model and DynamicPlanRecognition model.

Figure 3.4 illustrates the flowchart of knowledge involved in the ITPA.
Figure 3.4 Knowledge categories and trigger mechanism in the ITPA
3.6 Knowledge for Intelligent Pedagogical Agent

The intelligent pedagogical agent (IPA) in VR-ITS provides necessary and effective teaching to trainees so as to facilitate training procedure. It acts as a full-time expert trainer who is always energetically ready for tutoring trainees whenever needed.

The IPA receives information about the current states of each object entailed in the training (including trainees and virtual systems), such as if trainees have reached impasses or if they have asked for assistance and so on, from VE and other agents. Based on the state information, the IPA responds to trainees through various methods, for example, by providing instructions, hints, error diagnosis, and recommendations through visual or audio, and by demonstrating the operation required by trainees and even the disaster scenarios caused by trainee's misoperations.

The control engine in the IPA of the proposed VR-ITS is based on Petri nets model. Since in the IPA, there is an overall PetriNet-based representation about the training scenarios, and it can continuously receive perceptual input, and further it possesses the ability to monitor trainee’s activities and provide the trainee with instructions in terms of his/her current situation. Abnormal machine behavior knowledge will also be triggered by the firing of the transition that represents the wrong actions in VR-ITS for demonstrating the corresponding situation in reality. Adopting this strategy has greatly improved the performance of teaching procedure. Figure 2.5 summarizes us the knowledge categories and trigger mechanisms in the IPA.

3.6.1 Intelligent Tutoring

The instruction provided by the IPA can be classified into two main forms: task-driven instruction and impasse-driven instruction. They are separately prepared by the instruction provision model and the troubleshooting model. Since the proposed IPA can continuously monitor the attributes of each object in the environment, a trainee’s action is watched and matched with the current training plan. The effects of the action on systems are evaluated to determine whether the action was completed.
Figure 3.5 Knowledge categories and trigger mechanism in the IPA
There are four types of instruction knowledge in the task-driven instruction mode of IPA. The first two types of knowledge (OGMB and OGTP) are on-line guidance knowledge, which can provide instruction to a trainee as to what is the next step that he/she should take. OGMB stores the knowledge for machine behavior model, including basic machine operations and the related system response, and OGTP for task planning model. The knowledge in the later one can tell trainees the required parameters and tools for a specified task. The last two types of knowledge are the explanation of the function of systems or operations (FTMB) and training tasks (FTTP). Task-driven knowledge is provided to trainees whenever required.

The troubleshooting model will manage impasse-driven instruction. When trainees select the impasse-driven instruction mode, the system will let them operate the machine using their own knowledge and experience learnt before. It will not prompt any help message during the training period until he/she runs into a trouble. Moreover, the mistake made by the trainee will be recorded. When a trainee makes a incorrect operation or even reaches impasses, the IPA can discover them, interpret the impasse and suggest repairs to the procedure that will overcome them. The resulting explanation is used for tutoring the trainee. There are three types of impasse-driven knowledge in the IPA, which are related separately to the misoperations in machine behavior model (TSMB and AMB) and task planning model (TSTP) as well. However, well developed VR-ITS will permit trainee's unmatched operations with task plan, if the trainee does not rush into impasse, so that he/she can draw a novel solution for a task or draw a lesson from an accident.

3.6.2 Machine Abnormal Behavior Model

Machine Abnormal Behavior Knowledge models how the machines or systems will respond to an incorrect action. This kind of knowledge will also be managed by the troubleshooting model. When an incorrect action is performed, the related machine abnormal behavior knowledge will be activated automatically so that the trainee can watch the effect of his/her action on machines, production line, workpiece and so on. The ability to demonstrate abnormal behavior will become more important, especially when the action may cause serious results. To separate the machine abnormal behavior
from the normal behavior model will effectively decrease the complexity of the Petri nets representation for an industrial training system. All in all, including the machine abnormal behavior model in VR-ITS brings the system closer to the real world. This type of knowledge will guide system motion in run-time. In addition, output of this type of knowledge will directly influence the objects in VE.

3.7 Knowledge for Trainee Behavior Evaluation Agent

The Trainee Behavior Evaluation Agent (TBEA) is also based on the PetriNet-based training scenario model. By matching the trainee's action with the action transition in the current training plan and machine behavior model, performance evaluation knowledge (Figure 3.6) helps the TBEA to evaluate the trainee's performance. The Petri nets engine in Figure 3.6 is also in REPN model. The evaluation depends on task's completion time, error types and quantities. The TBEA can judge if the trainee reaches the required level of the training goal as well. The achievement status of the individual goals of the current training plan is continuously monitored. It will create a personal file for each trainee and record his/her own performance history. Performance evaluation aims at the weaknesses of trainees so as to provide more challengeable training scenarios to individual trainees until their performance is satisfactory. The TBEA promptly updates the trainee model that is then sent as an input message to the ITPA.

3.8 Summary

In this chapter, an overall KE architecture was proposed for VR-ITS. Furthermore, knowledge was analyzed according to their characteristics and responsibilities in different agents so that we can adopt different representation schemes in order to organize them for future use. In the proposed VR-ITS, trainees will interact with the virtual industrial systems, machines and other objects and experience the same training procedure as in the real world. All in all, the system, with its extensive KE support, will allow trainees to experience an increased amount of reality responses from its VE, thus enhancing the result of training.
Figure 3.6 Knowledge representation and inference mechanism for TBEA
CHAPTER 4

KNOWLEDGE ELICITATION AND

HYBRID KNOWLEDGE REPRESENTATION SCHEME

4.1 Introduction

Knowledge acquisition is the process of collecting the necessary knowledge for problem solving in the relevant industrial training, and to encode it in formalism amenable to efficient computer manipulation. Generally, knowledge that needs to be considered includes domain knowledge and control knowledge.

Knowledge acquisition can be viewed as the composite of two tasks: knowledge elicitation and knowledge representation. In knowledge elicitation, the domain and problem solving knowledge is gathered and structured in a conceptual model. The knowledge engineer uses various ways at his disposal for eliciting the domain and control knowledge, including interviews with the experts, available books, reports and manual references, etc. In addition, various semi-automatic and automatic techniques have been developed to assist the knowledge engineer with the knowledge acquisition task.

The acquired knowledge is converted in a form suitable to computer manipulation in knowledge representation. The processes of knowledge elicitation and knowledge representation are not necessarily sequential. It is usually the case that knowledge elicitation takes place throughout the lifecycle of development, since it may be realized that the knowledge base is incomplete or inaccurate and may need to be modified [Russell and Norvig, 1995].

Knowledge elicitation is the major bottleneck in knowledge base system development [Brown, 1982]. Human experts routinely use mental activities to solve problem. Such problem solving activities are difficult to verbalize or even to become aware of, and to be captured to knowledge base systems. Therefore the process of extracting the necessary knowledge for problem solving from human experts is extremely tedious, time-
consuming and difficult. In addition, knowledge representation and control will directly influence on the real-time response in complex and dynamic human-computer interaction and the usability of the system [Bloom, et al., 1992]. Inefficiency at speed in searching the knowledge due to large-scale knowledge space increases interaction time lag.

How to facilitate knowledge elicitation processes and how to organize knowledge base in the proposed VR-ITS so as to decrease the searching time in knowledge base are two critical issues not only for the design of an efficient KB system, but also for the whole VR-ITS. Nowadays, building VR-ITS requires more thorough understanding about how the system works and how trainees operate and respond to the system than ever before due to a higher expectation from the users. This is the case for both VR-ITS, and, particularly, knowledge-based systems. During the system development, a high quality training scenario analysis (task analysis) will help to understand both system and trainee’s work [Diaper, 1989], and eventually facilitate the development of KB system.

In the proposed VR-ITS, we use task/action-oriented task analysis approaches and hybrid representation schemes to analyze training tasks in order to increase the usefulness and usability of the system and to improve client acceptability and satisfaction. The task/action-oriented task analysis provides us with an approach to overcome the complex and concurrent characteristics of events in industrial training area. From high-level viewpoint, we decompose training goal hierarchically to achieve a related skeletal training plan. From low-level viewpoint, the behavior of physical systems and training tasks are represented by Petri nets. In the mean time, it will facilitate the knowledge elicitation procedure as well.

### 4.2 Task/Action-Oriented Knowledge Elicitation Scheme

To imbed knowledge into VR-ITS, Knowledge elicitation is normally the most time-consuming phases in the development of a knowledge-based system [Gonzalez and Dankel, 1994]. It is also a very important task since the content of the knowledge is what determines to a large extent the quality of the KE.
From KE standpoint, the strengths of using the task/action-oriented approach are that it can provide the context for knowledge elicitation; ensure that the acquired knowledge is required to support task performance; provide richer representation; make maintenance process easier; and reduce questions to domain experts and experienced trainers by focusing on crucial questions [Lee and Foong, 1994].

One of the most difficult aspects of a knowledge engineer's task is to assist the experts to represent and structure the domain knowledge. A conventional approach to knowledge elicitation is realised through a psychological method for extracting general knowledge from experts. However, it is very difficult for general domain experts to articulate their knowledge, because their problem-solving knowledge has already been converted into skills. When an expert does not recognize and arrange knowledge related to problem solving, he cannot express and represent knowledge well. Researchers have put an extensive effort into overcoming the knowledge acquisition barrier [Bradshaw and Booze, 1989; Gaines and Compton, 1992]. However, most of them intended to ask too many questions that bother domain experts. In VR-ITS, we use task/action-oriented approach to enhance the elicitation function, reduce general questions and focus on crucial questions [Chandrasekaren, 1986; Schreiber et al., 1992]. By classifying knowledge into different categories, knowledge engineers can easily define what kind of knowledge should be provided by an individual expert - the experienced worker, the expert trainer or the expert engineer. At the same time, it will help different experts to sort out their knowledge easier by knowledge classification. It will guide the knowledge elicitation process and the knowledge base's structure.

Experts have a tendency to generalize knowledge bases on their experience with special cases. In conventional methods, once this generalization is implemented, the expert has to experiment with other related cases covered by this generalization that might not be considered initially. The feedback loops to specialize on the generalized case to cover other cases that should not be generalized in the last place usually take time and let domain expert lost their patience. However, the task/action-oriented approach will effectively reduce the feedback times and make it easier.
The task/action-oriented approach proposes to define training goals first, and then hierarchically decomposes these goals into sub-goals, refines the sub-goals until the sub-goals can be supported by some independent training tasks. The training tasks can be represented by Petri nets and be further decomposed into sub-tasks step by step. In the end, a whole Petri nets representation can be derived. Knowledge engineers can follow these steps to design different types of knowledge elicitation forms. Domain experts and experienced trainers will fill out these forms step by step. By following the task/action-oriented approach, the corresponded knowledge can be elicited easier than ever. For instance, in the beginning, an experienced trainer may be requested to state the types of training in a special system, such as, what are the basic training and advanced training and how to determine an individual training goal to trainees. Hence, goal definition knowledge is extracted. Secondly, experienced trainers will be asked to design a set of training tasks so that trainees can achieve the requirements of the goal. The mapping knowledge will be obtained during this period. In VR-ITS, training tasks are represented by Petri nets. In the Petri nets representation, each transition means one time action of trainees. The action will cause the change of the system situation. Therefore, the knowledge for the changes— the motion of system — pertaining to the action can be extracted. Here, system includes production line, various machines, workpieces involved in VR-ITS and so on. Even though, conceptually, Petri nets can not only model the trainee’s normal operations, but also their misoperations, and even both the system’s normal and abnormal behaviors that caused by trainees’ operations, the approach to use Petri nets to describe the whole systems will suffer from one of the biggest disadvantages, the representation complexity. We propose to use knowledge base technology to describe the abnormal machine behavior in order to simplify the Petri nets representation and bring the Petri nets approach to be applicable. The VR-ITS will be developed close to the real scenario through the complete description of system’s behaviors. In the meanwhile, the knowledge about how to detect a trainee’s misoperations and when and where to give trainees hints, warnings and instructions, can be elicited by domain experts and experienced trainers. Therefore, VR-ITS can represent a powerful tool for training. In the end, experienced trainers are required to fill in the
form to evaluate a trainee’s performance. This knowledge extraction allows the system to propose individual challengeable training scenarios.

Although Petri nets can represent the whole systems, it will be too complex and expensive to do so. The trade-off between complexity and feasibility is to extract the abnormal machine behavior, the trainee’s misoperations detection, necessary instruction provision and trainee’s performance evaluation into knowledge bases.

4.3 Hybrid Knowledge Representation Scheme

The importance of representing domain knowledge in a more adequate way is emphasized by many researchers [Schreiber, et.al, 1992; Steels, 1984].

Some researchers have used C Language Integrated Production System (CLIPS) to represent knowledge in a VR simulation [Engelberg, 1994]. In VR-ITS, any single knowledge representation will be no longer sufficient, neither for the purpose of system construction nor for knowledge acquisition. In VR-ITS, we adopt a hybrid knowledge representation scheme to overcome the complex and various knowledge, such as, using decision tables to represent the mapping knowledge, using production rules to model abnormal machine behavior and to represent the knowledge of troubleshooting and instruction providing. Furthermore, we used an agent-oriented structure to facilitate the KE implementation.

4.3.1 Decision Table

A decision table is a matrix of rows and columns, rather than a tree, which shows conditions and actions. It is a vehicle for “breaking down” or understanding a problem [Hurley, 1983], and is proposed to represent mapping knowledge in VR-ITS. A decision table states what procedure to follow when certain conditions exist. This method has been used since mid-1950s. It was developed at General Electric for the analysis of business functions such as inventory control, sales analysis, credit analysis, and transportation control and routing.
In general, the decision table is made up of four sections: condition statements, condition entries, action statements, and action entries. Since the mapping knowledge may be in different levels, the corresponded decision tables shall be in multiple layers also. The definitions of decision tables in VR-ITS are as following.

Table 4.1 shows us the decision table for mapping knowledge for goals to sub-goals.

<table>
<thead>
<tr>
<th>CONDITION</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal No.</td>
<td>Goal statements</td>
</tr>
<tr>
<td></td>
<td>Goal entries</td>
</tr>
<tr>
<td>Sub-goal No.</td>
<td>Sub-goal statements</td>
</tr>
<tr>
<td></td>
<td>Sub-goal entries</td>
</tr>
</tbody>
</table>

The *goal statement* identifies the relevant goals. Goal entries tell which goal applies for decomposition. *Sub-goal statements* list the set of all sub-goals that can be taken when goals are requested. Sub-goal entries show what specific sub-goals in the set to take when the selected goal is true. Sometimes notes are added below the table to indicate when to use the table or to distinguish it from other decision tables.

Table 4.2 is the form of mapping knowledge from sub-goals to lower sub-goals. The meaning of relevant column is similar to the goal decomposition table.

<table>
<thead>
<tr>
<th>CONDITION</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-goal No.</td>
<td>Sub-goal statements</td>
</tr>
<tr>
<td></td>
<td>Sub-goal entries</td>
</tr>
<tr>
<td>Sub-goal No.</td>
<td>(Lower) Sub-goal statements</td>
</tr>
<tr>
<td></td>
<td>(Lower) Sub-goal entries</td>
</tr>
</tbody>
</table>

Note
Table 4.3 states the lowest level decision table: which specific task set will be taken, when selected sub-goals or combinations of sub-goals are true.

<table>
<thead>
<tr>
<th>CONDITION</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-goal No.</td>
<td>Sub-goal statements</td>
</tr>
<tr>
<td>Task No.</td>
<td>Task statements</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

The decision tables in VR-ITS are in a Y/N format. Y and N are used in each entry section.

4.3.2 Production Rule

Production rule-based systems are by far the most popular knowledge representation technique. This can be attributed to the excellent ability of rules to represent heuristic knowledge. Production rule-based systems are also easy to implement and to understand. Additionally, production rules are a natural and very common format used by experts to express problem-solving knowledge in many types of domains. It represents knowledge using this *IF – Then* format. The IF portion of a rule is a condition, (also called a premise or an antecedent), which tests the true value of a set of facts. If these are found true, the THEN portion of the rule (also called the action, the conclusion, or the consequent) is inferred as a new set of facts.

Production rules can be used to express a wide range of association. In VR-ITS, using the format - production rules - to represent knowledge involved in task execution will be more natural. Knowledge in this procedure is categorized into three types. They are situation-action (*machine behavior knowledge*), premise-conclusion (*troubleshooting knowledge and instruction providing knowledge*), and antecedent-consequent (*plan recognizing knowledge*).
4.4 Agent-oriented KE Structure

P. Maes defines a software agent as "a process that lives in the world of computers and networks and that can operate autonomously to fulfill a set of tasks" [Chorafas, 1998]. In this research, KE is constructed by using agent technology. It performs the role of "intelligent assistant". This can provide us the following advantages:

- **Abstraction and Encapsulation**: Knowledge agents abstracts out the essential characteristics, which provides a distinct knowledge agent boundary, and then hides the details by encapsulation, which separates the internal implementation of a knowledge agent from its interface.

- **Scalability**: Knowledge agents are equipped with search and filtering capability that run in the background to help people explore vast sources of information.

- **Flexibility and opportunism**: Because they can be instructed at the level of goals and strategies, knowledge agents can react promptly and appropriately to unforeseen events. Knowledge agents can also be instructed to execute tasks at specific times or automatically "wake up" and react in response to system-generated events.

- **Concurrence**: Multiple knowledge agents can run concurrently.

- **Task orientation**: Knowledge agents are designed to take the context of the training tasks and situation into account as they present information and take action.

- **Easy extension**: Design knowledge engine by agent technology smoothes the further extension to networked systems.

- **Simple program**: Inheritance of knowledge agents permit the reduction for the code of programming.

For programming knowledge agents, we want to separate heuristic knowledge in the form of rules from the rest of the system. To package rules in reusable knowledge base objects will let them shared by many intelligent agents [Tracy and Bouthoorn, 1997]. For example, a knowledge base object may contain rules that help to provide instruction; and this object will be shared by perhaps all agents of a specified type, and perhaps by different types of agents that require the instruction knowledge encoded in a knowledge base object.
We can finalize our requirements for a C++ class, KnowledgeBase:

1. Separate heuristic knowledge in the form of rules in reusable objects.
2. Simple notation for specifying rules.

The hierarchy of knowledge class in VR-ITS is illustrated in Figure 4.1.

The Knowledge class is a generic class. From it, we can define derived classes for all types of knowledge in VR-ITS. Its definition is following (List 4.1). MAXIMUM sets the maximum number of transition that needs to be linked to knowledge.

List 4.1

```cpp
// File knowledge.h

#ifndef knowledge_h
#define knowledge_h

#include "agent.h"
#include <stdio.h>

Class Knowledge {
    public:
        Knowledge ( );
    ~ Knowledge ( );

    private:
        const char *trans[MAXIMUM];
);

#endif
```
Figure 4.1 Knowledge class diagram
ITPA-KB, TBEA-KB, InstructionProvision and AbnormalMachineBehavior are the first level derived classes of Knowledge. ITPA-KB is the child class of Knowledge for the intelligent task plan agent. The child class, TBEA-KB, is to represent evaluation knowledge in the trainee behavior evaluation agent. Furthermore, TroubleShooting, OnLineGuide and Tutorial are the derived classes of InstructionProvision. The three derived classes can use constructs already created by the InstructionProvision class, but they are more specialized than their parent are. The definition of classes InstructionProvision and Troubleshooting are shown in list 4.2 and list 4.3. An enumerated constant MAX_TS_RULES sets the maximum number of troubleshooting rules that can be added to an instance of class Troubleshooting.

List 4.2

// File instructionprovision.h

#ifdef instructionprovision_h
#define instructionprovision_h

#include "knowledge.h"
#include <stdio.h>

Class InstructionProvision : public Knowledge {
    Public:
        virtual void remember (const char *trans, rule rl);
        virtual void modify (const char *trans, rule rl);
        virtual void forget (const char *trans, rule rl);
    }
#endif

List 4.3

// File troubleshooting.h

#ifdef troubleshooting_h
#define troubleshooting_h

#include "instructionprovision.h"
#include <stdio.h>

struct RULE_TS {
    const char *trans;
} - 38 -
typedef struct RULE_TS  rule_TS;

Class Troubleshooting : InstructionProvision {
    public:
        void remember (const char *trans,  rule_TS rf);
        void modify (const char *trans,  rule_TS rf);
        void forget(const char *trans,  rule_TS rf);
        int execute_TS(const char *trans,
                        char *constraint[],  rule_TS rf);
    private:
        int num_ts_rules;
        rule_TS   TS_rule[MAX_TS_RULES];
};

#ifdef

For the definitions of class OnlineGuide and class Tutorial, please see the Appendix I.

The class AbnormalMachineBehavior is a derived class of KnowledgeBase. The class IPA-KB is for the representation of knowledge in the IPA, which is multiple inheritance. It derived directly from class Troubleshooting, class OnLineGuide, class Tutorial and class AbnormalMachineBehavior.

    AbnormalMachineBehavior { ...........};

The class of knowledge for abnormal machine behavior is a little different from the above classes that provide users instruction. The output of class AbnormalMachineBehavior will directly influence the status of graphical objects in VE. An instance of a AbnormalMachineBehavior will usually be customized (by adding rule functions) to support a single type of graphic object. However, many instances of a particular type of a graphic object can share a single instance of a AbnormalMachineBehavior, because the AbnormalMachineBehavior class instances contain general knowledge, but contain no specific data pertaining to any particular graphic object (the rule functions are called with a pointer to current some particular instance of C++ class graphic_object).
The AbnormalMachineBehavior will be classified as general purpose utility class because they will work with any definition of class graphic_object. Listing 4.4 shows the C++ class interface for AbnormalMachineBehavior. An enumerated constant MAX_MB_RULES sets the maximum number of C language rule functions that can be added to an instance of class AbnormalMachineBehavior. All rule functions must match the prototype in the following C language typedef:

```
typedef int (*rule_function)(char *constraint[], Graphical_object*);
```

A rule function is expected to provide the following functionality:

- Accept the current state of VR-ITS from the IPA to see if its preconditions for execution are met; the function should immediately return the value of 0 if its preconditions are not met.
- If its preconditions are met, the rule function is allowed to make any changes to through the public C++ interfaces of, the graphic_object instances passed as pointers in the rule function argument list.

```
// File abnormalmachinebehavior.h

#ifndef abnormalmachinebehavior_h
#define abnormalmachinebehavior_h

#include "knowledge.h"
#include <stdio.h>

typedef int (*rule_function)(char *constraint[], Graphical_object*);

Class MachineBehavior : Knowledge {
    public:
        void remember (const char *trans, rule_function rf);
        void forget (const char *trans, rule_function rf);
        void modify (const char *trans, rule_function rf);
        int execute_mb_rule(char *constraint[], Graphical_object*);
    private:
        int num_mb_rules;
        rule_function MB_rule[MAX_MB_RULES];
};

#endif
```

- 40 -
Listing 4.5 shows an example of the implementation of the AbnormalMachineBehavior class.

Class member function remember() adds a new rule function to an internal list of different types rules. Function forget() deletes an existed rule function from the internal rule list. Function modify() modifies an existed rule function in the internal rule list.

Function execute_mb_rule() executes the rule functions which is linked to the fired transition and returns a nonzero value, indicating that rule’s preconditions were satisfied, and that the rule was properly executed.

List 4.5

    // File abnormalmachinebehavior.cpp

    #include <iostream.h>
    #include <stdlib.h>
    #include "abnormalmachinebehavior.h"

    void AbnormalMachineBehavior :: remember(const char *atitle,  rule_function rf )
    {
        int i;
        if (num_mb_rules >= (MAX_MB_RULES-1)) {
            #ifdef TEXT_MODE
                cerr << "Too many rules added to AbnormalMachineBehavior object.
            #endif
                exit(1);
        }
        // Inserting the rule at the end of the base:
        num_mb_rules++;
        rules[num_mb_rules] = rf;
        trans[num_mb_rules] = atitle;
    }

    void AbnormalMachineBehavior :: modify (const char *mtitle,  rule_function rf)
    {
        int i;

        // Find the index for modifying this rule:
        for (i = 0; (trans[i] == mtitle) // ( i > num_mb_rules ); i++)
            if (i > num_mb_rules )
                #ifdef TEXT_MODE
                    cerr << "There is no the rule you want to modify in
                            AbnormalMachineBehavior object. \n";
                #endif

    - 41 -
else {
    rules[i] = rf;
    trans[i] = mbtitle;
}

void AbnormalMachineBehavior::forget (const char *dttitle, rule_function rf)
{
    int i;
    if (num_mb_rules <= 0) {
        #ifdef TEXT_MODE
            cerr << "There is no rule to be deleted in AbnormalMachineBehavior object.\n";
        #endif
            exit(1);
    }
    // Find the index for deleting this rule:
    for (i = 0; i < num_mb_rules) (dttitle == trans[i]); i++;
    if (i <= num_mb_rules) {
        for (j = i; j < num_mb_rules; j++) {
            rules[j] = rules[j+1];
            trans[j] = trans[j+1];
        }
        num_mb_rules --;
    } else {
        #ifdef TEXT_MODE
            cerr << "There is no the rule you want to delete in AbnormalMachineBehavior
object.\n";
        #endif
    }
}

int AbnormalMachineBehavior::execute_mb_rule(char *constrant[], Graphical_object *g)
{
    int still_processing = 0;
    int rule_counter = 0;
    while ((still_processing == 0) && (rule_counter < num_mb_rules))
    {
        still_processing = (*rules[rule_counter])(constrant[], g);
        rule_counter++;
    }
    return 1 - still_processing;
}
Figure 4.2 shows the interaction of an instance of class AbnormalMachineBehavior with the (currently) abstract class graphical_object. When the public member functions execute_mb_rule() is called, it requires pointers to link the constrains provided by the state information from the IPA. These pointers are passed through to C language rule functions, which can both access and modify these data objects through their public class interface, referenced through these pointers.

![Diagram showing interaction between a MachineBehavior object and IPA](image)

Figure 4.2 Interaction between an Instance of Class AbnormalMachineBehavior and Instance of Class Graphical Objects and the IPA

4.5 Summary

One of the most important issues in VR-ITS is to organized knowledge well and adequately in order to provide effective and efficient training. By considering the knowledge category scheme described in last chapter, the knowledge elicitation process is suggested to follow the task/action-oriented approach. A hybrid representation scheme is proposed to overcome the variety and complexity of knowledge in industrial training field.
CHAPTER 5

APPLICATION: A VIRTUAL TRAINING PROTOTYPING SYSTEM FOR CNC MILLING MACHINE OPERATIONS

5.1 Introduction

To date, Computerized Numerical Control (CNC) milling machine tools are adopted widely in manufacturing. The expectation to increase the employment of numerical programmers and CNC machine tool operators is becoming higher and higher. However, the CNC equipment, material, operation and maintenance are very expensive and complex. Furthermore, to set up a CNC machine, adequate space is one of the essential requirements. Even though CNC machine tools are obtainable, a casual user may be restricted because of excessive costs and problems that could result from broken tools or damage machine components and even the user himself/herself. As a result, it becomes difficult to allow users to have on-the-job training experience in this field. When the user has to find some relative operational manual to read instead of obtaining the hands-on experience, virtual training is perhaps an effective alternative.

To fulfill the above challenges, we began to develop VR-based training environment for CNC milling machine operations [Lin, et al., 1996; Hon, 1996]. The ultimate objective is to allow users to acquire the same experiences and knowledge of how to operate a CNC milling machine through practice in an interactive and immersive computer-generated environment.

The virtual CNC milling operation training prototyping system was developed with multi-agent architecture, and corresponding knowledge was modeled in it by following the proposed approach in this research.

The training system built is based on DM2400 High Precision CNC Milling Machine model. The DYNA 2400 is a light (290 lb.) bench top type machine. It can be installed on any bench top capable of carrying its weight. It consists of X table, Y table, spindle, controller, electronic probe and fixtures.
5.2 System Configuration

5.2.1 Hardware Environment

The hardware environment of VR-CNC (Figure 5.1) includes SGI Onyx2 workstation, Acoustetron II 3-D Sound Server, VR4 HMD or CrystalEyes VR LCD Shuttle Glasses, Fastrak position and orientation Tracker.

![Hardware environment of VR-CNC](image)

Figure 5.1: Hardware environment of VR-CNC


*The Acoustetron II™ 3-D Sound Server:* 3-D Sound Effect is taken in VR-CNC milling operations training system using the Acoustetron II from Crystal River Engineering, Inc. Sound represents a largely untapped source of realism in VEs. In the real world, sound constantly surrounds us and pulls us into our world. In VE, sound enhances the immersiveness of the simulation and provides valuable information about the environment [Fouad and Hahn, 1996]. An auditory feedback plays an important role in our cognitive process. In virtual industrial training, we make use of this sensory feedback to make the trainee understand the characteristics of the machines, materials, etc under the normal or abnormal state during the machining process.
3-D Sound Effect in the CNC milling training system is necessary to let the trainees in the system was realized by using the Acoustetron II from Crystal River Engineering, Inc. to enhance the effectiveness of VEs. The Acoustetron II from Crystal River Engineering, Inc. performs real-time spatialization of multiple real-time audio sources. For each audio input, the system produces Left and Right outputs, which are mixed and played through conventional headphones, earphones, or speakers. The processing creates the perception that the source is positioned at any specified location in three-dimensional space. The Acoustetron II is a stand-alone 3-D sound server system that can be controlled via a communication protocol (a RS232 serial connection by default) from any client computer that is capable of implementing the communication protocol. The system consists of four signal-processing cards housed in an industry standard 486DX4 PC host controller. Each card holds a Motorola DSP56001 clocked at 40 MHz, and high-resolution stereo A/D and D/A converters, with input and output sampling rates of up to 44,100 samples per second.

**VR4 HMD:** A Virtual Research VR4 head-mounted display (HMD) was used for viewing the environment. Two channels of a Polhemus Fastrak electromagnetic tracking device tracked the user's viewpoint and the position and orientation of the virtual hand model.

**Fastrak Position and Orientation Tracker:** Position and orientation tracking is accomplished using the Fastrak device, sold by the Polhemus Corporation. This device consists of a transmitter, which sequences through the generation of three orthogonal magnetic fields AC, and one or more receivers. The receivers each consist of three orthogonal coils of wire. In the magnetic field, an electric field is induced in the coils. The strength of this field in each coil, with respect to which magnetic field is being generated, provides six positional measurements for the receiver: the x, y, z location of the receiver with respect to the transmitter, and the roll, pitch, and yaw angles of the receiver with respect to a calibrated set of axes. The particular tracking device was selected because of its low cost and the ease with which the Fastrak device can be driven on a PC platform. The primary limitation with this device is range; the device loses functionality at a range of between 5 and 10 feet between transmitter and receiver. Calibration of the virtual space was done to compensate the limitation.

Polhemus Fastrak, CrystalEyes glasses and Emitter were used to get stereo visual effect. The device controllers for a Cyberglove, mouse were also developed in training system.
5.2.2 Software Environment and Some Techniques

*WorldToolKit*: Sense8's WorldToolKit for UNIX was chosen as main developing tools. This software is in the form of a collection of C-library routines, with some demonstration drivers provided, to generate polygonal representations of graphics objects, determine stereo-scopic projections of the objects in left- and right-eye point-of-view, and incorporate position information from a variety of different tracking devices. Although the library can be used in C language applications, it is more natural to develop object-oriented program to create and manipulate the graphics objects. The library is easily used by C++ drivers.

*3D Studio 4.0*: All the objects were built by using 3D Studio 4.0 in PC. These objects were then converted to WorldToolKit for Unix format.

*Perceptual Cue Tuning Techniques*: Because of device limitations, certain objects in the VE may fail to meet their functional requirements without including compensational scaling, distortions, or other types of perceptual cue tuning [Zeltzer and Pioch, 1996]. Each instance of such cue tuning should be operationally validated through tests with human subjects, using the appropriate displays from the integrated VE system. In our system, the objects such as Control Panel, Spindle Adjuster was placed in the position that is different from the position of real machine.

5.3 Quick Look at the Prototyping System

5.3.1 Overall Architecture

The prototyping system was developed by following the proposed framework [Lin, 1998] and knowledge in it was modeled by following the approach proposed in this research. To overcome the complexity and variety of the intelligent VR-based CNC training system, a multi-agent structure was employed in it. Figure 5.2 illustrates the overall architecture of the system. Different agent takes charge of different function, which leads to a more logical and conceptually simpler detailed internal architecture.

The training task plan agent (ITPA) is in charge of planning training task under the help of task plan base, goal definition knowledge and mapping knowledge. The intelligent pedagogical agent (IPA) provides instruction (by task-driven or impasse-driven) with the supporting of two types of knowledge, on-line guide knowledge and error detection knowledge. The performance evaluation knowledge is in trainee behavior evaluation.
agent that provides performance evaluation for trainees. All of the three agents communicate with other agents defined within the VE.

![Diagram](image)

Figure 5.2 Overall architecture of the VR-based CNC prototyping training system

In the prototyping system, trainees interact with it through multi-sense, visual, audio and haptic, and the position of their head and hand are tracking during training. In addition, a trainee in VEs is considered as a higher level agent that can control the behavioral characteristics of other agents in the VE. He/she can view the behavioral specifications of individual agents and can directly manipulate these agents.
5.3.2 Development of the System

The development of the system began with function analysis for the training system. Then responsibility of each agent was analyzed and the virtual scene was designed according to the functional requirements.

On one hand, after assigning the specified responsibilities to each agent, knowledge extraction process can be activated and followed by knowledge representation schemes. At the same time, the agents in the system can be sketched. After a detailed specification was achieved, programming stage began.

On the other hand, subsequent to the design of VE, developers can model the 3-D objects with sounds. Texture capturing and editing can be done in this period also.

After the above procedures, it is the step to integrate different components. The last step for developing the system is validation and verification through examples and examined by domain experts. The design process is iterative (Figure 5.3).

![Diagram](image_url)

Figure 5.3 The iterative development process of the VR-CNC milling operation training system.
5.3.3 Virtual CNC Milling Machine Graphical Model

The comparison between the real CNC milling machine prototype and the virtual one are shown in Figure 5.4 and Figure 5.5.

![Figure 5.4](image1.png)

Figure 5.4  Components of the actual CNC milling machine.

![Figure 5.5](image2.png)

Figure 5.5  Graphical model of the three-axis CNC milling machine model
The virtual three-axis CNC milling machine contains both dynamic and static virtual objects. For instances, machine tools, spindle, and tool holder are dynamically while the machine base is a stationary object. The XY-table is made up from two solid models: one for the y movement (front/back), the value of y being related to the base; and one for the x movement (left/right), the value of x being related to the y-bed. The chuck of the machine, which holds the cutting tool, can move in the z direction (up/down), and z is again related to the base. The component parts of a machine tool can move, but none ever changes its shape. Workpieces are also represented by solid models. However, as they can be machined, their shapes can be changed.

5.4 Design and Implementation for Knowledge Engine

5.4.1 Knowledge Extraction

Knowledge extraction and representation for CNC milling machine training involved in the whole developing procedure. We extracted the required domain knowledge from the manuals for DM2400 machine [DM2400, 1989] and requesting the experienced worker and trainer. The domain knowledge acquisition form is shown in Table 5.1. The form is composed of four columns. Because we need to relate knowledge to corresponding transition (action/task), the most left two columns record the transition number and task/action name. Some limitations can be filled into the third column. The conclusion part of a rule can be filled in the "message" column.

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task/Action Name</th>
<th>Additional Condition</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to the knowledge categories, we extracted knowledge separately. Table 5.2, 5.3 and 5.4 represent the different knowledge forms for detecting error, providing on-line operation guidance and tutorial for the function of machine. For the explicit knowledge acquisition form, please see appendix II. The item "additional conditions" in the table Error detection is to simplify the Petri nets representation and implementation of the system.
### Table 5.2 Error detection

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task Name</th>
<th>Additional Conditions</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T11</td>
<td>Power on</td>
<td>Press (key) ↔ PO</td>
<td>Wrong! Press the POWER UP button first.</td>
</tr>
<tr>
<td>T12</td>
<td>Initialization</td>
<td>(Press (key) ↔ YES/NO)</td>
<td>Wrong! Press YES/NO please.</td>
</tr>
<tr>
<td>T22</td>
<td>Manual mode selection</td>
<td>(Press(key) ↔ M)</td>
<td>Wrong! Press MANUAL button as you set the goal.</td>
</tr>
<tr>
<td>T25</td>
<td>Program mode selection</td>
<td>(Press(key) ↔ P)</td>
<td>Wrong! Press PROGRAM.</td>
</tr>
<tr>
<td>T221</td>
<td>Tool calibration ?</td>
<td>Press (key) ↔ (YES ∨ NO)</td>
<td>Wrong action! Yes/No.</td>
</tr>
<tr>
<td>T222</td>
<td>Tool calibration</td>
<td>(Press (key) = NO)</td>
<td>Wrong! You failed to achieve your goal, press YES.</td>
</tr>
<tr>
<td>T223</td>
<td></td>
<td>(Press (key) = YES)</td>
<td>Wrong! You failed to achieve your goal, press NO.</td>
</tr>
</tbody>
</table>

### Table 5.3. On-line operation guidance

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task Name</th>
<th>Additional Conditions</th>
<th>Operation Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>T22511</td>
<td>Measurement Selection</td>
<td></td>
<td>Press NO button to escape the current measurement setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Press Yes button to confirm the measurement setting displayed.</td>
</tr>
<tr>
<td>T22513</td>
<td>Metric system</td>
<td>((Press (key) = NO) ∧ (pre_meas = MM) ) ∨ ((Press (key) = YES) ∧ (pre_meas = INCH))</td>
<td>The dimensional system you should use is Metric (MM).</td>
</tr>
<tr>
<td>T22512</td>
<td>English system</td>
<td>((Press (key) = YES) ∧ (pre_meas = MM) ∨ ((Press (key) = NO) ∧ (pre_meas = INCH))</td>
<td>The dimensional system you should use is English (INCHES).</td>
</tr>
<tr>
<td>T22520</td>
<td>X axis feed rate</td>
<td></td>
<td>If you want to use the displayed feed rate, press Next. Otherwise, press CLEAR.</td>
</tr>
<tr>
<td></td>
<td>specification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5.4. CNC machine function tutorial

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task Name</th>
<th>Additional Conditions</th>
<th>Function Tutorial</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>*******</td>
<td>Press the YES button to initialize, the x and y tables and the z axis head will go to the home position, Z first then X &amp; Y.</td>
</tr>
<tr>
<td>T11</td>
<td>Power on</td>
<td>*******</td>
<td>Press the NO key will allow you to move each axis</td>
</tr>
<tr>
<td>T12</td>
<td>Initialization</td>
<td>*******</td>
<td></td>
</tr>
</tbody>
</table>

.press(key) = NEXT

The X axis feed rate has been set down.

Press(key) = CLR

Next Step: you should enter the new X axis Feed Rate using the numeric keys. The format is XX.X.

(Workpiece(type) = Plastics) ∨ (T_tip(FR, X) < 10) ∨ (T_tip(FR, X) > 16))

For plastics, the normal feed rate is from 10 to 16 INCHES per minute.

(Workpiece(type) = Aluminum) ∨ (T_tip(FR, X) < 5) ∨ (T_tip(FR, X) > 10))

For aluminum, the normal feed rate is from 5 to 10 INCHES per minute.

(Workpiece(type) = Steel) ∨ (T_tip(FR, X) < 0.1) ∨ (T_tip(FR, X) > 5))

For steel, the normal feed rate is from 0.1 to 5 INCHES per minute.

(Press_alrea(key)=CLR) ∨ (Press(key) = NK)

After you input the X axis' new feed rate, Press NEXT to finish the X axis feed rate setting.

Press(key) = NEXT

The X axis feed rate has been set down.

(Workpiece(type) = Plastics) ∨ (T_tip(FR, X) < 10) ∨ (T_tip(FR, X) > 16))

For plastics, the normal feed rate is from 10 to 16 INCHES per minute.

(Workpiece(type) = Aluminum) ∨ (T_tip(FR, X) < 5) ∨ (T_tip(FR, X) > 10))

For aluminum, the normal feed rate is from 5 to 10 INCHES per minute.

(Workpiece(type) = Steel) ∨ (T_tip(FR, X) < 0.1) ∨ (T_tip(FR, X) > 5))

For steel, the normal feed rate is from 0.1 to 5 INCHES per minute.
### 5.4.2 Knowledge Representation

#### 5.4.2.1 Mapping Knowledge

To find out the related training tasks that can satisfy the required training goal, we use decision table to represent the internal relationship between goals and sub-goals, or between sub-goals and definite training tasks. Therefore, decision tables may be of multiple levels.

Figure 5.6 demonstrates a decomposition procedure, which decomposes the goal G00 into several sub-goals.
The goal G00 is to have trainee master basic manual operation skills for the CNC machine, but not tool calibration. It can be decomposed into three sub-goals first. Sub-goal SG00_01 is to let trainees master machine initialization. Sub-goal SG00_2 asks trainees to master the basic machining skills. In this example, to mill a slot and drill two holes on the specified position of a workpiece (SG00_21, SG00_22, SG00_23) can meet the basic requirements of SG00_2. The third sub-goal in the first level is to let trainees know how to finalize the task and shut down the machine. SG00_01 and SG00_02 may be decomposed furthermore. The sub-goals in the lowest hierarchy can be supported by some training tasks directly.

The decision table (Table 5.5) shows us the mapping relationship between the sub-goals that represented by leaf nodes in Figure 5.6 and training tasks.

<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>DECISION RULES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG00_11: To familiarize trainees with initialization the CNC milling machine</td>
<td>Y Y Y Y Y Y</td>
</tr>
<tr>
<td>SG00_12: To master operation setting</td>
<td>N Y Y Y Y Y</td>
</tr>
<tr>
<td>SG00_21: To master machining parameters setting up</td>
<td>N N Y Y Y Y</td>
</tr>
<tr>
<td>SG00_22: To learn how to drill holes at the specified position in the manual mode</td>
<td>N N N Y N Y</td>
</tr>
<tr>
<td>SG00_23: To learn how to mill a slot at the specified position in the manual mode</td>
<td>N N N N Y Y</td>
</tr>
<tr>
<td>SG00_03: To know how to finalize the task and shut down the machine</td>
<td>Y Y Y Y Y Y</td>
</tr>
<tr>
<td>T1: Initialization</td>
<td>X X X X X X</td>
</tr>
<tr>
<td>T22: Operation mode selection</td>
<td>X X X X X</td>
</tr>
<tr>
<td>T222: Tool calibration with probe</td>
<td>X X X X X</td>
</tr>
<tr>
<td>T226: Manual measurement selecting</td>
<td>X X X X</td>
</tr>
<tr>
<td>T228: Feed rate setting</td>
<td>X X X X</td>
</tr>
<tr>
<td>T229: Manual reference point setting</td>
<td>X X X</td>
</tr>
<tr>
<td>T31: Spindle on</td>
<td>X X X</td>
</tr>
<tr>
<td>T32: Milling a slot</td>
<td>X X</td>
</tr>
<tr>
<td>T33: Drilling a hole</td>
<td>X X</td>
</tr>
<tr>
<td>T34: Drilling a hole</td>
<td>X X</td>
</tr>
<tr>
<td>T4: Homing and shutdown</td>
<td>X X X X X X</td>
</tr>
</tbody>
</table>
In the end, a training procedure was proposed by the system. As required, all the operational training processes are in the manual mode not in the program mode and tool calibration. All the measurements in the milling machine are scaled in inches. Then we cut a concave from (0, 0.75, 0.05) to (1, 0.75, 0.05) and two holes at the position (1.25, 0.75, 0.5); (1.75, 0.75, 0.5). In this situation we assume that we know the accurate length, width and height and we set (0, 0, 0) at the left top corner of the workpiece. This initial point corresponds to machine zero.

The workpiece used in this study for milling machine operations is shown in Figure 5.7 and the final product after machining is shown in Figure 5.8.

![Image](image.png)

**Figure 5.7  Workpiece**

![Image](image.png)

**Figure 5.8 Final product**
5.4.2.2 Knowledge in IPA

For enhancing the training effect, knowledge was embedded in the IPA. The responsibility of knowledge in IPA are to detect trainees' operation errors timely, provide necessary hints, and furnish on-line operation guidance.

Figure 5.9 shows an example of required knowledge for machine behavior model. The sub-REPN represents the procedure of manual operations setting up and acts as inference engine to active the related knowledge agent. Knowledge involved in is divided into two types as we stated in chapter two, task-driven knowledge (on-line guidance and tutorial) and impasse-driven knowledge (error detection). On-line guidance knowledge was related to each normal action transition, while error detection knowledge was added into the transition that represents a misoperation.

5.4.2.3 Knowledge for Task Model

An REPN-based task plan representation is shown in Figure 5.10 for a given mechanical part and its corresponding knowledge representation. Knowledge involved in a given task includes the required parameters, tools and so on. The structure of task-driven knowledge and impasse-driven knowledge in the task model is the same as in the machine behavior model. The plan recognition knowledge was for the ITPA. It was linked to the control place in the REPN representation.

5.4.2.4 Implementation

According to the nature of the knowledge involved in the domain of VR-based CNC training, it would be very naturally to use production rule to represent it. At the mean time, the knowledge engine was organized by using agent-oriented technology. Figure 5.11 is a snapshot of for the instruction provision scenario in the virtual CNC training system.
Figure 5.9 Knowledge in manual operation setting up procedure
Figure 5.10 Knowledge involved in Task Model
Knowledge in the system was linked to transitions in REPN representation. When a transition is fired, the knowledge related to it may be activated. For example, figure 5.12 shows the knowledge representation scheme and inference mechanism during manual operation selection procedure. If trainees do not perform the correct operation, T224' is fired and the corresponding virtual agent ErrorDetection that stored the impasse-driven knowledge is activated to propose the recover approach to trainees. If the training session is running under guidance, the virtual agent OnLineGuide will be in charge of instruction prompting (task-driven). Another example is shown in figure 5.13 to illustrate the knowledge representation of plan recognition and inference mechanism.

5.5 Summary

In this Chapter, a virtual CNC milling machine operations training prototyping system is presented. Under the help of knowledge, the IPA can instruct trainees to carry out training tasks by “virtually” operating step by step, and can detect the trainees' misoperation in no time. By incorporating knowledge into the ITPA and TBEA, the system provides the flexibility to structure training programs, monitors and measures the progress of a training session in run time. The system provides more information to the trainee in the immersive 3-D environment than a 2-D flat screen used in some graphic environments for training.
Figure 5.12 Implementation of knowledge in IPA
Figure 5.13 Implementation of knowledge for PR
CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

Although VR technology is of a great potential to be applied to training systems and enormous effort has been paid on the development of VR-ITS in different field, little research has been carried out on how to effectively integrate AI technology with VR-ITS in order to enhance the effect of VR-ITS and reduce the cost of training. The subject of this study is to develop a practical methodology to incorporate AI into VR-ITS. In this research,

1. A novel architecture for constructing a knowledge engine for intelligent VR-ITS has been proposed. The REPN-based agent-oriented knowledge engine can effectively cope with the complexity of knowledge in VR-based industrial training area and satisfy the requirements for knowledge in practical intelligent VR-ITS.

2. As one of the important components of the proposed KE, knowledge agents have been developed. To construct knowledge by agent technology makes knowledge be more easily expressed & understood, and the inheritance of agents can reduce the code of programming. Besides, distinct agents can run concurrently in the VR-ITS. In the mean time, it will also be easy to extend the knowledge agents to networked systems.

3. Based upon the agent design, the proposed hybrid knowledge representation scheme allows developers to represent particular knowledge in different forms so that all types of knowledge can be integrated into the system more effectively.

4. REPN is introduced into KE and it will take charge of the following two roles:
   - *As the inference engine* to respond to trainee' or system's requests in run-time
   - *As an efficient knowledge extraction tool*. During the development of VR-ITS, based on the proposed architecture, this knowledge extraction approach promises that the corresponding knowledge will be elicited more adequately and integrally.
5. The proposed methodology for design and development of KE for VR-ITS has been applied to knowledge construction in a VR-based CNC training prototyping system. The prototype indicates that this methodology is practically feasible to develop knowledge base systems for intelligent VR-ITS. The testing of effectiveness for the knowledge modeling in the VR-based CNC training system was performed by two groups of experiments. The results indicate the acceptable performance of the system, and the improvement performance of trainees through virtual training. Also, the positive training transfer was revealed by the experiments.

6.2 Future Work

This research is just a beginning work for the methodology study on knowledge modeling for intelligent VR-ITS. So far, we did not consider the topic of machine learning. If knowledge agents are able to continually improve their behavior by noticing recurrent patterns of actions and events, this will have a significant effect on the development of knowledge-based systems and many other related fields. The topic of machine learning is significant in size and scope, so significant that any detailed discussion is clearly beyond the scope of this research. However, because its potential benefits to knowledge-based systems, future works can be suggested to enhance the level of intelligence of the system.

In addition, the inference engine proposed in the KE is based on REPN. However, as an effective task modeling tool, Petri nets are widely used to analyze and model a large amount of real systems. To extend the knowledge modeling methodology to those systems modeled by generic Petri nets will also have a great potential use.
APPENDIX I

GENERIC KNOWLEDGE CLASS DEFINITIONS IN VR-ITS

// File knowledg.h

#ifndef knowledge_h
#define knowledge_h

#include "agent.h"

#include <stdio.h>

Class KnowledgeBase {
    public:
        KnowledgeBase( );
        ~KnowledgeBase( );
    private:
        const char *trans[MAX_RULES];
};

#endif

// File InstructionProvision.h

#ifndef InstructionProvision_h
#define InstructionProvision_h

#include "knowledge.h"

#include <stdio.h>

Class InstructionProvision : public KnowledgeBase{
    public:
        virtual void add_rule(const char *trans, rule rl);
        virtual void modify_rule(const char *trans, rule rl);
        virtual void delete_rule(const char *trans, rule rl);
};

#endif

// File troubleshooting.h

#ifndef troubleshooting_h
#define troubleshooting_h

#endif
#define troubleshooting _h

#include "instructionprovision.h"
#include <stdio.h>

struct RULE_TS {
    const char *trans;
    char *constraint[MAX];
    char *message;
};

typedef struct RULE_TS rule_TS;

Class TroubleShooting : public InstructionProvision {
    public:
        void add_rule(const char *trans, rule_TS rl);
        void modify_rule(const char *trans, rule_TS rl);
        void delete_rule(const char *trans, rule_TS rl);
        int execute_TS(const char *trans, char *constraint[], rule_TS rl);
    private:
        int num_ts_rules;
        rule_TS TS_rule[MAX_TS_RULES];
};

#endif

// File on_line_guide.h

#ifndef onlineguide_h
#define onlineguide_h

#include "instructionprovision.h"
#include <stdio.h>

struct RULE_GUIDE {
    char *constraint[MAX];
    char *message;
    const char *trans;
    rule_guide *next;
};

typedef struct RULE_GUIDE rule_OG;

Class OnLineGuide : public InstructionProvision {
    public:
        void add_rule(const char *trans, rule_OG rl);
    }
void modify_rule(const char *trans, rule_OG rl);
void delete_rule(const char *trans, rule_OG rl);
int execute_guide(char *current_state, char *constraint, rule_guide *rl);

private:
  int num_ip_rules;
  rule_guide GP_rule[MAX_OG_RULES];
};

#endif

// FunctionTutorial.h

#ifndef FUNCTIONTUTORIAL_H
#define FUNCTIONTUTORIAL_H

#include "instructionprovision.h"
#include <stdio.h>

struct RULE_FT {
  char *current_goal;
  char *message;
  const char *trans;
  RULE_FT *next;
};

typedef struct RULE_FT rule_FT;

Class FunctionTutorial : public InstructionProvision {
public:
  void add_rule(const char *trans, rule_FT rl);
  void modify_rule(const char *trans, rule_FT rl);
  void delete_rule(const char *trans, rule_FT rl);
  int execute_FT(rule_FT *rl);
private:
  int num_ft_rules;
  rule_FT FT_rule[MAX_FT_RULES];
};

#endif
## APPENDIX II

### KNOWLEDGE ACQUISITION TABLES

Table 1. Knowledge for Error Detection

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task Name</th>
<th>Additional Conditions</th>
<th>IPA's error detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>*****</td>
<td>*****</td>
</tr>
<tr>
<td>T11</td>
<td>Power on</td>
<td>Press (key) &lt;&gt; PO</td>
<td>Wrong! Press the POWER UP button first.</td>
</tr>
<tr>
<td>T12</td>
<td>Initialization</td>
<td>(Press (key) &lt;&gt; YES/NO)</td>
<td>Wrong! Press YES/NO please.</td>
</tr>
<tr>
<td>T22</td>
<td>Manual mode selection</td>
<td>(Press(key) &lt;&gt; M)</td>
<td>Wrong! Press MANUAL button as you set the goal.</td>
</tr>
<tr>
<td>T25</td>
<td>Program mode selection</td>
<td>(Press(key) &lt;&gt; P)</td>
<td>Wrong! Press PROGRAM.</td>
</tr>
<tr>
<td>T221</td>
<td>Tool calibration ?</td>
<td>Press (key) &lt;&gt; (YES v NO)</td>
<td>Wrong action! Yes/No.</td>
</tr>
<tr>
<td>T222</td>
<td>Tool calibration</td>
<td>(Press (key) = NO)</td>
<td>Wrong! You failed to achieve your goal, press YES.</td>
</tr>
<tr>
<td>T223</td>
<td></td>
<td>(Press (key) = YES)</td>
<td>Wrong! You failed to achieve your goal, press NO.</td>
</tr>
<tr>
<td>T224</td>
<td>Manual operation or diagnostics?</td>
<td>Press(key) &lt;&gt; (YES v No)</td>
<td>Wrong! Please press YES/NO to select manual operation/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>diagnostics.</td>
</tr>
<tr>
<td>T225</td>
<td>Manual sub mode selection</td>
<td>(Press (key) = NO)</td>
<td>Wrong! Please press YES to enter manual operation mode.</td>
</tr>
<tr>
<td>T225</td>
<td>Diagnostics mode selection</td>
<td>(Press (key) = YES)</td>
<td>Wrong! Please press NO to enter diagnostics mode.</td>
</tr>
<tr>
<td>T22511</td>
<td>Measurement Selecting</td>
<td>Press (key) &lt;&gt; (YES v NO)</td>
<td>Wrong action! YES/NO.</td>
</tr>
<tr>
<td>T22513</td>
<td>Metric system</td>
<td>(((Press (key) = NO) ∧ (pre_meas = MM)) ∨ ((Press (key) = YES) ∧ (pre_meas = INCH)))</td>
<td>Warning: You may fail to achieve your goal. The measurement you just set is metric. Do you want to change it now? (Y/N)</td>
</tr>
<tr>
<td>T22512</td>
<td>English system</td>
<td>(((Press (key) = YES) ∧ (pre_meas = MM)) ∨ ((Press (key) = NO) ∧ (pre_meas = INCH)))</td>
<td>Warning: You may fail to achieve your goal. The measurement you just set is INCHES. Do you want to change it now? (Y/N)</td>
</tr>
<tr>
<td>T22520</td>
<td>X axis feed rate specification</td>
<td>Press(key) (\Leftrightarrow) (NEXT (\lor) CLR)</td>
<td>Wrong! Press Next/CLEAR, please.</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------</td>
<td>-------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>T22522</td>
<td>X axis new feed rate inputting</td>
<td>(Press_alrea(key) = CLR) (\land) (Press(key) (\Leftrightarrow) NK)</td>
<td>Wrong! Please enter numbers.</td>
</tr>
<tr>
<td>T22524</td>
<td>Y axis feed rate specification</td>
<td>Press(key) (\Leftrightarrow) (NEXT (\lor) CLR)</td>
<td>Wrong! Press Next/CLEAR, please.</td>
</tr>
<tr>
<td>T22426</td>
<td>Y axis new feed rate inputting</td>
<td>(Press_alrea(key) = CLR) (\land) (Press(key) (\Leftrightarrow) NK)</td>
<td>Wrong! Please enter numbers.</td>
</tr>
<tr>
<td>T22428</td>
<td>Z axis feed rate specification</td>
<td>Press(key) (\Leftrightarrow) (NEXT (\lor) CLR)</td>
<td>Wrong! Press Next/CLEAR, please.</td>
</tr>
<tr>
<td>T2242a</td>
<td>Z axis new feed rate inputting</td>
<td>(Press_alrea(key) = CLR) (\land) (Press(key) (\Leftrightarrow) NK)</td>
<td>Wrong! Please enter numbers.</td>
</tr>
<tr>
<td>T2243</td>
<td>Set reference point</td>
<td>Press(key) (\Leftrightarrow) (X (\land) Y (\land) Z)</td>
<td>Wrong! Please press the axis first.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Press_alrea(key) = NEXT) (\land) (Press(key) (\Leftrightarrow) ZERO_CO)</td>
<td>Warning: You did not set the current location as reference point.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Move(X) &gt; max_x) (\lor) (Move(Y) &gt; max_y) (\lor) (Move(Z) &gt; max_z)</td>
<td>Wrong! Axis destination beyond maximum travel.</td>
</tr>
<tr>
<td>T2244</td>
<td>Manual Machining</td>
<td>(Press(key) = NEXT (\land) (Spindle = OFF)</td>
<td>A gross error! You should turn on the spindle before starting machining.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Move(X) &gt; max_x) (\lor) (Move(Y) &gt; max_y) (\lor) (Move(Z) &gt; max_z)</td>
<td>Wrong! Axis destination beyond maximum travel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Workpiece(type) = Plastics) (\land) (T_tip(FR, X) &lt; 10)</td>
<td>Warning: Feedrate is too low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Workpiece(type) = Plastics) (\land) (T_tip(FR, X) &gt; 16)</td>
<td>Warning: Feedrate is too high.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Workpiece(type) = Aluminum) (\land) (T_tip(FR, X) &lt; 5)</td>
<td>Warning: Feedrate is too low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Workpiece(type) = Aluminum) (\land) (T_tip(FR, X) &gt; 10)</td>
<td>Warning: Feedrate is too high.</td>
</tr>
<tr>
<td>T31</td>
<td>Power Off</td>
<td>(Spindle = ON) (\land) (Press(key) = PO)</td>
<td>Wrong! Please turn off spindle before you switch off power.</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Spindle = OFF) (\land) (Press(key) (\leftrightarrow) PO)</td>
<td>Wrong! Please press the red Power on the left top of the control box.</td>
</tr>
<tr>
<td>T32</td>
<td>Emergency Stop</td>
<td>Press(key) (\leftrightarrow) EM</td>
<td>A gross error! <strong>Strike Emergency button</strong> on the right top of the control box to stop the machine immediately.</td>
</tr>
<tr>
<td>Transition No.</td>
<td>Task Name</td>
<td>Additional Conditions</td>
<td>IPA's operation instruction</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>******</td>
<td>The main power switch on the left side of the control box.</td>
</tr>
<tr>
<td>T11</td>
<td>Power on</td>
<td>******</td>
<td>Press the YES button to initialize, or press the NO key to move axes.</td>
</tr>
<tr>
<td>T12</td>
<td>Initialization</td>
<td>******</td>
<td>Press the YES button to initialize, or press the NO key to move axes.</td>
</tr>
<tr>
<td>T25</td>
<td>Program mode selection</td>
<td>******</td>
<td>Press PROGRAM button for program mode.</td>
</tr>
<tr>
<td>T222</td>
<td>Tool calibration</td>
<td>******</td>
<td>Press YES button to enter tool calibration.</td>
</tr>
<tr>
<td>T223</td>
<td></td>
<td>******</td>
<td>Press NO button to escape tool calibration.</td>
</tr>
<tr>
<td>T225</td>
<td>Manual operation</td>
<td>******</td>
<td>Press YES button to enter manual operation.</td>
</tr>
<tr>
<td>T226</td>
<td>Diagnostics</td>
<td>******</td>
<td>Press No button for loading diagnostic routine.</td>
</tr>
<tr>
<td>T22511</td>
<td>Measurement Selection</td>
<td>******</td>
<td>Press NO button to escape the current measurement setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Press Yes button to confirm the measurement setting displayed.</td>
</tr>
<tr>
<td>T22513</td>
<td>Metric system</td>
<td>((Press (key) = NO) ∧ (pre_meas = MM) ) ∨ ((Press (key) = YES) ∧ (pre_meas = INCH))</td>
<td>The dimensional system you should use is Metric (MM).</td>
</tr>
<tr>
<td>T22512</td>
<td>English system</td>
<td>((Press (key) = YES) ∧ (pre_meas = MM) ) ∨ ((Press (key) = NO) ∧ (pre_meas = INCH))</td>
<td>The dimensional system you should use is English (INCHES).</td>
</tr>
<tr>
<td>T22520</td>
<td>X axis feed rate</td>
<td>******</td>
<td>If you want to use the displayed feed rate, press Next. Otherwise, press CLEAR.</td>
</tr>
<tr>
<td></td>
<td>specification</td>
<td></td>
<td>The X axis feed rate has been set down.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Press(key) = NEXT</td>
<td>Next Step: you should enter the new X axis Feed Rate using the numeric keys. The format is XX.X.</td>
</tr>
<tr>
<td><strong>T22522 X axis new feed rate inputting</strong></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, X) &lt; 10) ∨ (T_tip(FR, X) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22522 X axis new feed rate inputting</strong></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, X) &lt; 5) ∨ (T_tip(FR, X) &gt; 10))</td>
<td>For aluminum, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22522 X axis new feed rate inputting</strong></td>
<td>(Workpiece(type) = Steel) ∧ ((T_tip(FR, X) &lt; 0.1) ∨ (T_tip(FR, X) &gt; 5))</td>
<td>For steel, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Press_alrea(key)=CLR) ∧ (Press(key) = NK)</td>
<td>After you input the X axis’ new feed rate, Press NEXT to finish the X axis feed rate setting.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>Press(key) = NEXT</td>
<td>The X axis feed rate has been set down.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, X) &lt; 10) ∨ (T_tip(FR, X) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, X) &lt; 5) ∨ (T_tip(FR, X) &gt; 10))</td>
<td>For aluminum, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Steel) ∧ ((T_tip(FR, X) &lt; 0.1) ∨ (T_tip(FR, X) &gt; 5))</td>
<td>For steel, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>******</td>
<td>If you want to use the displayed feed rate, press Next. Otherwise, press CLEAR.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>Press(key) = NEXT</td>
<td>The Y axis feed rate has been set down.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>Press(key) = CLR</td>
<td>Next Step: you should enter the new Y axis Feed Rate using the numeric keys. The format is XX.X.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, Y) &lt; 10) ∨ (T_tip(FR, Y) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, Y) &lt; 5) ∨ (T_tip(FR, Y) &gt; 10))</td>
<td>For plastics, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>T22524 Y axis feed rate specification</strong></td>
<td>(Workpiece(type) = Steel) ∧ ((T_tip(FR, Y) &lt; 0.1) ∨ (T_tip(FR, Y) &gt; 5))</td>
<td>For plastics, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td>T22526</td>
<td>Y axis new feed rate inputting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Press_alrea(key) = CLR) ∧ (Press(key) = NK)</td>
<td>After you input the Y axis' new feed rate, Press NEXT to finish Y axis feed rate setting.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Press(key) = NEXT</td>
<td>The Y axis feed rate has been set down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, Y) &lt; 10) ∨ (T_tip(FR, Y) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, Y) &lt; 5) ∨ (T_tip(FR, Y) &gt; 10))</td>
<td>For plastics, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Steel) ∧ ((T_tip(FR, Y) &lt; 0.1) ∨ (T_tip(FR, Y) &gt; 5))</td>
<td>For plastics, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td>T22528</td>
<td>Z axis feed rate specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>******</td>
<td>If you want to use the displayed feed rate, press Next. Otherwise, press CLEAR.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Press(key) = NEXT</td>
<td>The Z axis feed rate has been set down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Press(key) = CLR</td>
<td>Next Step: you should enter the new Z axis feed rate using the numeric keys.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, Z) &lt; 10) ∨ (T_tip(FR, Z) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, Z) &lt; 5) ∨ (T_tip(FR, Z) &gt; 10))</td>
<td>For plastics, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(material) = Steel) ∧ ((T_tip(FR, Z) &lt; 0.1) ∨ (T_tip(FR, Z) &gt; 5))</td>
<td>For plastics, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td>T2252a</td>
<td>Z axis new feed rate inputting</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Press_alrea(key)=CLR) ∧ (Press(key) = NK)</td>
<td>After you input the Z axis' new feed rate, Press NEXT to finish the Z axis feed rate setting.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Press(key) = NEXT</td>
<td>The Z axis feed rate has been set down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Plastics) ∧ ((T_tip(FR, Z) &lt; 10) ∨ (T_tip(FR, Z) &gt; 16))</td>
<td>For plastics, the normal feed rate is from 10 to 16 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Workpiece(type) = Aluminum) ∧ ((T_tip(FR, Z) &lt; 5) ∨ (T_tip(FR, Z) &gt; 10))</td>
<td>For aluminum, the normal feed rate is from 5 to 10 INCHES per minute.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2253</td>
<td><strong>Set reference point</strong>&lt;br&gt;((\text{Workpiece(type) = Steel}) \land (\text{T_tip(FR, Z) &lt; 0.1}) \lor (\text{T_tip(FR, Z) &gt; 5}))  For steel, the normal feed rate is from 0.1 to 5 INCHES per minute.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|           | **Set Reference point**<br>
<p>|           | If the value displayed is not what you desired, press CLEAR key.                                                                         |
|           | <strong>Enter the desired position, then press the NEXT.</strong>&lt;br&gt;((\text{Press_area(key) = (X \lor Y \lor Z)}) \land (\text{Press(key) = CLR}))    |
| T2253?    | <strong>Move axis</strong>&lt;br&gt;<strong>Tip:</strong> Decrease the tool length to avoid exceeding the limitation.                                                   |
|           | <strong>Next step:</strong> Press the ZERO COODS to set your reference zero of the axis.                                                             |
|           | <strong>Next step:</strong> Press the NEXT.                                                                                                          |
| ....T2253?| <strong>Zero coordination</strong>&lt;br&gt;<strong>Press the ZERO COODS button.</strong>                                                                                   |
| T2253?    | <strong>Find current axis position</strong>&lt;br&gt;The value displayed is the distance from the home position or reference position, until you zero a particular coordinate. |
| T2252     | <strong>Feed rate setting</strong>&lt;br&gt;<strong>Press the “FEED RATE.”</strong>                                                                                                |
| T22520    | <strong>X-axis FR prompting</strong>&lt;br&gt;The current feed rate on X axis is showing up, if you need a new feed rate, press CLEAR key. Otherwise, press NEXT. |
| T22521    | <strong>Clear old setting</strong>&lt;br&gt;Press(key) &lt;&gt; CLR.                                                                                                 |
|           | <strong>Please press CLEAR button.</strong>                                                                                                           |
| T22522    | <strong>Enter new value</strong>&lt;br&gt;Press(key) &lt;&gt; NK.                                                                                                   |
|           | <strong>Enter the new feed rate using the numeric keys.</strong>                                                                                       |
| T22523    | <strong>Press(key) &lt;&gt; NEXT</strong>&lt;br&gt;<strong>Please press NEXT button.</strong>                                                                                   |</p>
<table>
<thead>
<tr>
<th>T22524</th>
<th>Y-axis FR prompting</th>
<th>******</th>
<th>The current feed rate on Y axis is showing up, if you need a new feed rate, press CLEAR key. Otherwise, press NEXT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T22525</td>
<td>Clear old setting</td>
<td>Press(key) &lt;&gt; CLR</td>
<td>Please press CLEAR button.</td>
</tr>
<tr>
<td>T22526</td>
<td>Enter new value</td>
<td>Press(key) &lt;&gt; NK</td>
<td>Enter the new feed rate using the numeric keys.</td>
</tr>
<tr>
<td>T22527</td>
<td></td>
<td>Press(key) &lt;&gt; NEXT</td>
<td>Please press NEXT button.</td>
</tr>
<tr>
<td>T22528</td>
<td>Z-axis FR prompting</td>
<td>******</td>
<td>The current feed rate on Z axis is showing up, if you need a new feed rate, press CLEAR key. Otherwise, press NEXT.</td>
</tr>
<tr>
<td>T22529</td>
<td>Clear old setting</td>
<td>Press(key) &lt;&gt; CLR</td>
<td>Please press CLEAR button.</td>
</tr>
<tr>
<td>T2252a</td>
<td>Enter new value</td>
<td>Press(key) &lt;&gt; NK</td>
<td>Enter the new feed rate using the numeric keys.</td>
</tr>
<tr>
<td>T2252b</td>
<td></td>
<td>Press(key) &lt;&gt; NEXT</td>
<td>Please press NEXT button.</td>
</tr>
<tr>
<td>T2253?</td>
<td>Exit the manual mode</td>
<td>******</td>
<td>Press any other “MODE” key.</td>
</tr>
<tr>
<td>T2253?</td>
<td></td>
<td>(Press(key) = NEXT) ∧ (Spindle = OFF)</td>
<td>The spindle can turned ON/OFF by the switch on the side of the spindle head.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Be attention: For manual control the Spindle ON/OFF control switch, which must be in the LOCAL position.</td>
</tr>
<tr>
<td>T31</td>
<td>Power Off</td>
<td>******</td>
<td>Please turn off the spindle by the spindle switch first, then press the power again to shutdown the machine.</td>
</tr>
<tr>
<td>T32</td>
<td>Emergency Stop</td>
<td>******</td>
<td>For release from emergency stop, you should turn right the emergency button, and then Power UP to start again.</td>
</tr>
</tbody>
</table>
### Table 3. Knowledge for CNC Machine Function Learning

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Task Name</th>
<th>Additional Conditions</th>
<th>IPA’s function learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>*****</td>
<td></td>
</tr>
<tr>
<td>T11</td>
<td>Power on</td>
<td>*****</td>
<td></td>
</tr>
<tr>
<td>T12</td>
<td>Initialization</td>
<td>*****</td>
<td>Press the YES button to initialize, the x and y tables and the z axis head will go to the home position, Z first then X &amp; Y. Press the NO key will allow you to move each axis.</td>
</tr>
</tbody>
</table>
| T22            | Manual mode selection      | *****                 | This mode is used to:  
a) Calibrate the height of tools which are mounted in tool holders.  
b) Calibrate the electronic probe.  
c) Move the machine's X, Y, and Z axes, for manual operation (i.e. nonprogrammed machining).  
d) Call up machine diagnostics.  
e) Demonstrate the machine using a stored program. |
| T25            | Program mode selection     | *****                 | Pressing PROGRAM key initiates the program run mode. You should position the display at the PROGRAM START either via the LINE MODE (enter the line number) or by backstepping through the program to the beginning of the program. |
| T222           | Tool calibration           | *****                 | You can calibrate the followings:  
1) The electronic probe.  
2) A set of quick change tools.  
**Notice:** If tool calibration is required, it should only be done **immediately following initialization.** |
<p>| T225           | Manual operation           | *****                 | In this mode, you can operate the machine manually by moving the X, Y, and Z axes, one at a time, from the keyboard. In some cases, it is much easier just go to the manual mode, do some operations, and exit, rather than going through the program mode, especially if you just wishes to drill a few holes. |</p>
<table>
<thead>
<tr>
<th>T226</th>
<th>Diagnostics</th>
<th>******</th>
<th>Diagnostic routine will be loaded, which permit a variety of checks to be performed on the machine and controller to verify if it is operating correctly.</th>
</tr>
</thead>
</table>
| T2251 | Measurement Selecting | ****** | The dimensional system used in the machine is either English (INCHES) or Metric (MM). In either case:  
- A YES response by you will put the virtual machine into the next sequence of prompts.  
- By a NO response, the display will then temporarily show the alternate selection of units and then will continue with the next sequence of prompts. |
|       |             |        | Before you can manually operate the machine, each axis’ feed rate must be specified so that the machine knows at what rate to make the move. |
| T2252 | Feed Rates Setting | ****** | In general, tougher material require slower speeds: (INCHES/minutes)  
- Plastics (10-16)  
- Aluminum (5-10)  
- Steel (0.1-5) |
|       |             |        | The feed rate you specified is too low. The damages may be caused as the followings:  
1. It may take too long time to machine the workpiece. The tool may too heat and easily to become blunt.  
2. The bits of workpiece may wrap and block the tool.  
3. The quality of machined surface is not good. |
|       |             |        | The feed rate you specified is too fast. The damages may be caused as the followings:  
1. It may break the tool.  
2. It may produce intolerable noise.  
3. The quality of machined surface is not good. |
<p>| T22520 | X axis feed rate specification | ****** | When the FR X = XXX prompt appears it will display the value of the previously used Feed Rate on the X axis. |</p>
<table>
<thead>
<tr>
<th>T22524</th>
<th>Y axis feed rate specification</th>
<th>******</th>
<th>When the FR Y = XXX prompt appears it will display the value of the previously used Feed Rate on the Y axis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T22526</td>
<td></td>
<td>******</td>
<td></td>
</tr>
<tr>
<td>T22528</td>
<td>Z axis feed rate specification</td>
<td>******</td>
<td>When the FR Z = XXX prompt appears it will display the value of the previously used Feed Rate on the Z axis.</td>
</tr>
<tr>
<td>T2252a</td>
<td></td>
<td>******</td>
<td></td>
</tr>
<tr>
<td>T2243</td>
<td>Set reference point</td>
<td>******</td>
<td>Reference point is a point in space, or more usually on the workpiece, to which all workpiece-cutting geometries and their dimensions are referenced in the coming operations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can enter the reference point for the Z, X, Y, and U coordinates. (The U coordinate is used only when an optional rotary table is in use).</td>
</tr>
</tbody>
</table>
|             | Press_alrea(key) = ( X ∨ Y ∨ Z ) |        | The current value displayed shows the current position of that axis.  
  • If you did not set reference point yet, the value is relative value to the home position.  
  • Otherwise, it is related to your reference point. |
| T2244       | Manual Machining              | ****** | In this mode, the user can perform the following manual operation from keyboard:  
  1. Move any axis from its present position to a new position.  
  2. Any axis can be moved or jogged in either direction  
  3. Zero the current value on the display.  
  4. Find the current position on any axis.  
  5. Chang the Feed Rate on any axis.  
  6. Move the Z axis to its HOME position.  
  7. You can exit this mode at any time. |
| T31         | Power Off                     | ****** | Before the end, spindle should be turned off.   |
| T32         | Emergency Stop                | ****** | When some accident happens, use emergency stop to avoid serious damage. |

Key: PO = POWER, M = MANUAL, CLR = CLEAR, NK = numeric keys, EM = Emergency Stop.  
******: NULL
APPENDIX III

USABILITY ANALYSIS

III.1 Introduction

As Berry and Hart (1990) emphasize, "The ultimate criterion of success for most expert systems is whether they will actually be used, and used to effect, by individuals other than the system developers." To many users, the knowledge-based system is the system's interface. If the interface is hard to use, then they will conclude that the system is hard to use. They may even conclude that the system is not useful, even though it may provide them with useful results if they could only master its interface [Adelman and Riedel 1997].

Several approaches, including empirical evaluation, analytic modeling, and techniques such as heuristic evaluation and cognitive complexity [Kieras and Polson 1985], have been used to evaluate the usability of human-computer. When operational prototypes has been developed, system usability can be evaluated using empirical testing and objective measures [Eberts, 1994]. To consider special issues of VR systems, empirical testing is the most useful and reliable approach for evaluating the usability. It refers to the group of methods for obtaining users' opinions about the usability of evolving prototypes and operational systems. There are many techniques for empirical testing. We adopted videotape analysis, Likert-like questionnaire and automatic data collection to evaluate our prototyping VR-CNC training system.

III.1.1 Performance Measurement Methods

Videotape analysis: by videotaping typical users carrying out tasks in the prototyped VR-CNC training system, we went back and reviewed the tapes to collect objective data that occurs so quickly during the experiment that it could not be recorded. This includes data on the errors that subjects made, choices made when there were multiple ways to complete a task, the success of users in completing a task and the time required to do so (including recovering from the impasse).

Likert-like questionnaires: It can serve as a good model for a questionnaire that evaluates subject's higher-level subjective response to the virtual training system.
Likert-like questionnaires are rating scales with which users retrospectively express their subjective satisfaction with specified aspects of human-computer interface. Typically, users circle numbers (such as 1-7 in our experiments).

**Automatic data collection:** The trainee behavior evaluation agent can capture information about the user's interaction automatically. It saves us from having to transcribe and determine all of the data from videotape.

### III.1.2 Factors to Measure

To evaluate the usability of a VR system, whether users can complete tasks is the first critical measurement that must be determined [Kenyon and Afenya, 1995]. At the meantime, we need to examine their error rates in doing so and how long time they took to accomplish these tasks or recover from impasses. We must also examine how users feel subjectively about it. The improvement measurement of trainees is represented with learning curves under the condition of allowing repetitive trials.

**Task completion:** A most rudimentary but important measure to assess is whether typical users of the system can accomplish the tasks they are called upon to an accomplish using the prototyping VR-CNC training system. In the +system, we do not provide any written material to users. Expecting users to deliberate written manuals is almost impossible in the case of VEs with visual displays (e.g., HMDs), since the display cut out the surrounding world.

In our experiments for VR-CNC training system, we taught subjects a few basic operations first. For example, how to use mouse/dataglove to grasp an object in VEs. We did not provide a one-on-one human tutor to teach every one of its subjects how to use the interface of VR-CNC training system, since it would be best for the evaluator in the usability studies not to coach the subjects. The subjects were only coached by the training system itself if they needed. We provided the subjects in our experiments with a series of tasks and without coaching and written materials. The experimenters recorded whether each of these tasks is attempted and completed.

**Errors:** The experimenters recorded errors when users tried to complete tasks with the prototype, or after it by videotape analysis and automatic data collection. Unlike task completion, which is usually quite easy to observe and record as an evaluation
proceeds, errors sometimes happen in burst and are difficult to reconstruct — and it will become worse if the evaluator has to both observe user's actions and record errors. It is one of the reasons why we used videotape technique. An error defined here is any user action that does not move the user towards the goal state during the task being performed. This means that some actions there are exploratory and do not cause any subsequent problems is permitted in the prototyping VR-CNC training system, and hence a novel solution for a task may be proposed by the user.

*Task completion time:* There might be differences to measure task completion time between the virtual training environment of evaluation laboratory or site and the real environment in which the real CNC machine is used. However, a decrease in task completion time should be noted as an indicator of learning. Furthermore, in some industrial areas, time is so crucial that we can not ignore it as we develop the related training systems [Kozak and Hancock, 1993]. In our experiments, each task completion time in VR-CNC training system was measured.

### III.2 Experiment Design

For completely evaluating the prototyping VR-CNC training system we developed, we designed two experiments (see Figure III.1 and III.2). The treatments in different experiments are also shown in the figure.

![Diagram](image.png)

*Figure III.1 Experiments for evaluating the performance of the IPA*
III.2.1 Hypothesis

*Task performance in the experiments*

VR-CNC training system was developed by following the criteria of the framework about how to develop an efficient VR-based industrial training system, and supported by knowledge base system that was developed by following the proposed knowledge modeling approach. Therefore, significant performance improvement can be predicted after VR-CNC training. Because we modeled the real world as veridical as possible in the VR system, the IPA with knowledge would effectively help trainees to recover from the impasses via providing timely instruction, and the significant difference in recovering time for different type of errors was expected. So we get the following two hypothesis:

Hypothesis 1: The performance of subjects who are trained in VR-CNC training system will be better than of those who operate the real CNC machine without VR training.

Hypothesis 2: The recover time for the subjects who take the VR-CNC training first will be shorter than it for those who do not take the VR training.

In addition, we expected to find similar learning curves for real-world and VR training. These hypotheses are evaluated by examining both training time data and task performance data.
Hypothesis 3: There are similar learning curves between the real world and virtual training in regards to completion time and errors.

Subject attitudes under the conditions with and without IPA

Subject attitudes mainly refer to user satisfaction. Following the data session, each participant filled a questionnaire to assess user acceptance. Subject attitudes to the system performance were evaluated by this experiment.

We expected that IPA in VR-CNC would provide timely instruction to help the subjects to recover from the deadlocks, and the system permit users to move freely and give users more instruction if they need. As a result, the performance of subjects in VR-CNC with the IPA would improve more significantly than VR-CNC without the IPA. At the same time, subjects in VR-CNC training system with the IPA would have higher satisfaction level than subjects in VR-CNC without the IPA.

Hypothesis 4: The subject performance in VR-CNC with IPA will improve more significantly than without IPA.

Hypothesis 5: Subjects in VR-CNC with IPA will feel more satisfied than without IPA.

III.2.2 Methodology

The experimental task

The experimental task required subjects to learn how to use CNC milling machine to mill a slot and drill two holes on specified positions. The experiment was conducted in the VR Application Laboratory of IEEM of HKUST. The lab is closed so that experimenters were able to control the environment. That means that potential confounding variables were eliminated or held constant. We also used one subject at a time to provide for high recognition accuracy and to control over unforeseen environmental variables. The between-subjects design with 1 factor and multiple-test method was selected.

Training

Each subject was given 2 hours introduction to the CNC machine and a brief introduction to the VE and the task by researchers. The training for some basic operation in VE was given to the subjects, for instance, how to push buttons, how to
move an object and so on. When the subject expressed confidence in using the system, the training session ended. All training was given by the same experimenters to warrant constancy of procedure.

**Interfaces**

Subjects used one of the two VR-based training versions: VR-CNC with IPA and VR-CNC without IPA that is designed and implemented by following the framework we proposed for developing VR-based industrial training. Each subject's training environment and task were identical. Upon completion of the task, subjects were asked to complete a post-test questionnaire (see Appendix IV). All subjects told us that they liked the idea of using VR to provide training, but some did find it to be somewhat frustrating to use.

**Subjects**

Our experiment utilized 18 students in Hong Kong universities. To get poised data, they were randomly and equally divided into three treatment groups. Six of them were asked to do a given experimental task by VR CNC with the IPA, while the other six used VR-CNC without the IPA to do the same task. Trainees in these two groups were required to finish the task by exceeding six trials. The third group was asked to do the same task on the real machine in the Manufacturing System Design Lab after reading the operation manual <<DM2400 Program Manual>> for two hours. The group experienced the VR-CNC with IPA would do the same task on the real machine and in the same lab too, after they accomplished the six trials in VR system.

**The dependent variables**

The dependent variables in this project included three performance variables (task completion, completion time and errors), and user attitude (satisfaction).

**Performance recorded:**

1. Task completion: times that the subject can complete the whole training task;
2. Completion time: the total time that the subject required to complete the whole training task, time that subjects completed the sub-tasks that constituted the whole training task, and the recovering time that the subject required to recover from the impasses;
3. Errors: error numbers and types that the subject made during the task performing.

*User attitudes (satisfaction):*

Succeeding completion of the task, each participant responded to a questionnaire using multi-item scales to assess user acceptance and satisfaction. Data for satisfaction were coded on an interval assumed Likert-scale of 1 to 7 with 7 representing a low likelihood of the measure, and 1 indicating a high likelihood, as perceived by the subject (see Table III.1).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SA</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>Completely Agree</td>
<td>Agree</td>
<td>Neither agree nor disagree</td>
<td>Disagree</td>
<td>Completely Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
</tbody>
</table>

Table III.1: Likert-like for satisfaction questionnaire

**III.3 Results and Analysis**

**III.3.1 Questionnaire Analysis**

The survey results (see Table III.2) show that most of subjects favorably accept the user interface design and consider the system easy to use. Most of subjects felt that the VR-CNC feedback was quick enough. The feeling of immersion (by 3-D sound effect, and CrystalEye) and degree of freedom in interaction were acceptable.

**III.3.2 Dependent Measures Analysis**

The performance was evaluated on three metric dependent variables (task completion times, completion time, and errors) and satisfaction was based on the degree of satisfaction towards methods as described as above. The one way ANOVAs were performed for all these measurements. The significant differences between the three environments were found for all the measures.

*Training Transfer*

For the completion time and the number of error comparisons in real CNC operations between the two groups, the VT group is significantly different from the only manual-reading group with the former having fewer errors. (see Figure III.3 and III.4)
Table III.2: Response of user interface evaluation

<table>
<thead>
<tr>
<th>Survey Question</th>
<th>No. of Acceptance</th>
<th>No. of M</th>
<th>No. of Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SA</td>
<td>CA</td>
<td>A</td>
</tr>
<tr>
<td>1. The VR-CNC’s approach to training is acceptable.</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2. The VR-CNC’s objects look like real.</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3. The VR-CNC is easy to use after brief introduction by the experimenter.</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4. The virtual training environment has high fidelity in visual and audio effects.</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5. The VR-CNC response time is acceptable.</td>
<td>10</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6. The VR-CNC permits the user to control the order in which different operations are done.</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>7. Completing the task with the VR-CNC is fast enough for my needs.</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>8. I had the sense of immersion when I was in the VR-CNC environment.</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>9. I have a lot of confidence in the VR-CNC’s approach to training.</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10. The VR-CNC lets the user feel freely while doing the training task given.</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure III.3: The mean of the completion time committed by six novices performing a given task as a function of the number of VR-CNC training sub-tasks completed
Figure III.4: The mean of the number of errors committed by six novices performing a given task as a function of the number of VR-CNC training sessions completed.

ANOVA analysis

Table III.3 shows us the one-way ANOVA analysis for the completion time.

The result of one-way ANOVA analysis for the numbers of error is showed in Table III.4.

Table III.3: Results of one-way ANOVA analysis of Variance for the Completion Time of each task/action

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Task</th>
<th>Mean VT-Group</th>
<th>Mean MR-Group</th>
<th>F(1, 11)</th>
<th>MSe</th>
<th>p</th>
<th>Difference (p&lt;0.10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>1.24</td>
<td>1.2917</td>
<td>0.21</td>
<td>0.008</td>
<td>0.657</td>
<td>NSID</td>
</tr>
<tr>
<td>T21</td>
<td>Mode Selection</td>
<td>0.1317</td>
<td>0.25</td>
<td>4.06</td>
<td>0.042</td>
<td>0.072</td>
<td>SSID</td>
</tr>
<tr>
<td>T226</td>
<td>Measurement Selecting</td>
<td>0.058</td>
<td>0.07</td>
<td>1.96</td>
<td>0.0004</td>
<td>0.192</td>
<td>NSID</td>
</tr>
<tr>
<td>T228</td>
<td>FR setting</td>
<td>0.9</td>
<td>1.5</td>
<td>10.59</td>
<td>1.098</td>
<td>0.009</td>
<td>SID</td>
</tr>
<tr>
<td>T229</td>
<td>Ref. Point setting up</td>
<td>1.01</td>
<td>1.44</td>
<td>1.68</td>
<td>0.538</td>
<td>0.224</td>
<td>NSID</td>
</tr>
<tr>
<td>T31</td>
<td>Spindle on</td>
<td>0.19</td>
<td>0.36</td>
<td>5.18</td>
<td>0.09</td>
<td>0.046</td>
<td>SID</td>
</tr>
<tr>
<td>T32</td>
<td>Mill a slot</td>
<td>3.01</td>
<td>5.40</td>
<td>8.19</td>
<td>17.21</td>
<td>0.017</td>
<td>SID</td>
</tr>
<tr>
<td>T33</td>
<td>Drill a hole</td>
<td>1.89</td>
<td>2.05</td>
<td>9.00</td>
<td>2.245</td>
<td>0.013</td>
<td>SID</td>
</tr>
<tr>
<td>T34</td>
<td>Drill a hole</td>
<td>0.77</td>
<td>1.01</td>
<td>6.87</td>
<td>0.1633</td>
<td>0.026</td>
<td>SID</td>
</tr>
<tr>
<td>T4</td>
<td>Shutdown</td>
<td>0.3967</td>
<td>0.41</td>
<td>0.12</td>
<td>0.00083</td>
<td>0.736</td>
<td>NSID</td>
</tr>
<tr>
<td>Total</td>
<td>Manual Operation</td>
<td>8.90</td>
<td>13.238</td>
<td>9.86</td>
<td>56.46</td>
<td>0.011</td>
<td>SID</td>
</tr>
</tbody>
</table>

Notes:

VT – Virtual Training  SID -- Significant difference
MR – Manual Reading   SSID -- Slightly significant different
MSe -- Mean Square in error NSID -- No significant difference
Table III.4: Results of one-way ANOVA analysis of Variance for the Number of Error of each task/action

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Task</th>
<th>Mean VT Group</th>
<th>Mean MR Group</th>
<th>F(1, 11)</th>
<th>MSe</th>
<th>p</th>
<th>Difference (p&lt;0.10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Initialization</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Identical</td>
</tr>
<tr>
<td>T21</td>
<td>Mode Selection</td>
<td>0.333</td>
<td>0.667</td>
<td>0.71</td>
<td>0.333</td>
<td>0.418</td>
<td>NSID</td>
</tr>
<tr>
<td>T226</td>
<td>Measurement Selecting</td>
<td>0</td>
<td>0.333</td>
<td>2.50</td>
<td>0.333</td>
<td>0.145</td>
<td>NSID</td>
</tr>
<tr>
<td>T228</td>
<td>Feed Rate Setting</td>
<td>0</td>
<td>0.5</td>
<td>5.00</td>
<td>0.75</td>
<td>0.049</td>
<td>SID</td>
</tr>
<tr>
<td>T229</td>
<td>Ref. Point Setting Up</td>
<td>0.1667</td>
<td>0.1667</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>NSID</td>
</tr>
<tr>
<td>T31</td>
<td>Spindle on</td>
<td>0</td>
<td>0.1667</td>
<td>1</td>
<td>0.0833</td>
<td>0.341</td>
<td>NSID</td>
</tr>
<tr>
<td>T32</td>
<td>Mill a slot</td>
<td>1.167</td>
<td>5</td>
<td>34.35</td>
<td>44.08</td>
<td>0</td>
<td>SID</td>
</tr>
<tr>
<td>T33</td>
<td>Drill 1st hole</td>
<td>0.0667</td>
<td>2</td>
<td>10</td>
<td>5.333</td>
<td>0.01</td>
<td>SID</td>
</tr>
<tr>
<td>T34</td>
<td>Drill 2nd hole</td>
<td>0.1667</td>
<td>0.8333</td>
<td>3.35</td>
<td>1.333</td>
<td>0.156</td>
<td>NSID</td>
</tr>
<tr>
<td>T4</td>
<td>Shutdown</td>
<td>0</td>
<td>0.5</td>
<td>5</td>
<td>0.75</td>
<td>0.049</td>
<td>SID</td>
</tr>
<tr>
<td>Total</td>
<td>Manual Operation</td>
<td>3.167</td>
<td>9.833</td>
<td>32</td>
<td>133.33</td>
<td>0</td>
<td>SID</td>
</tr>
</tbody>
</table>

* --- no value due to original data are identical

Analysis of learning curves

Note that the means of the times of failure, completion time and number of errors committed by novices performing a given task are the functions of the number of the training trials completed, the learning curves for these were illustrated as Figure III.5, Figure III.6, and Figure III.7 respectively.

For the task failure (completion) times between the two interfaces, the subjects using VR-CNC with the IPA always completed the task while the subject using VR-CNC without IPA not. (Figure III.5)

![Comparison of Failure Times](image_url)

Figure III.5: The average failure time required to perform a given task as a function of the number of VR-CNC training sessions completed by subjects
For the completion time comparison (Figure III.6) between the two interfaces, the results are the same as the above. It is interesting to note that all novices in the same group rapidly approached the same level of proficiency under the conditions with or without the IPA.

![Comparison of Completion Time](image)

Figure III.6: The average time required to perform a given task as a function of the number of VR-CNC training sessions completed by six novice students

Figure III.7 illustrates the comparison of mean error number between the two interfaces carried out by subjects during their task performing in the six trails.

![Average Error Number](image)

Figure III.7: The mean of the number of errors perpetrated by subjects performing the given task as a function of the number of trials

**III.4 Evaluation of the IPA's Performance in Recovery from Impasses**

In an intelligent VR-ITS, with the help of knowledge, the IPA should be able to recognize and interpret trainee impasses, assist them to find the way to get back to normal, and that the resulting tutoring would help trainees require skills in CNC milling manual operations. The study was conducted with twelve students. All of them have no VR experience and rarely CNC operation experience. They were
divided into two groups. The difference between two groups was that the trainees in Group I were tutored by IPA during the exercise, while the trainees in Group II did not receive any instruction during training.

Hypothesis 6: The recover time of trainees from impasses in group I with IPA will be significantly shorter than in group II without IPA, although all of the trainees will exhibit performance improvement through the experiment.

To evaluate the effectiveness of knowledge in IPA, the questions in Table III.5 were used as the evaluation criteria. The goal for this part of the evaluation was to determine how well IPA was able to understand the trainee's behavior in making a decision of whether or not to provide tutoring. Since here IPA is an impasse-driven tutor, this means that it has to be able to detect impasses as they are previously defined and avoids making a false detection. Hence, Q1 asks whether IPA recognized all of the actions taken by the trainees that one would expect them to take under nominal conditions. Q2 through Q5 cover each of the three impasse types (i.e. action constraint violations, plan dependency violations, and goal failures). Q2 and Q3 address the variations of recognizing whether a plan dependency violation has occurred or not. Q2 covers the impasse case and Q3 covers the case where a deviant action is warranted by the situation. Q4 asks whether the tutor was able to recognize and explain when the trainee failed to achieve the goals of a plan, and Q5 asks whether individual action constraint violations were recognized.

Table III.5: Evaluation criteria and result for the effectiveness of the IPA

<table>
<thead>
<tr>
<th>No.</th>
<th>Question Statement</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Did the IPA provide the user with effective directions so that one always knows what to do next?</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Q2</td>
<td>Did the IPA correctly interpret trainee deviation from the default procedure when there were no situational factors requiring the deviation?</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Q3</td>
<td>Did the IPA correctly interpret trainee deviation from the default procedures when they were in reaction to situational factors?</td>
<td>83.3%</td>
<td>16.7%</td>
</tr>
<tr>
<td>Q4</td>
<td>Did the IPA recognize when the trainee failed to achieve a goal? Was it able to explain the goal failure?</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Q5</td>
<td>Did the IPA recognize all action constraint violation? Was it able to explain the action constraint violation?</td>
<td>83.3%</td>
<td>16.7%</td>
</tr>
</tbody>
</table>
Result Analysis

The results, shown in Table III.5, of the experiment suggest that the IPA supported by knowledge holds promise as a method for recognizing trainee impasses and for explaining how to resolve them in a satisfactory manner. In addition, after analyzing the recorded videotape, we found that IPA did not make any misinterpretations, and all of its analyses fast enough for a promptly interaction with the trainee.

The study also suggests that the way providing training with IPA helped trainees to improve their skills in CNC milling operations more quickly than trainee without tutoring (see the above section). The trainees from both groups reached impasses while performing the task, but there was a significant difference between the two groups in the amount of time it took to resolve these impasses. While both groups acquired the same amount of skill in cases where there was an action constraint violation, the trainee in Group I (tutored by IPA) resolved impasse and acquired the new knowledge significantly faster than the trainees in Group II. Table III.6 summarizes data from the experiment that shows how long it took the trainees from the two groups to execute selected steps (T1, T21, T226, T228, T229, T31, T32, T33, T34, T4).

![Comparison of Recover Time](image)

Figure C8: Mean time to recover on the task "manual operation training" in VR-CNC as a function of the number of task trials completed by subjects with and without tutor

In the training phase, the subjects in Group I received assistance from the IPA whenever an impasse was encountered, while the Group II subjects were left to struggle with impasse by themselves. In the task of "basic manual operation training" there is a marked difference in the results between the tutored and untutored subjects. These results appear to confirm the obvious: a trainee who receives tutoring at an impasse point spends less time looking for a solution than one who does not have a
tutor. It is interesting to note that, although the novices used in this investigation had very different levels of prior experience related to the task, all novices rapidly approached the same level of proficiency for recovering from the impasse. All of the subjects exhibited improved performance. This is consistent with our hypothesis that people learn by resolving impasses (deadlocks), which is a tenet held by a number of learning researchers (e.g. VanLehn, 1988; Laird et al., 1987; Newell, 1990).

III.5 Summary

As the comparison for the completion time between two groups (VT group and MR group), the VR training group obviously learned something during the training period. For procedural skill tasks, such as T21, T228, T31, T32, T33, T34, due to similarity between the training context and task context, transfer of training of was improved significantly. For motor skill tasks, such as T229 (Reference point setting up), the learning did not significantly transfer to the real-world task. Much further real training may be needed. For tasks T1, T226, and T4, since they are too simple for operations, no significant difference was found. For the error numbers, significant differences were found for the task/action T228, T32, T33, and T4.

Despite geometric similarity and task similarity, the virtual training did not provide adequate training for a motor skill task in the real world. One way to improve transfer of training is to improve the similarity between the training context and task context. Although system limitations make this prospect unlikely for the near future, we can always try to make the virtual training environment more like the real world. A more promising approach to improve transfer may be to make the real training environment, in essence, more like the virtual training environment. We can begin by determining the key aspects of the real world task context, which are necessary for adequate performance. By carefully modeling only the necessary parts of the real world, we can begin to bring the virtual and real context closer together. This is particular importance when considering real world designs that might significantly benefit from the convenience of using VR training.
APPENDIX IV

QUESTIONNAIRES FOR VR-CNC TRAINING EXPERIMENTS

Name:  
Age:  

1. Questions for system usability evaluation

1. VR-CNC’s approach to training is acceptable.  
2. VR-CNC’s objects look like real.  
3. VR-CNC is easy to use after brief introduction by the experimenter.  
4. The virtual training environment has high fidelity in visual and audio effects.  
5. VR-CNC response time is acceptable.  
6. VR-CNC permits the user to control the order in which different operations are done.  
7. Completing the task with the VR-CNC is fast enough for my needs.  
8. I had the sense of immersion when I was in the VR-CNC environment.  
9. I have a lot of confidence in the VR-CNC’s approach to training.  
10. The VR-CNC lets the user feel freely while doing the training task given.

2. Questions for performance evaluation of the IPA

<table>
<thead>
<tr>
<th>No.</th>
<th>Question Statement</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Did the IPA provide the user with effective directions so that one always knows what to do next?</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>Did the IPA correctly interpret trainee deviation from the default procedure when there were no situational factors requiring the deviation?</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>Did the IPA correctly interpret trainee deviation from the default procedures when they were in reaction to situational factors?</td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>Did the IPA recognize when the trainee failed to achieve a goal? Was it able to explain the goal failure?</td>
<td></td>
</tr>
<tr>
<td>Q5</td>
<td>Did the IPA recognize all action constraint violation? Was it able to explain the action constraint violation?</td>
<td></td>
</tr>
</tbody>
</table>
REFERENCES


Cardoso, J., R. Valette and D. Dubois, (1990), Petri nets with uncertain markings, in Advances in Petri Nets 1990, Springer-Verlag, pp. 64-78.


Hon, C. L., 1996, Master thesis, *A virtual reality-based training system (VRTS) for CNC milling machine operations*, HKUST.


Using virtual environments to train firefighters.


