Using UML to develop Knowledge-based System for Adaptive Scheduling

WEIDA WANG[1], HE XU[2], WENJIAN LIU[1]
1. School of Machatronics Engineering
Harbin Institute of Technology
92 Xidazhi Street, Nangang District, Harbin, Heilongjiang, 150001
PEOPLE'S REPUBLIC OF CHINA

2. School of Mechanic and Electric Engineering
Harbin Engineering University
145 Nantong Street, Nangang District, Harbin, Heilongjiang, 150001,
PEOPLE'S REPUBLIC OF CHINA

Abstract: - It is one of the most important issues that constructing better Knowledge-Based System (KBS) to fulfill an efficient adaptive scheduling to complex manufacturing system. A method for developing and documenting the KBS for adaptive scheduling, using the Unified Modeling Language (UML) is presented . It employed UML to describe the system analysis, system design and system implementation of KBS, so various views were built up to explore the static structure and dynamic behavior of KBS. This method alleviates the highly dependent on domain knowledge experts. The quality and efficiency of the KBS are improved. Finally a case is given to validate the feasibility and reliability of the proposed approach.

Key-Words: - Adaptive scheduling; KBS; UML

1 Introduction

The adaptive scheduling is a good production scheduling approach. It can identify the current state of manufacturing system and then deicide on the appropriate dispatching rules to be used [1]. It is a distinctive feature that great ability to withstand constant disruption and maintain the system running effectively in the adaptive scheduling.

The knowledge-based system (KBS) for adaptive scheduling, which is an important bridge between the domain expert or the automatic knowledge acquisition technology and the scheduler, is the foundation of an adaptive scheduling system. For a concrete problem proposed by the scheduler, KBS will refer to its knowledge base that contains the knowledge generated by the expert or the acquisition technology, obtain the suitable solution. The knowledge can map the production state into the appropriate dispatching rule.

The problem of developing a KBS well is one of the most frequent problems that knowledge engineers face [2, 3]. When KBS are developed by rapid prototyping that is most popular approach currently, good design relies on the knowledge engineer's programming skills, and on his ability to devise, remember, and dynamically update a design specification. It is possible for the system to get out of control so that even its author can not understand why apparently small changes have large effects on the overall system.

The Unified Modeling Language (UML) is the *defacto* standard for object modelling in the software engineering area. It can provide a powerful framework and notation for modeling business and objects.

Hakansson [4] suggested using UML to represent the expert's domain knowledge. Chung and Pak [5] utilized UML diagrams to develop a knowledge-based business system for a small financial institute.

As UML is rarely adopted in developing KBS of the production and manufacturing area, the study on applying UML approach under this context in this paper would be of added value. This article presents a methodology that use UML to create the model documentation of the KBS for adaptive scheduling. It contains applying UML diagrams to describe the system analysis, design and implementation of KBS.

The rest of this paper is organized as follows. A brief review about UML is given in section 2. The proposed method is introduced in detail in section 3 . A case applying the proposed method is presented in section 4. Conclusions are given in section 5.

2 A Brief Review of UML

UML is a general-purpose visual language for developing models depicting various view of a system that can be used as a blueprint to develop and implement the software applications. It is used to visualize, specify, construct, and document the artifacts of a system [6]. UML can present the static structure and the dynamic behavior of a system. The static structure defines the kinds of objects important to a system and to its implementation, as well as the relationships among the objects. The static views are composed of use case diagram, class diagram, component diagram and deployment diagram. The dynamic behavior defines the history of objects over time and the communications among objects to accomplish goals. The dynamic views are made up of state chart diagram, activity diagram, sequence diagram and collaboration diagram.

3 Proposed Method

3.1 System Analysis

There is no need for one use case diagram to capture everything about a system. A well-structured use case diagram is focus in communicating one aspect of a system. It contains only those use cases and actors (classes representing external agents) that are essential for understanding that aspect. It must provide that details consistent with its level of abstraction.

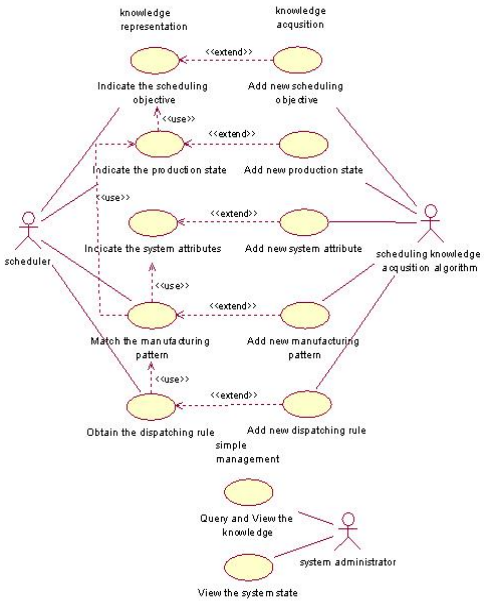


Fig. 1 The use case diagram of KBS

As shown in Fig.1, use cases under this context could be divided into three parts namely knowledge representation, knowledge acquisition and simple management. The role and responsibility of the scheduler, the scheduling knowledge acquisition algorithm and the system administrator on the KBS can be illustrated clearly. The scheduler is responsible for daily dispatching tasks in the shop floor. It should be the one that research for solution in the KBS. It is the knowledge user. The acquisition algorithm, using the machine learning technology, generates the knowledge that is represented in the form of IF-THEN. It is the knowledge provider. The system administrator is in charge of the KBS.

The scheduler is association with only use cases in the knowledge representation. This part contains five use cases, including “Indicate the scheduling objective”, “Indicate the production state”, “Indicate the system attributes”, “Match the manufacturing pattern” and “Obtain the dispatching rule”.

The case “Obtain the dispatching rule” uses another case “Match the manufacturing pattern”, which also needs to use the case “Indicate the system attributes” to work out the rule.

The acquisition algorithm is association with only use cases in the knowledge acquisition. There are five cases in the acquisition. They are “Add new scheduling objective”, “Add new production state”, “Add new system attribute”, “Add new manufacturing pattern” and “Add new dispatching rules”.

There are related by the “extend” relationship rather than the “include” between cases in the representation and those in the acquisition. Because the latter are optionally useful for the former only. The latter are useful when KBS has a request for a new scheduling objective, new production state, new system attribute, new manufacturing pattern or new dispatching rule.

There are two cases around the administrator. They are “Query and View the knowledge” and “View the system state”.

Activity diagrams are used to model the dynamic aspect of the system. It is essentially a flow chart showing flow of control from activity to activity. These diagrams commonly contain activity states, action states, transitions along with notes and constraints. Fig.2 describes the workflow of KBS.

The scheduler should obtain the suitable rule in reasonable time, so KBS needs to define the boundary of the production state by capturing scheduling objectives. The state domain contains related production states and corresponding system attributes and their value ranges. KBS can obtain the acceptable rule based on the scheduling knowledge.

In order for the scheduling knowledge acquisition algorithm to effectively transfer its knowledge, KBS has to capture the problematic production state without the suitable solution, make training examples and request the acquisition algorithm to correlate the objective, the problematic production state and the rule.

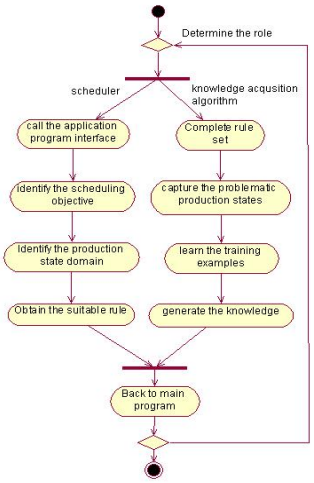


Fig. 2 The activity diagram of the scheduler and the scheduling knowledge acquisition algorithm

3.2 System Design

Class diagrams are the most common found in modeling object-oriented systems and show a set of classes, interfaces and collaborations and their relationships. Classes are populated with attributes and methods. Attributes represent the model inputs and methods present the functions needed for calculations or data manipulation. Relationship between classes are represented by lines and described by labels, arrowheads and notation. As shown in Fig.3, the KBS contains four main entity classes, including class SchedulingObjectives, class ProductionStates, class DispatchingRules and class SystemAttributes.

Attributes of each class rather than operations are given in the diagram because the former are more important for the decision-making. The first decision is to determine the scheduling objective that should be satisfied. Then the next decision is to identify the current problematic state, which is described by a conjunction of system attributes. Finally a suitable choice would be found from the rule list according to the objective and state mentioned above.

The relationship between class SchedulingObjectives and class ProductionStates is a composition and not an aggregation. The aggregation represents a “while/part” relationship. A composition is a stronger form of aggregation. With a composition, the parts live and lie with the whole and can not be transferred. Obviously if an objective does not exist, its corresponding production state will not exist. That is, a production state does not exist independently without an objective.

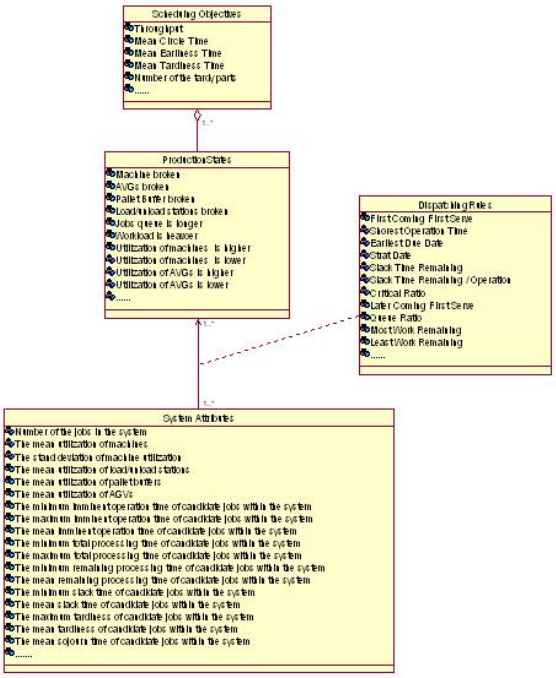


Fig. 3 The class diagram of KBS.

The relationship between class ProductionStates and class SystemAttributes is an association with multiplicity. In the KBS, a production state is represented by a conjunction of a set of system attribute. In other words, one state may have many attributes, and one attribute may relate to many states.

Class DispatchingRules is used to describe the properties of relationship between the two classes aforementioned. It is an association class.

Sequence diagrams show the interaction of a set of objects and their relationships by the messages dispatched among them. It emphasizes the time ordering of messages. A good way to model the dynamic aspects of the system is by building up the storyboards of scenarios including the interactions of certain important and interesting objects and messages that may be dispatched among them.

The sequence diagram shows objects arranged along the X-axis from left to right and messages ordered in creasing time, along Y-axis from top to bottom. They commonly contain objects, links and messages along with notes and constraints. It also has an object lifeline, a vertical dashed line, which represents the existence of an object over a period of time.

Fig.4 contains of objects that are reacted from the previous class diagram. They are represented on the horizontal axis at the top of the diagram. All the objects are linked by the association instances as per their sequence of participation in the process. The objects send call message and invoke a method to start an action.

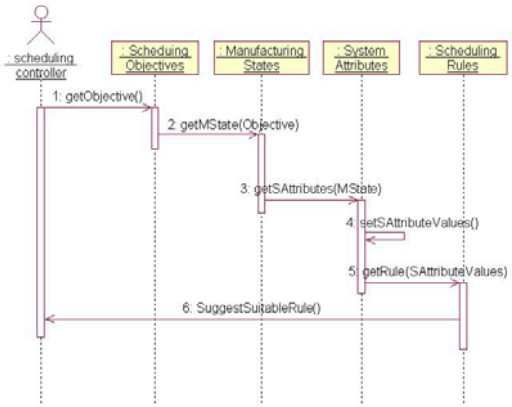


Fig. 4 The sequence diagram of KBS

The state chart diagram is used to model the dynamic aspects of the system and mostly to model the behavior of reactive objects. A reactive object continuously changes its state during its lifetime as a response to the events dispatched from outside its context. These diagrams maybe attached to classes and use cases. These diagrams will be used whenever following the change of state of an object in more important than following the flow of activities.

As an example, the KBS for adaptive scheduling is chosen to trace its states through its lifetime after initiation. Fig.5 illustrates the state changes during figuring out the suitable rule in the KBS. It starts with the initial state, and various events in the information process continuously change the state of KBS. These different states have been clearly shown in the diagram.

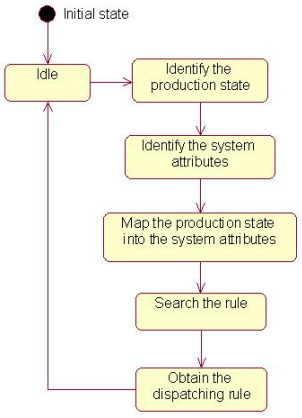


Fig. 5 The state chart diagram of KBS

3.3 System Implementation

A component diagram can show dependences of various sorts among such components. The component which arrow is leaves is said to depend on the component to which it goes. The dependency is shown by a clashed arrow between components. Stereotypes may be used to define the type of dependency. The component which the arrow leaves is said to depend on the component to which it goes. As shown in Fig.6, the KBS is divided into four parts. They are KbsSupp, AppsPackage, ResPackage and KbsSupp.db.

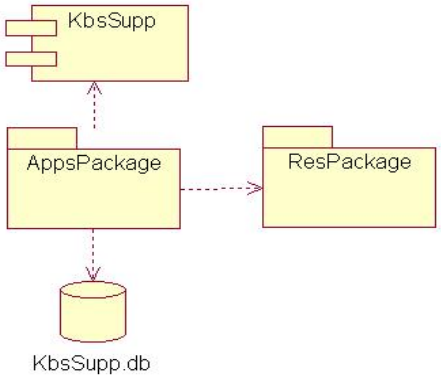


Fig. 6 The component diagram of KBS

The manufacturing pattern and the interfaces of the KBS are stored in KbsSupp. AppsPackage is a package that offers many built-in functions for common usages of KBS. Based on the scheduler's or the algorithm's request. The AppsPackage would obtain patterns and interfaces information from KbsSupp to perform the required tasks. AppsPackage also needs ResPackage's support. The latter provide for the former some basic functions, such as access the database.

KbsSupp.db is a back-end database that used for storing the configurations, variables and rules. The scheduler will query the KbsSupp.db via AppsPackage when it does an online dispatching task. If there is a sutiable rule in KbsSupp.db, AppsPackage will directly deliver it to the scheduler. If no acceptable rule exists, the scheduling knowledge acquisition algorithm will learn the training cases, generate new knowledge and correlate the scheduling objectives, the current problematic production state and dispatching rule at the request of AppsPackage. Then AppsPackage submit the resolved rule to the scheduler. The new knowledge will be updated into KbsSupp.db if the rule satisfies the scheduler. Otherwise this knowledge will be rejected and a new request of AppsPackage will be repeated for the acquisition algorithm's help.

The deployment diagram shows what runs where. As depicted in Fig.7, only Monitor.exe, ResClient.dll, KbsSupp.dll, AppsServer.dll and ResServer.dll are needed in the client PC. The design-time components, such as KbsSupp, AppsPackage, ResPackage and KbsSupp.db, are not necessary. In other words, users do not need to install the expert system tools package and copy the related program source/database to the target PC during deployment.

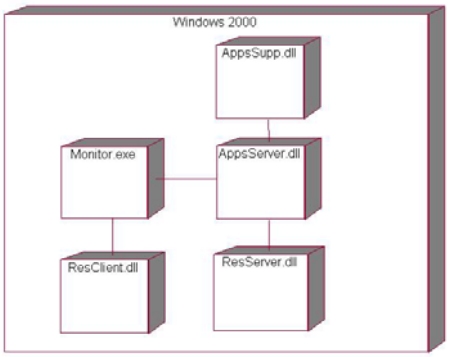


Fig. 7 The deployment diagram of KBS

Monitor.exe supply well graphical user interfaces for the system administrator's querying and viewing system. It needs ResClient.dll to start the program. Many application programming interfaces (API) are encapsulated in AppsServer.dll. During execution, it obtains the pattern and basic interfaces information from the KbsSupp.dll and calls other functions from ResServer.dll. The scheduler and the acquisition algorithm can directly call APIs in AppsServer.dll to realize the complex operation.

There is a main difference between the component diagram and the deployment diagram. The former do not show instances of components, merely the dependencies which apply between all components of one type and all components of another. Instances are shown in the latter.

4 Application

To confirm the feasibility and reliability of our propositions, the KBS for adaptive scheduling (AS-KBS), which is a package of adaptive scheduling system (AS-SCHED) for the workshop in a spacecraft company, is developed. The workshop is considered as a manufacturing environment that produces a low-volume/high-variety of discrete jobs.

AS-KBS is developed in Visual C++. The developed AS-KBS package can be linked with other applications in C++ language via Visual C++. The package is integrated with other packages in AS-SCHED using the "Loose Coupling" integration methodology [7], in which individual intelligent system components communicate with each other using the eXtensible Makeup Language (XML) file.

The colored part in Fig.8 is the position of AS-KBS in the entire system. The main modules that belong to AppServer.dll or AppsSupp.dll in KBS are shown in the part.

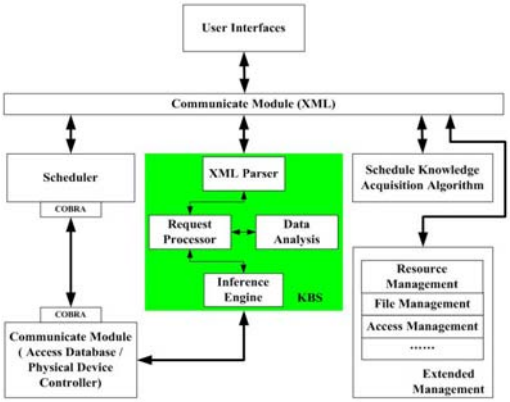
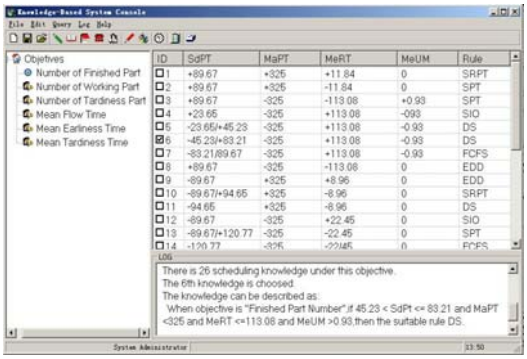


Fig. 8 The architecture of AS-SCHED

AS-SCHED has been applied in a jobbing shop of the spacecraft company. AS-KBS provides a smart service for other packages of AS-SCHED according to their resolutions. AS-KBS has perfect performance in practice. The one of user interfaces of AS-KBS is shown in Fig.9.



Notes:
SdPT: The standard deviation of the total processing time of candidate jobs;
MaPT: The maximum total processing time of candidate jobs;
MeRT: The mean remaining processing time of candidate jobs;
MeUM: The mean utilization of machines.

Fig. 9 User Interface: Knowledge-Based System Console

5 Conclusions

A successful KBS for supporting an adaptive scheduling system would facilitate real time decision making, be robust to various to production requirements. This work introduces an approach that UML is utilized to model the KBS. The proposed method makes the heavily dependent on the domain knowledge experts alleviated. Experts, system

designers, system developer and system administrator can effectively communicate in a unified way and articulates their thoughts. So the quality and efficiency in developing the KBS can improve greatly. The KBS that based on the proposed approach has been implemented, tested and validated in the spacecraft manufacturing company.

Acknowledgements

This work is partially supported by the COSTIND (Commission of Science Technology and Industry for National Defense) Research Project, China, under Contract 20030119.

The author gratefully acknowledges the contributions of the KBS Working Group. Its members have supplied significant amounts of labor in the developing AS-KBS, and without their hard work, AS-KBS could have never been developed as fully as it existed today.

References:

[1] S. C. Park, N. Raman, M. J. Shaw, Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach, *IEEE Transactions on Robotics and Automation.* Vol.13, No.4, 1997, pp 486-502.

[2] K. S. Metaxiotios, D. Askounis, J. Psarras, Expert systems in production planning and scheduling: A state-of-the-art survey, *Journal of Intelligent Manufacturing*, Vol.13, 2002, pp 253-260.

[3] H. Aytug, M. A. Lawley, K. McKay, S. Mohan, R. Uzsoy, Executing production schedules in the face of uncertainties: A review and some future directions, *European Journal of Operational Research*, Vol.161, 2005, pp 86-110.

[4] A Hakansson, UML as an approach to modeling knowledge in rule-based systems, *In Proceedings of ES 2001, the Twenty-first SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligent*, 2001,pp 187-200.

[5] W. W. C. Chung, J. J .F. Pak, A case study: using UML to develop a knowledge-based system for supporting business in a small financial institute, *International Journal of Computer Integrated Manufacturing*, Vol. 19, No.1, 2006, pp59-68.

[6] James. Rumbaugh, Ivar. Jacobson, Grady. Booch, *The United Modeling Language Reference Manual*, Addison-Wesley, Reading, Mass, 1999.

[7] L. R. Medsker, *Hybrid intelligent systems*, Kluwer, Academic Publishers, Boston, MA, 1995.