생산통제용 일정계획 편집기(Gantt Chart Editor)의 객체지향적 설계†

김승권*, 김선욱**, 홍윤호*, 이준열*

고려대학교 산업공학과*, 단국대학교 산업공학과**

ABSTRACT

생산일정계획 수립 시스템을 통하여 수립된 생산계획을 현장에서 실천하고자 할 때, 일반적으로 계획수립 후 얼마 지나지 않아 계획수립시 고려할 수 없었던 기계고장 또는 자재조달 지연, 작업지체, 주문취소 등과 같은 요인으로 생산일정을 계획대로 실천할 수 없는 경우가 발생한다. 이런 상황에서는 기존의 생산계획을 계속 실행할 수 없으므로 손쉽게 대처할 수 있는 방안이 필요하다.

본 연구에서는 일정계획편집기를 이러한 문제점을 해결할 수 있는 대안으로 제시한다. 일정계획편집기를 GUI, 객체지향적 설계를 통해 구현하므로서 이상상황과 일정변동에 효과적으로 대처할 수 있다. GUI환경을 통해, 수립된 생산일정과 이상상황, 작업진척, 수행도 등을 화면에 Gantt 도표로 표시해 주어 현장 파악을 용이하게 하고 작업추가, 삭제, 작업시간 변경, 대체공정으로의 이동 등의 편집을 mouse와 icon을 이용 편리하게 수행할 수 있다. 편집작업시 자동으로 생산일정의 타당성을 만족시켜 줄 수 있는 지능화를 위한 기초 단계로서 객체지향설계를 이용한 규칙기반 일정계획 편집기를 설계한다.

Key Words

생산일정계획, 객체지향적 설계, 일정계획 편집기,
Graphics-User Interface(GUI)

1. 서 론

본 연구진은 중소기업을 대상으로 운영일정 생산계획을 수립해 주는 통합생산관리 시스템(IPSS) 구축을 위한 연구를 수행해 왔다.[6] 이 연구에 연속해서 통합생산관리 시스템에 생산통

†본 논문은 한국과학재단의 목적기초 연구비 지원으로 이루어졌음.(과제번호 90-02-00-06)

제 측면을 보강하려는 목적으로 일정계획편집기를 개발하고자 한다. 일정계획기 개발의 의도는 생산통제과정에서 사용자 인터페이스를 통해 융통성 있는 일정운영을 가능하게 하려는데 있다. 생산계획 소프트웨어에서 자재소요와 부하를 고려하여 생산일정계획을 수립한다 하더라도 생산일정계획을 사용자가 현실상황(인력, 자재변동, 기계고장)을 반영해서 수정하기가 쉽지 않고 - 즉, 전체적인 재 일정계획을 피하면서 신속히 대처하기 쉽지 않고 -, 생산계획자의 관리경험을 반영하기도 쉽지 않다. 따라서, 생산일정계획을 쉽게 이해할 수 있고 대화를 통해 성능향상도 꾀해 볼 수 있는 환경이 필요하다.

그 대안으로 일정계획을 그래픽으로 표현하며, 이것에 사용자가 편집을 통해 생산일정계획에 유연성을 부여할 수 있는, 또한 사용자의 지식을 쉽게 이용할 수 있는 컴퓨터-사용자 체계를 개발하고자 한다. 이 체계에서는 사용자의 편의성을 위하여 시스템이 생산계획과 통제시에 발생하는 제약을 고려해 주고 운영과정의 많은 부분을 시스템이 해결해 주어야 한다. 이유는 사용자가 생산계획과 통제시에 가지는 과다한 정보로 인한 하중을 줄여서 창의적인 일에 집중할 수 있도록 하기 위해서 이다.

본 연구에서는 이러한 문제의 해결을 위해 생산통제용 규칙기반(Rule-based) 일정계획 편집기를 설계해 보았다. 기존의 많은 생산일정 및 통제용 소프트웨어들은 절차식 언어나 인공지능 언어로 구현되었는데 반해 본 논문에서는 객체지향프로그램(Object Oriented Programming)을 시도하였다.[1] 모듈화를 통해 각 기능들을 객체로 분리 독립시켜 사고할 수 있어서 문제의 복잡도를 줄일 수 있으므로 초기 설계 뿐만 아니라 시스템의 요구사양 변경, 확장이 용이하다. 일정계획 편집기는 객체지향 언어인 C++로 작성된다. 이 언어는 완전한 동적바인딩이 제공되는 것은 아니지만 객체지향프로그램 언어로서 널리 이용되고 객체지향설계를 적용하는데 용이하다.[2,3] 이 프로그램 언어는 FORTRAN, PASCAL 등 구조적 언어에 비해 지식표현과 모듈화가 쉽고 인공지능적 기법들을 응용하기에 편리하다.[2]

2. 본 론

가. 시스템 개요

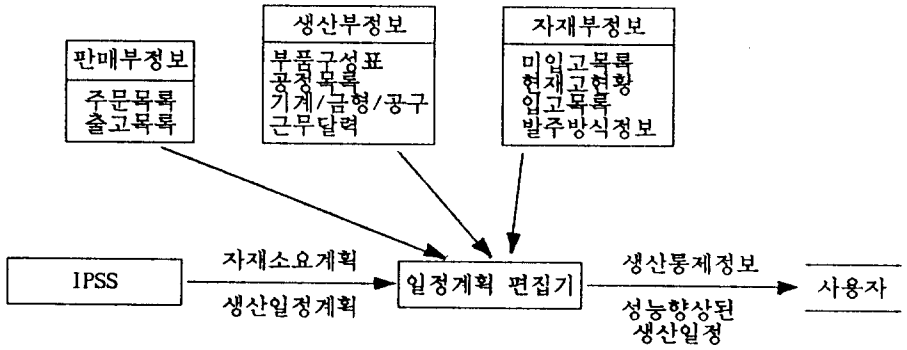

그림 1 구조도

일정계획 편집기는 이미 수립된 일정계획과 각종 생산관련 정보를 입력 받아 실행이 되므로 생산일정계획기가 필요하다. 일정계획 편집기의 정보 입력원이 되면서 같이 하나의 시스템이 되는 것이 본 연구진이 개발해 온 통합생산관리 시스템(IPSS, Integrated Production Scheduling

System)이다.[6] 일정계획편집기는 전체 시스템 중 제일 하위 단계에 해당한다. 즉, IPSS에서 운영일정단계의 계획을 수립하면 이를 입력 받아 사용자와 대화를 통해 일정계획을 향상시키고 현장통제기능을 보조하도록 설계했다.

IPSS는 중소기업용 통합생산관리 시스템으로서 판매부, 자재부, 생산부 업무를 통합시켜 같은 환경에서 업무를 처리할 수 있게 하였다. 사용자가 정해 준 계획기간에 대해 각종 생산정보와 주문정보를 이용해서 자재소요계획, 생산일정계획을 내준다.

IPSS는 운영일정계획에 촛점을 맞추어 설계, 구현된 것이다. 따라서 생산통제의 측면보다는 계획 측면이 강조된 시스템이다. IPSS의 경우 생산계획이 세워진 후 (일일, 주간, 월간) 생산일정계획을 뽑아 본 후에, 이것을 생산관리자가 직접 진도관리하면서, 통제해 주어야 한다. 그리고 생산계획이 크게 차이가 날 경우 다시 생산계획을 수립 해야 한다. 여기에 사용자와 대화를 통해 생산일정을 변경하고 통제할 수 있는 일정계획편집기를 도입한다면 IPSS의 전체시스템의 강화될 것이다.

나. 주요 기능과 구성

편집기는 생산일정계획을 모니터 화면에 일일 단위로 간트 차트형식으로 보여 준다. GUI환경 속에서 사용자가 작업시간 변경, 작업순서 변경, 작업삭제, 작업 추가 등을 수행할 수 있으며 이것을 실행한 즉시, 작업물의 선후행관계, 자원제약을 고려하여 실행가능한 일정계획을 유지해 준다. 이 과정을 통해 일정계획을 사용자의 경험에 비추어 현실성을 가지도록 사용자가 편리하게 수정할 수 있다.

본 시스템에서는 제품목록, 부품목록, 원자재목록, 자재 소요계획과 미입고 목록, 일정계획 정보, 공정목록, 부품구성표, 주문목록을 정보로 하여 사용자가 편집을 하였을 경우에 제약조건을 형성하여 실행 불가능한 생산일정의 발생을 막아 주며 관리정보를 가공하여 화면에 표현해 준다.

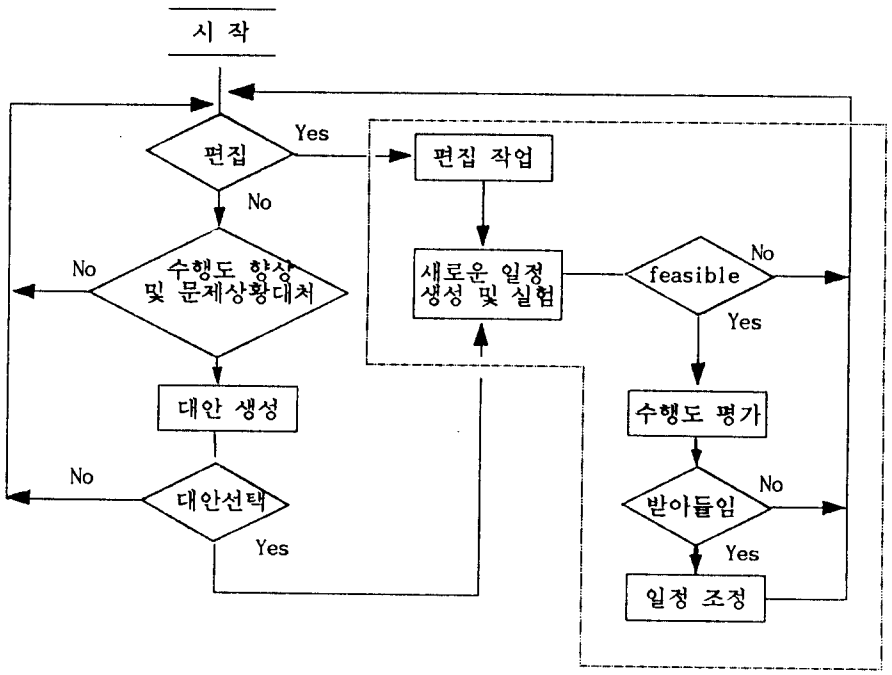

그림 2 순서도

편집기의 설계범위는 그림 2와 같다. 본 논문에서 설명하는 부분은 그림에서 점선으로 표시된 부분이다. 그림에서 '새로운 일정생성 및 실험'와 '대안생성시'에 규칙베이스(rule base)을 이용하게 되어 있다. '대안선택'의 경우에는 '대안생성'부에서 만들어진 일정변경안들을 선택하도록 하는 것이며 '새로운 일정생성 및 실험'는 '편집'과 '대안선택'을 통해 변경된 일정의 실행가능성을 평가한다.

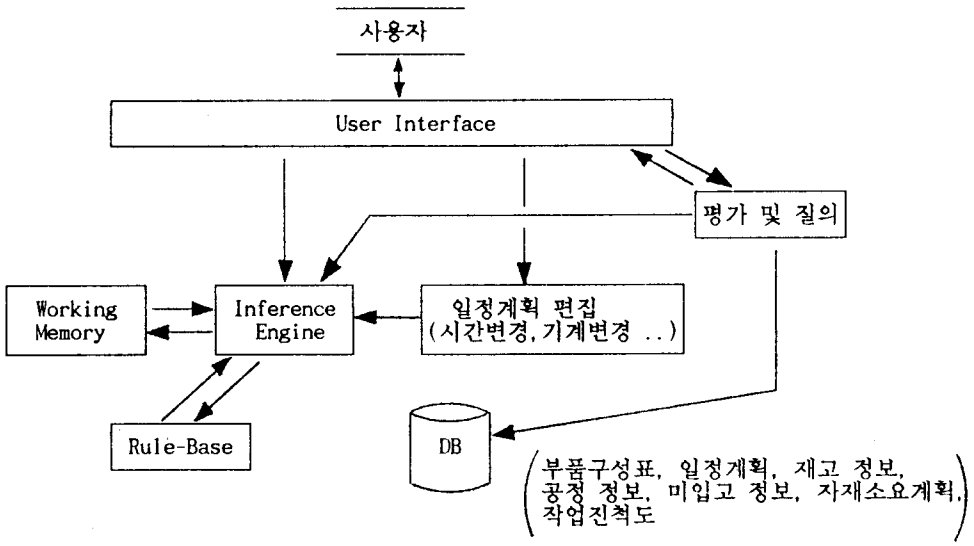

그림 3 편집기 구성도

다. 규칙기반시스템(Rule based system) 설계

편집기는 여러 제약 조건내에서 실행가능한 일정계획을 내주어야 하므로 고려해야 할 제약이 많고 앞으로 성능개선을 위한 지식을 추가할 계획이므로 규칙기반시스템으로 구현하는 것이 바람직하다.

현재 rule-base를 활용하는 일정계획 편집기의 구조를 그림 3에 나타내었다. 규칙기반시스템이 작동되는데 있어서 중요한 구성요소는 다음과 같다.

1. Rule-base, 2. 추론기관(Inference engine), 3. Working memory

Rule-base에는 현재 편집을 시도 했을 경우에 제약조건이 어겨진 상태를 해결해 주는 통제규칙(control rule)을 저장한다. 이것은 앞으로 성능향상을 위한 지식과 문제상황 대처를 위한 지식이 추가로 입력될 것이다. 추론기관은 전진추론(forward chaining)을 하도록 설계하였다. 충돌해소(conflict resolution)방법으로는 크기순서화(Size ordering)과 문맥제한(Context limiting)방법을 사용했다.[7]

Working memory에는 rule base에서 사용하는 변수들의 값들과 임시저장변수를 저장하고 있으며 제약조건을 어긴 상태를 실행가능한 상태로 만들어 가는 과정에서 일정의 중간단계들을 저장해 놓는다.[4]

위 3가지 구성요소들을 위한 객체를 살펴보면 다음과 같다.

1. rule base를 구축하기 위한 객체.

Class	Member	설 명
Predicate	instance variable Relation Subject Object	 관계 변수 변수 또는 값
Rule	instance variable rule_num ifhead, iftail Thenhead, Thentail member function IF() AND() THEN()	 Rule 번호 IF부분 predicate들을 저장 Then부분 Predicate들을 저장 IF부에 predicate 저장함수 predicate 추가함수 THEN부에 predicate 저장함수
Rule_Set	instance variable Number_of_Rule head, tail member function Append_rule() Delete() Find()	 Rule의 갯수 Rule 저장 Rule의 추가 Rule의 삭제 Rule 검색

2. working memory를 이루는 객체.

Class	Member	설 명
queue_element	instance variable job	최종 결정된 작업정보를 가진다
Queue	instance variable head, tail member function Enqueue() Head() Dequeue() Empty()	 queue_element 저장 queue에 추가 queue의 맨 앞에 있는 요소 참조 queue에서 원소를 빼냄 queue가 비었는지 확인
Stack_Element	instance variable job NewMachine NewDate new_start_time	 stack에 저장할 job 옮겨질 새로운 기계 옮겨질 날자 새로운 시간

Class	Member	설 명
Stack	instance variable	
	list_length	Stack 크기
	first_item,current_item	Stack_element 저장
	member function	
	push()	Stack에 입력
	pop()	Stack에서 Stack_element를 꺼냄
	top()	Stack의 맨위에 있는 원소 참조
	empty()	Stack이 비어 있는가 확인
Working_Memory	instance variable	
	Stack job_stack	일정의 feasible상태를 조정하는 과정에서 중간 결과 저장
	Queue job_queue	최종 결과 저장
	그 밖에 임시변수들	임시변수의 값을 저장
	member function	
	input_condition()	변화된 상황을 입력

3. 추론기관(inference engine)을 위한 객체.

Class	Member	설 명
True_Rule_Num	instance variable	
	rule_num	Rule 번호 저장
True_Rule_Num_Set	instance variable	
	true_rule_num	trigger된 Rule의 갯수
	head, tail	True_Rule_Num를 저장
	member function	
	Add_Rule_Num()	trigger된 Rule 번호 저장함수
	DeleteAll()	Rule 삭제
Forward_Chaining	instance variable	
	rule_set	Rule_Set을 저장
	true_rule_set	True인 Rule 번호를 저장
	working_memory	Working Memory
	member function	
	Inference()	추론 시작
	Match()	Rule의 if조건이 성립여부 검사
	Add_True_Rule()	trigger된 룰 번호 저장
	Conflict_Resolution()	conflict resolution
	Fire()	룰을 fire시킨다.
	기타 룰 fire시 작동되는 procedure들	

현재는 편집작업시 제약조건을 만족시키면서 실행가능한 일정을 만들어 줄 때 이용되는 통제규칙들이 rule-base에 저장된다. 아직은 수행도 향상을 위한 지식과 문제 상황대처를 위한 지식이

들어 있지는 않지만 현재 rule-base는 새로운 지식을 추가하기 쉬운 형태로 되어 있으므로 기본 골격은 잡혀 있는 상태이다. 지식획득과 부수적인 알고리즘을 모듈로 짜서 넣으면 된다. 이것은 객체 단위로 모듈화가 되어 있으므로 시스템의 전체 구조를 흐트리지 않고 확장 가능한 객체지향 언어의 장점이라고도 할 수 있다.

라. 객체설명

Ivar Jacobson[3]은 객체지향형 시스템 설계에 있어서 크게 interface object, entity object 및 control object 등으로 구분하였다. 전체객체를 이에 따라 분류하고 객체의 직계구조를 나타내었다.

(1) **interface objects**

사용자와 시스템이 서로 대화를 주고 받기 위해서 정의되는 객체(종류)를 말하는 것으로, icon, button, dialog box 등과 같은 GUI용 객체들이 여기에 속한다.

(객체의 직계구조)

Icon (아이콘 생성 객체)
Button (버튼 생성 객체)
Scroll (스크롤바 생성 객체)
Draw_Box (상자 화면 출력 객체)
Gtext ─┬─ Gputs - Gprintf - Gprintfxy - TextCenter (문자열 출력)
└─ Ggetche - Ggets - Gscanf - Gscanfxy (문자열 입력)

DrawingTool
- Info (작업물에 관한 정보 출력)
- Pathobj (작업물이 거쳐 가는 경로 보기)
- SaveTool (편집된 일정계획 저장하기)
- MoveJob (작업물을 이동)
- Erasejob (작업물 삭제)
- Appendjob (작업물 추가)
- MoveJob (작업물 이동)
- ChangeJob (작업물 교환)
- MachineScrollUp (기계목록 scroll up)
- MachineScrollDown (기계목록 scroll down)
- DateSkipDayForth (다음날로 날자옮김)
- DateSkipDayBack (전날로 날자옮김)
- DateSkipWeekForth (일주일 후 날자로)
- DateSkipWeekBack (일주일 이전 날자로)
- ToggleTool (한글/영문 모드 변경)
- QuitTool (작업종료)

Interact ── UserInteract (class DrawingTool의 객체들을 메모리에 등록하여 사용자와 대화가 가능하도록 입출력을 담당하는 객체)
Gantt(class UserInteract의 객체들의 메시지를 관리하는 객체)
MouseObj ── KbdMouseObj (마우스 동작 관리자)

(2) **entity objects**

시스템이 오랫 동안 계속 취급하는 정보를 entity object로 모델링한다. 재고정보, 미입고 정보, 주문 정보, 공정목록, 일정계획을 관리하는 객체들이다.

(객체의 직계구조)

```
LinkedList ─┬─ Job_list        <-> Job                 (일정계획)
            ├─ OperationList   <-> OperationListNode   (공정목록)
            ├─ BOM_list        <-> BOM                 (부품구성표)
            ├─ OHinv_list      <-> OHinvNode           (자재 및 현재고목록)
            ├─ SRinv_list      <-> SRinvNode           (미입고목록)
            ├─ Progress_list   <-> ProgressNode        (작업진척도)
            ├─ Reqmatrl_list   <-> ReqmatrlNode        (자재소요계획)
            └─ Ruel_Set        <-> Rule <-> Predicate  (Rule Base 객체)
Queue <-> queue_element
Stack <-> Stack_Element
True_Rule_Num_Set <-> True_Rule_Num
                    (여기서 "<->"는 friend 관계)
```

(3) **control objects**

interface, entity object 어느쪽에도 속하지 않으며, 앞의 두 object type들을 연결시켜 주며 완충시켜 주는 역할을 한다.

(객체)

Forward_Chaining	(forward chaining하는 inference engine)
Working_Memory	(임시저장기억장소)
Score	(일정의 성능평가를 위한 객체)

3. 결론 및 향후연구

객체지향설계를 이용한 Rule-based 일정계획 편집기를 설계하는 것을 살펴 보았다. 이 편집기에는 사용자에게 도움을 주는 정보의 그래프화가 시도되었고, 사용자의 지식을 이용할 수 있는 대화환경을 제공하므로서 생산일정계획의 실용성, 수행도 향상을 기대할 수 있다고 생각한다. 이것은 제약 조건들을 검사하여 실행가능한 일정을 만드는 통제규칙뿐만 아니라 일정들의 성능향상을 위한 지식을 수용하는 규칙들을 추가하는데 용이한 틀을 제공한다. 앞으로 문제 상황 대처에 관한 지식, 사용자가 정해준 수행도 향상을 위한 지식을 획득하여 rule base에 추가하므로서 일정계획을 융통성 있게 운용할 수 있으리라 기대된다. 그러므로서 현재 시스템에 생산통제기능이 본격적으로 갖추어 지게 될 것이다.

향후연구계획으로는 사용자가 지정한 다중평가기준 측면에서 성능을 향상시킬 가능성이 있는 작업물에 대한 변경을 제시해 주는 지식기반 일정계획 편집기(Knowledge Based Gantt Chart Editor)에 대한 연구가 기대된다. 이 일정계획편집기는 사용자가 수행해야 하는 많은 정보처리과정을 대신해 주므로서 사용자로 하여금 일정변동상황에 대한 예측을 쉽게 알 수 있도록 하므로서 생산성을 향상시킬 수 있다. 또한 계획에서 벗어나는 이상상황을 입력 받아서 전체적으로 다시 일정계획을 세우지 않고도 짧은 시간 안에 재 일정계획을 수립할 수 있는 기능을 갖추게 될 것이다.

4. 참고 문헌

[1] Andrew Kusiak, Intelligent Manufacturing Systems, prentice hall, 1990.

[2] Ernest R. Tello, Object-Oriented Programming for Artificial Intelligence:

[3] Ivar Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

[4] Tony Hasemer, John Domingue, Common LISP programming for Artificial Intelligence, Addison-Wesley, 1989.

[5] Wolfgang Kreutzer, Bruce McKenzie, Programming for Artificial Intelligence: Methods, Tools and Applications, Addison-Wesley, 1990.

[6] 김승권, 김선욱, 이준열, 홍윤호, "대화식 생산일정계획 수립을 위한 객체지향형 설계", 한국경영과학회/대한산업공학회 '93 춘계공동학술대회 발표논문 및 초록집, pp.83-92.(1993)

[7] 황종선, 정태충 공역, P. H. Winston원저, 인공지능, 生能.