유전자 알고리즘을 이용한 매트릭스조직의 객체지향개발 프로젝트 스케줄링

이 건 호* · 김 은 진*
*숭실대학교 산업정보시스템공학과

Project Scheduling for Object-Oriented Development in Matrix Organization

Gun Ho Lee* · Uen Jin Kim*
*Department of Industrial/Information Systems Engineering, Soongsil University

Abstract

This paper discusses a scheduling problem on object-oriented developments over multiple teams with limited resources in matrix organization.

The objective of the problem is to minimize the makespan of overall projects. There are tangible and intangible advantages such as efficient resource share, improvement of productivity, development efforts and cost reduction, etc. by dispatching resources properly to the development teams. Traditionally, the project scheduling has been done with a manager's intuition or heuristic.

We present a scheduling model with illustrative examples, stochastic search approach, and apply a variety of problems generated randomly to the approach. The results are analysed.

Keywords : Matrix Organization, Object Oriented Development, Project, Scheduling

1. 서 론

다양한 프로젝트들의 효율적인 관리는 기업의 핵심적인 성공요인이 되고 있다. 객체지향형 소프트웨어 개발 프로젝트 스케줄링에서 다수의 프로젝트 진행을 동시에 수행할 경우 각 프로젝트 작업의 선행관계와 제한된 인력자원으로 전체 프로젝트의 납기일 준수를 위한 계획과 관리는 어렵고 복잡한 문제이다.

매트릭스 조직에서 다수의 객체지향 개발 프로젝트를 수행하는 것은 소프트웨어개발 관계자에게 많은 관심의 대상이 되고 있다.

매트릭스 조직에서 객체지향 개발의 특징은 사이클이 반복적으로 진행 되며, 프로젝트에 필요한 자원의 기능부분을 추출하며 먼저 완료된 프로젝트의 유휴 인력들이 다른 프로젝트의 투입 될 수 있도록 하여 자원의 활용도를 높이는 것이다.

본 연구에서는 유전자 알고리즘을 이용하여 매트릭스 조직에 적합한 프로젝트 스케줄링방법을 제시하고자 한다. 프로젝트 스케줄링은 제한된 인력자원을 최대한 활용하여 각 프로젝트 납기일 내에 완료하는 조건으로 전체 프로젝트의 총 처리시간(makespan)의 최소화를 목적으로 한다.

2007년 6월 접수; 2007년 9월 수정본 접수; 2007년 9월 게재확정

프로젝트 스케줄링의 연구로는 소프트웨어 개발을 위한 퍼지 프로젝트 스케줄링 시스템 [1], 퍼지 PERT를 이용한 시스템 개발 스케줄링 [2], PERT/CPM에서의 프로젝트 완료시간 예측과 주경로 파악 및 통제를 위한 퍼지기법 [3], 자원제약 하의 복수 프로젝트 일정계획을 위한 휴리스틱 기법 [4], Constraint Programming을 이용한 자원제약 동적 다중 프로젝트 일정계획[5] 등이 있다. 연구 [6]은 개발주기가 반복되지 않는 전통적인 소프트웨어 개발모형인 폭포수모형을 대상으로 하여 구조적 개발방식인 하향식(Top-Down) 개발에 적합한 일정계획 수립방법을 제시하고 있다.

작업우선순위를 고려한 스케줄링 [7], 스캐줄링을 위한 퍼지 최적 모델[8], 카테고리 프로젝트 일정계획에 적용하는 방법[9], 자원제약 하의 다중모드 스케줄링 [11], PERT 네트워크에 time-cost를 적용하는 연구 [12]가 있다.

연구 [10]은 제한된 인적자원으로 다수의 프로젝트를 동시에 혹은 순서상으로 일정계획을 수행하지만 각 프로젝트는 개발주기가 반복하는 객체지향형 개발에는 적용이 불가능하다.

객체지향 프로젝트 관련 연구로는 개체지향 시스템의 중소규모 개발 프로젝트 [13], 객체지향 시스템 개발에서의 UC 연구 [14], 동적스케줄링의 리얼타임연구 [15], 객체지향 시뮬레이션 리얼타임연구 [16] 등이 있다. 객체지향 스케줄링에 관한 문헌과 연구로는 소프트웨어스케줄링 [17], 객체지향 어플리케이션 개발방법 [18], 객체지향 스케줄링연구 [19] 등이 있다.

Bruegge와 Dutoit [20]는 작업분해구조를 이용한 태스크 식별을 거쳐 태스크 사이의 의존도를 분석한 뒤, 태스크를 스케줄에 매핑하는 기법을 연구 하였다.

Pooley와 Stevens [21]는 객체지향 프로젝트에 적합한 기법을 제시하였고 Canto[22]는 다른 연구에 비해 상세한 지침을 3레벨로 나누어 작성하게 하는 것을 제안하였다. Fayad외 2인 [23]은 OO로 변화된 프로세스에 대해 계획 및 선행 프로젝트 단계, 기술 삽입단계, 프로젝트 관리 3단계로 나누어 연구하였다.

허진선 외 2인 [24]은 Use-Case(UC) 기반 객체지향 프로젝트 스케줄링 기법을 제안하였다.

시스템의 기능적인 요구사항이 기술된 UC 다이어그램을 이용하여 객체지향 프로젝트 스케줄을 도출해가는 과정을 UC 식별, 상호의존성 분석을 통한 초기 PERT 차트작성, 각 UC의 특성 규명, Iteration 개수 결정, Iteration에 UC 할당, 유용한 자원과 제약 사항 고려, Revised PERT 차트작성의 7단계로 나누어 제안한다. 각 단계에 대한 입력물과 중간 산출물, 그리고 수행지침을 제시하였다.

하지만 프로젝트매니저의 역할에 크게 비중을 두어 매니저의 역량에 따라 프로젝트의 성공여부가 정해지는 한계점이 있다.

객체지향 프로젝트에 대한 다양한 연구가 수행 되었지만, 다수의 프로젝트를 동시에 수행할 수 있는 매트릭스 조직의 객체지향 일정계획에 대한 연구는 이루어지지 않았다.

따라서 본 연구에서는 유전자 알고리즘을 이용하여 객체지향형 매트릭스 조직에 적합한 객체지향개발(Object Oriented Development; OOD) 프로젝트 스케줄링 방법을 제시하고자 한다.

2. OOD 스케줄링과 매트릭스 조직

OOD는 소프트웨어 시스템을 작은 단계로 나누어 반복적이고 점증적으로 개발한다. UC로 시스템을 분할하고 초기에 핵심적인 S/W 아키텍처를 구성한다는 점에서 Prototyping 기법과 다르다. 객체지향의 개발에서는 Prototyping과 비슷한 원리로 조기 피드백으로 위험을 최소화 하고 점증적으로 개발하므로 새로운 요구사항 및 변경사항에 대하여 유연하게 대처할 수 있어 클래스의 재사용성과 확장성이 우수하다.

일반적으로 객체지향 개발의 전체 스케줄링에 대한 절차는 개념화 영역, 형상화 영역, 솔루션화 영역으로 나누고 더 세분화하여 모델링 단계, 설계 단계, 컴포넌트 단계, 아키텍처 단계, 솔루션 검증 단계 순으로 진행할 수 있다. 각 단계에서는 문서 산출물과 각 단계에 진행될 세부사항들이 정의 되어 있다[24].

본 연구에서는 모델링 단계에서 문제의 정의, 요구사항 분석을 한다. 동시에 프로젝트 개발을 위한 태스크를 식별하는데 해당 문제에 대한 진술서, 라이프사이클의 활동들을 고려한다. 사용자와 프로젝트 관리자는 프로젝트 진행사항 검토에 대한 동의 하에 각 산출물을 최소한 하나의 태스크에 할당하고, 작업을 한 사람이 평가하고 수행하기 편리한 단위로 작업분해구조로 분해한다.

요구사항과 산출물이 상세할수록, 태스크를 정의하기 쉬우나 잘 정의되지 않은 요구사항은 매니저가 태스크 계획을 수립한다. 작업분해구조는 지나치게 상세하지 않도록 하고 태스크를 식별을 마친 후 태스크별로 UC 다이어그램을 작성한다. 본 연구의 다수의 프로젝트를 고려하는 스케줄링에서 PERT차트작성을 위한 과정은 허진선 외 2인 [24]의 7단계와 유사하다.

1단계에서 요구사항분석 후에 UC를 식별하고 개발해야 할 기능모델인 UC 다이어그램에 기술된 UC들을

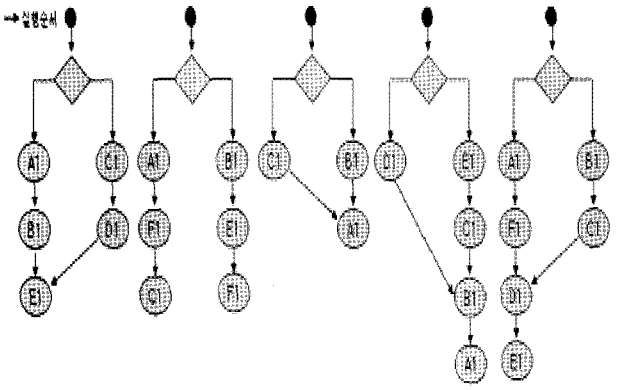
나열한 뒤 기능적 고찰을 수행한다. 각 UC의 기능에 대해 파악하는 과정에서 UC의 세분성(granularity)을 검토한다.

단계2에서는 UC들 간의 상호의존성을 분석하여 개발 순서상의 전체적인 UC의 개발순서를 파악하여 계획단계에 반영한다. 전체 UC간의 실행순서에 대한 개요를 작성한 후 병렬적으로 수행 가능한 UC 군으로 분류하고 분류된 UC 내에서 실행순서가 명확히 드러나는 UC를 표시한다.

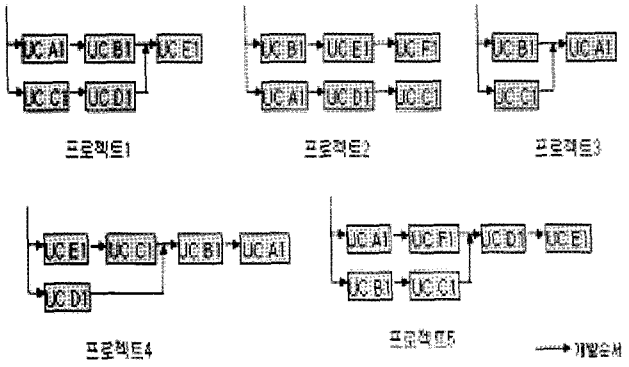
UC 다이어그램에 명시적으로 드러나는 UC간의 실행순서는 Include, Extend, Generalization 등의 UC 간의 의존성을 통해 알 수 있다.

UC를 이용한 Activity 다이어그램을 작성하면 요구사항을 분석 할 때에 UC 다이어그램을 바탕으로 작성된 Activity 다이어그램을 이용하여 전체적인 작업순서를 결정 할 수 있고, UC를 서브시스템으로 나누어 개발 순서를 결정하는 두 가지 방법이 있다. 서브시스템은 서로 상호작용이 많고 비슷한 기능 군이므로 작업시기가 인접하도록 하며 서브 시스템간의 우선순위를 고려하여 스케줄링에 반영한다.

위에서 제시된 지침을 기반으로 UC를 이용한 Activity 다이어그램을 작성한다 <그림 1>.



<그림 1> UC를 이용한 Activity 다이어그램



<그림 2> Preliminary PERT차트

<표 1> UC 특성표

UC ID	특 성			
	복잡도	완성도	우선순위	위험도
P1-UC1	H	H	H	H
P1-UC1	K	M	K	K
P1-UC2	M	L	K	M
P2-UC1	L	M	K	K
P2-UC2	K	K	L	K
P2-UC3	L	M	M	L
P3-UC1	K	M	K	L
P3-UC2	K	K	M	K
:	:	:	:	:

H:높음, M:중간, L:낮음

<그림1>에서 UC이용한 Activity 다이어그램과 Include, Generalization의 상호관계를 기반으로 Preliminary PERT차트를 작성한다.

Preliminary PERT차트는 단계3의 UC의 가중치를 이용하여 명확한 검증을 할 수 있다.

단계3에서는 각 UC의 특성을 분석하여 특성에 따라 수행시기, 수행기간이 변경된다. 단계1의 결과물인 UC 리스트를 입력받아 여러 특성 분류 기준을 통해 분석한다.

각 UC는 복잡도, 완성도(Requirement Completeness), 우선순위(Managerial, Strategical Priority), 위험도 등의 기준을 통해 특성을 분류한 것이 <표 1>의 UC 특성표이다. UC 특성표를 <표 1>과 같이 작성하고 각 UC 의 특성을 비교하기 쉽게 매트릭스를 적용하면 각 UC의 특성값을 얻을 수 있다.

단계 2에서 UC 특성표를 기반으로 하여 각 점수마다 가중치를 부여한 후 가중치 총합을 계산한다. 우선적으로 복잡도만을 고려하면, 복잡도가 높을수록 개발기간을 뒤로 정하는 것이 유리하다.

복잡도가 Yi일때 복잡도 수준이 Yh > Ym > Yl라고 하면 각각에 대한 수의 값은 Yh < Ys < Yl의 크기이며 복잡도가 클수록 작은 값을 부여한다. 또한 완성도가 높은 UC일수록 개발하기 쉽기 때문에 완성도가 높을수록 큰 값을 부여한다.

<표 2> UC 특성 가중치

UC ID	특 성				
	Complexity(Y)	Complexity(S)	Priority(P)	Risk(R)	TotalWeight
P1-UC1	Yh	Sh	Ph	Rh	TWP1-UC1
P1-UC2	Ym	Sl	Pl	Rm	TWP1-UC2
P1-UC3	Yl	Sm	Pm	Rl	TWP1-UC3
P2-UC1	Yh	Sh	Ph	Rh	TWP2-UC1
P2-UC2	Ym	Sl	Pl	Rm	TWP2-UC2
:	:	:	:	:	:

<표 3> 핵심적 기능 추출

UC ID	특 성		
	복잡도	우선순위	Total Weight
TWP1-UC1	Sh	Ph	TW P1-UC1
TWP1-UC3	Sh	Ph	TW P1-UC3
TWP2-UC1	Sl	Pl	TW P2-UC1
TWP2-UC4	Sl	Pl	TW P2-UC4
TWP3-UC2	Sm	Pm	TW P3-UC2
:	:	:	:

<표 2>의 총 가중값은 각 UC에 대한 우선순위를 반영하여 단계2에서 도출된 표 3의 총가중값을 적용하여 Preliminary PERT 차트가 적절히 유도되었는지 검증하여 정제된 PERT차트를 작성한다.

단계4에서는 단계3의 결과인 정제된 PERT차트와 과제의 기간, 프로세스 모델을 기반으로 Iteration의 개수를 결정한다. 그 안에서 결정된 개수에 따라 각 Iteration을 수행하는 동안에 할당된 UC는 할당된 기능 모델링 단계->설계 단계->컴포넌트 단계->아키텍처 단계->솔루션 검증 과정으로 개발 된다.

즉, 각 Iteration에서는 이미 기능적 분석이 완료된 Case모델을 받아서 실행 가능한 프로그램으로 변형할 수 있는 형태로 설계한다.

앞의 UC 특성 가중치 표에서 분석된 결과 중에 복잡도가 높거나 위험도가 높은 UC는 여러 Iteration에 걸쳐 개발을 수행하고 완성도가 낮은 UC는 처음부터 개발하는 것이 아니라, 충분한 이해를 얻은 후의 Iteration에서 수행하는 것이 적합하다.

따라서 복잡도가 높고 UC가 많을수록 Iteration개수를 늘려서 개발을 진행해 나가는 것이 효과적이다. 한 Iteration당 유용한 리소스를 완전히 이용해야 전체 개발기간을 줄일 수 있다. Iteration개수는 프로젝트 진행 상황에 따라 융통성 있게 조절 할 수 있다. 그러나 각 Iteration의 기한은 지키면서 개발을 한다. 만약 한 Iteration 기간 내에 몇몇 작업을 동시에 수행해야 할 경우 병렬처리 수를 적절히 조정 할 수 있다.

단계5에서는 Iteration에 UC를 할당하는 것이다. 핵심적인 기능은 본 시스템이 실행되기 위한 최소한의 핵심 UC를 의미한다. Iteration 에 UC를 할당하는 방법은 2가지이다. 첫 번째는 전체개발 순서대로 그대로 Iteration을 나누어 개발한다. 첫 번째 Iteration 에서는 전체 시스템 요구사항의 한 서브시스템 중 많은 부분을 구현 하고 다음 Iteration 에서는 다른 서브시스템을 구현하는 형식으로 진행된다. 두 번째는 각 Iteration이 진행될수록 핵심 기능에서 부수적인 기능을 추가하는 형식으로 할당할 수 있다.

단계 2에서 중간 산출물로 나온 Activity 다이어그램 상에서 메인 흐름으로 표현되는 UC들과 단계 4의 UC 특성 중에 우선순위와 완성도에 가중치를 더 부여하여 계산한 값을 기반으로 하고 각 Iteration 에 포함될 UC는 앞서 언급한 정보와 프로젝트 매니저의 Heuristic이 적용되어 결정되어 진다.

단계 5에서 정해진 Iteration 개수에 따라 아래의 <표 4>와 같이 UC를 각 Iteration 에 할당한 할당표를 작성 하도록 한다. 위험도를 줄이기 위한 방안으로 여러 반복을 통한 점증적인 개발 프로세스를 적용한다.

모든 상황에 적용되지는 않지만 일반적으로 UC 다이어그램에서 기본적인 CRUD(Create, Read, Update, Delete)의 기능 중 첫 번째 Iteration 에서는 생성관련 UC를 개발하는 경우가 많이 있다.

<표 4> UC 할당

	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Project 1 할당 될 UC	P1-UC1 P1-UC3 P1-UC2 P1-UC4	P1-UC5		
Project 2 할당 될 UC	P1-UC1 P1-UC4 P1-UC4 P1-UC5	P1-UC3 P1-UC6		
Project 3 할당 될 UC	P1-UC2 P1-UC3	P1-UC1		
Project 4 할당 될 UC	P1-UC5 P1-UC4 P1-UC3	P1-UC2	P1-UC1	
Project 5 할당 될 UC	P1-UC1 P1-UC2	P1-UC6 P1-UC3	P1-UC4	P1-UC5

단계6에서는 유용한 자원과 여러 제약 사항을 고려하는데 자원에는 사람, 재사용 가능한 소프트웨어 컴포넌트, 하드웨어, 소프트웨어 툴 등이 있다. 프로젝트에서 요구되는 인력자원을 파악한 후 현재 개발 인력의 능력이나 전문성을 평가하여 인력 평가표를 작성한다.

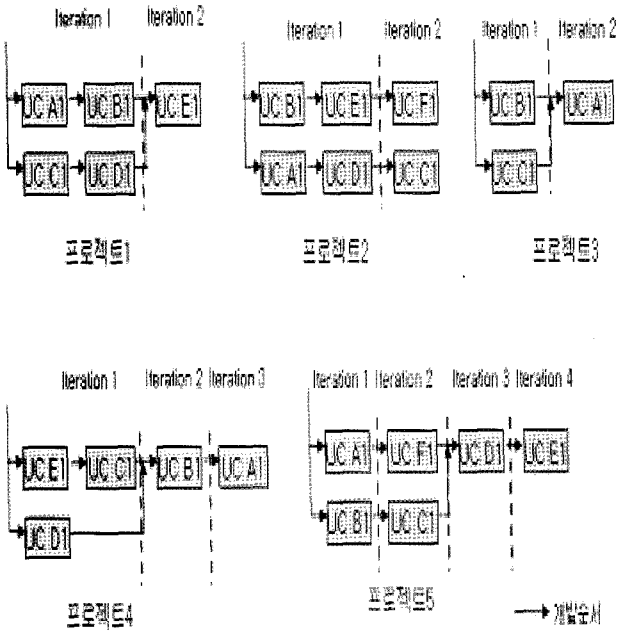
각 작업 별로 인력을 배당은 유휴인력을 최소화하기

위하여 한 프로젝트의 업무를 마감하면 진행 중에 있는 다른 팀을 지원하도록 한다.

단계7에서는 1단계에서 6단계까지의 산출물과 결과로 Revised PERT 차트를 작성한다.

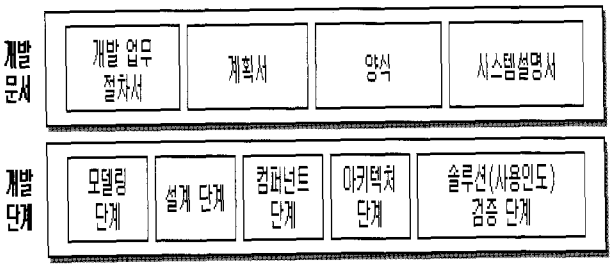
액션중심의 UC modeling은 위의 7단계를 거쳐 보다 상세하게 작성하고 이를 기반으로 자료중심의 Class modeling을 할 수 있다. Revised PERT 차트 참고하여 Class 다이어그램을 작성한다. 동적모델링은 Revised PERT 차트를 기준하여 순차 다이어그램과 협력 다이어그램, 상태 다이어그램, 활동 다이어그램을 작성한다.

설계단계에서는 시스템 아키텍처 및 서브시스템 정의와 컴포넌트 및 클래스 설계와 동시에 코딩환경(물리적)을 구축한다. 컴포넌트 단계에서는 프로그래밍 즉 클래스를 코딩하고 아키텍처 단계로 들어가 기존 데이터 전환 준비, 구현된 컴포넌트 통합, 통합 테스트와 시스템 버전을 만든다. 솔루션 검증 단계에서 시스템 릴리즈 및 시스템 테스트를 하고 요구사항을 충족되면 프로젝트는 완료된다.



<그림 3> Revised PERT 차트

<그림 4>는 프로젝트의 모델링 단계(Modeling Step; MS), 설계단계(Design Step; DS), 컴포넌트 단계(Component Step; CS), 아키텍처 단계(Architecture Step; AS), 솔루션 검증단계(Solutions verification Step; SS)와 개발문서를 나타내고 있다.

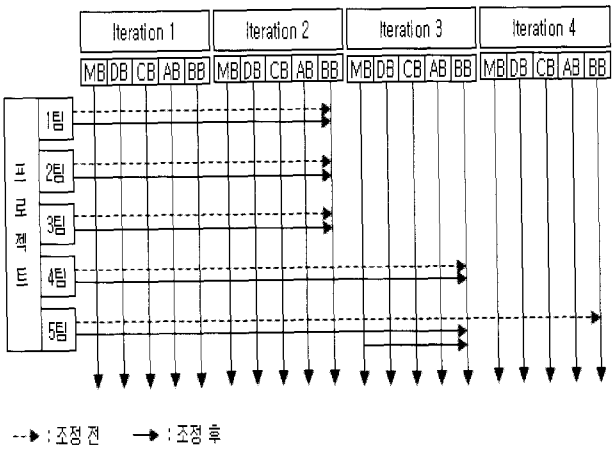


<그림 4> 프로젝트 개발문서와 단계구조

Iteration은 다음의 가정에서 수행되도록 한다.

- Revised PERT 차트 상의 각 Iteration은 MS, DS, CS, AS, SS 순으로 납기일과 각 단계의 일정을 준수하여 진행한다.
- 프로젝트 팀의 유휴자원을 최대한 활용 한다.
- 선행 Iteration의 인력도 가용하면 후행 Iteration에서 투입될 수 있다.
- 임의의 프로젝트를 완료한 자원은 다른 프로젝트에 투입될 수 있다.
- UC 간 의존성 추출표와 UC 특성 가중치를 근거로 선후행 Iteration간에 순서상의 제약조건이 없다면 유휴인력을 투입하여 두 Iteration을 병렬로 수행한다.
- Iteration들 간의 순서상의 제약 때문에 단계별로 개발해야 할 때 다른 프로젝트 종결 후 가용인력을 병목현상이 발생하는 Iteration에 지원하여 개발일정을 단축시킨다. 인력 평가표 및 팀별 자원의 구성과 팀별 지원 가능 자원 상세를 기반으로 한다.

<그림 5>의 객체지향 프로젝트 단계를 매트릭스에서는 Iteration을 4단계로 진행하고 있다. 복잡하고 긴 기간의 프로젝트의 경우 Iteration을 여러 단계로 개발할 수 있을 것이다. 5팀의 4단계는 3단계와 순서상의 제약조건이 없어 병렬수행이 가능하여 프로젝트 시간 단축의 효과를 나타내고 있다.



<그림 5> 프로젝트 개발일정 조정전과 후

3. 매트릭스조직의 N프로젝트 스케줄링

본 연구는 N개의 객체지향개발 프로젝트로 각 프로젝트는 개발과정이 부분적으로 반복되어 점증형으로 완성하는 것을 대상으로 한다.

각 기능을 수행할 수 있는 팀에는 개발인력을 포함하고 각각의 프로젝트에 적절히 배치하여 프로젝트를 수행한다. 소프트웨어 개발에서의 주요 자원은 개발인력이므로 본 연구에서는 개발인력 위주의 일정계획을 수립한다.

<그림 5>와 같이 매트릭스 조직에서는 각 개발인력은 소속 기능팀의 작업을 수행하고, 또한 다른 기능팀을 지원하여 프로젝트를 수행할 수 있다. 프로젝트의 작업을 완료한 유휴 개발인력은 진행되고 있는 다른 프로젝트에 재배치되어 소속팀이 아닌 기능팀에서도 프로젝트를 수행한다.

<표 5> 프로젝트의 자원배치

프로젝트	MS	DS	CS	AS	SS	인력수
P1	1	1(1)	2	1	1	6(1)
P2	1	1(1)	1(2)	1	1(1)	5(4)
P3	1(1)	1(2)	1	1(2)	1(1)	5(6)
P4	2(1)	2(1)	1	2(2)	1(1)	7(5)
P5	2(1)	2(1)	1(2)	3(2)	2	11(6)
인력수	7(3)	7(6)	6(4)	8(6)	6(3)	34(22)

(*): *은 지원받은 개발인력 수

팀은 프로젝트 소속팀과 기능적으로 지원하는 기능팀으로 구성되며 각 프로젝트 개발인력의 구성은 <표 5>와 같이 프로젝트의 각 기능에 해당하는 소속팀과 기능팀의 개발인력이 할당될 수 있다.

소속팀의 작업능률과 기능팀의 다르다고 가정하며 개발자의 능력에 따라 지원할 수 있는 팀과, 지원할 수 없는 팀을 구분하고 있다 <표 6>.

각 프로젝트는 납기일 이내에 완료하여야 하며 각 프로젝트의 처리시간(processing time)을 고려하여 인력배치와 일정계획을 수립하여야 한다. 각 프로젝트의 기능별 Iteration에 해당하는 처리시간은 <표 7>과 같이 주어져야 한다. 각 기능의 처리시간의 합은 프로젝트의 처리시간이다.

본 연구에서는 각 기능에 개발인력을 적절히 배치하여 전체 프로젝트의 총 처리시간(make span)을 최소화하고자 한다.

Revised PERT 차트를 참조하여 각 Iteration구간의 총 처리시간을 <표 7>과 같이 나타낼 수 있다.

<표 6> 인력 평가표 및 팀별 자원의 구성

개발인력	소속팀	지원가능기능	업무능력	개발인력	소속팀	지원가능기능	업무능력
MS1	MS	DS,CS,SS	상	CS4	CS	MS,AS	중
MS2	MS	CS,AS	중	CS5	CS	MS,DS	하
MS3	MS	DS,AS,SS	중	CS6	CS	DS,AS,SS	중
MS4	MS	CS	하	AS1	AS	SS	하
MS5	MS	CS,SS	중	AS2	AS	MS,DS,CS	중
MS6	MS	DS,CS	하	AS3	AS	MS,SS	하
MS7	MS	CS,AS	하	AS4	AS	DS	하
DS1	DS	MS,SS	중	AS5	AS	DS,CS,SS	중
DS2	DS	CS,AS,SS	상	AS6	AS	MS,SS	중
DS3	DS	MS,CS	중	AS7	AS	DS,CS,SS	중
DS4	DS	CS,AS	중	AS8	AS	MS,DS,CS,SS	상
DS5	DS	MS,SS	하	SS1	SS	MS	하
DS6	DS	CS,SS	하	SS2	SS	DS,CS	중
DS7	DS	CS,AS,SS	상	SS3	SS	MS,AS	중
CS1	CS	MS,AS	하	SS4	SS	CS	하
CS2	CS	DS,AS,SS	중	SS5	SS	MS,DS,AS	중
CS3	CS	MS,AS	하	SS6	SS	CS,AS	하

<표 7> 프로젝트별 기능의 구성

프로젝트	기능	Iterations				총처리시간	납기 마감일
		1	2	3	4		
P1	MS	17.5	17.5	0	0	35	248
	DS	23	23	0	0	46	
	CS	35	35	0	0	70	
	AS	23.5	23.5	0	0	47	
	SS	25	25	0	0	50	
P2	MS	26	26	0	0	52	260
	DS	26.5	26.5	0	0	53	
	CS	27.5	27.5	0	0	55	
	AS	20.5	20.5	0	0	41	
	SS	29.5	29.5	0	0	59	
P3	MS	17.5	17.5	0	0	35	208
	DS	21	21	0	0	42	
	CS	23	23	0	0	46	
	AS	22	22	0	0	44	
	SS	20.5	20.5	0	0	41	
P4	MS	24	18	18	0	60	280
	DS	23	17.5	17.5	0	58	
	CS	23	17.5	17.5	0	58	
	AS	23	17.5	17.5	0	58	
	SS	18	14	14	0	46	
P5	MS	15	15	11	11	52	329
	DS	16	16	12	12	56	
	CS	19	19	13	13	64	
	AS	23	23	16.5	16.5	79	
	SS	23	23	16	16	78	

4. 유전자 알고리즘의 적용

각 프로젝트는 모델링 단계, 설계 단계, 컴포넌트 단계, 아키텍처 단계, 솔루션 검증 단계로 진행되고, 각 기능에는 개발인력이 배치되어 염색체를 구성한다. 생성된 개체군은 하나의 염색체로 표현하고 적합도의 평가를 통해 가용한 해를 얻는다. <그림 6>은 염색체(Chromosome) 정의이며 프로젝트의 기능과 인력을 개체군으로 구성하여 염색체를 표현한다. 인력은 해당 기능팀의 인력과 다른 기능팀의 인력을 포함한다.

초기해(Initial Population) 생성은 유전자 정보를 기본으로 하여 프로젝트별 기능에 인력을 적절히 배치하여 가용한 해를 생성한다. 생성방법은 기능을 수행할 수 있는 기능팀 인력과 기능팀을 지원할 수 있는 지원인력을 무작위 선택하여 개발인력의 처리시간 합이 기능의 처리시간을 만족하도록 한다. 다른 프로젝트에 배치된 개발인력은 작업 완료시간을 처리시간 계산 시 반영하고 작업시간이 계산된 자원을 프로젝트별 각 기능에 배치하고 Iteration 단위로 진행하면서 프로젝트 납기일을 체크한다.

프로젝트	프로젝트1					프로젝트2				
기능	MS	DS	CS	AS	SS	MS	DS	CS	AS	SS
인력	MS1	DS1 MS3	CS2 AS2	AS1	SS1	MS3	DS4 CS2	CS1 MS1 MS4	AS2	SS3 DS5

<그림 6> 염색체 정의

- 교배(Crossover) 연산

본 연구에서는 두 교배점을 정하고 교배점 사이의 프로젝트의 자원을 교환하는 부분일치 교배(partially matched crossover : PMX) 방식[21]으로 구간을 일치시켜 교배를 한 후 그 외 나머지 부분의 결과에 의해 중복을 피하여 조정하는 연산을 한다.

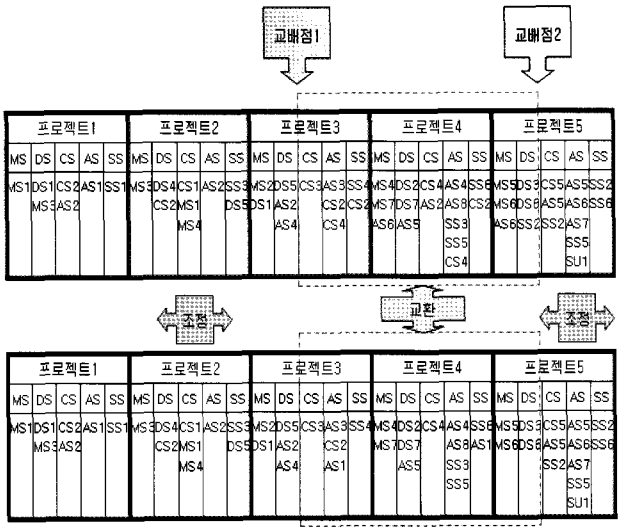
균등분포를 이루는 [0, n]의 범위에서 교배점을 무작위로 선택하고 두 교배점 사이가 교배구간으로 실제로 교배가 이루어지는 부분이다. 교배 전 염색체를 표현하면 <그림 7>과 같이 나타낼 수 있고 교배점 사이 구간의 자원을 교환하여 새로운 개체군을 생성한다.

교배연산은 두 개체군에서 프로젝트 자원의 일부분을 프로젝트의 기능을 기준으로 교배점을 생성하고 개체군을 교환할 때 선행 기능팀에 있는 개체군을 다른 프로젝트 구간에서 이동한다.

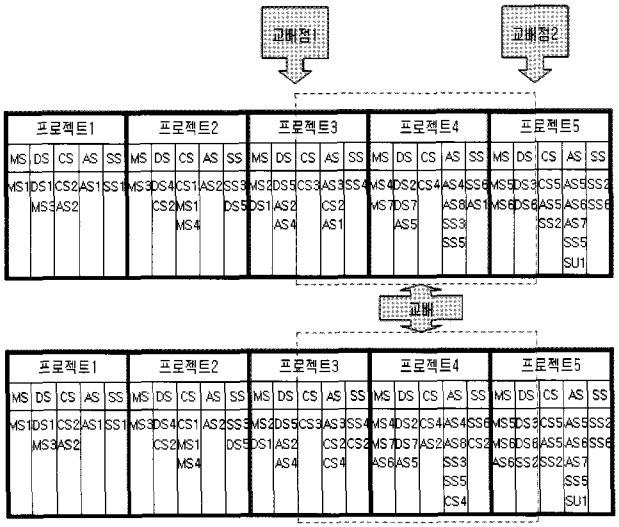
교배구간의 교환 시 같은 개체군 종류의 기능 팀끼리 교환하고 선행 기능팀에 있었던 개체 즉 인력을 교환하므로 요구사항의 파악이 용이하여 시간이 단축되며 교배 후 교배구간 밖의 재조정을 줄일 수 있다.

전체 프로젝트의 처리시간이 점진적으로 줄이기 위한 교배연산 과정은 다음과 같다.

BEGIN
  교배점의 무작위 선택
  선택된 프로젝트 내의 기능 교배점의 무작위 선택
  중복 개체군의 제거, 부족 개체군 충원
  교배구간 내의 개체군 교환 (swapping)
  교배구간 내의 개체군 처리시간 계산
  교배구간 밖의 개체군 조정
END



<그림 7> 교배 전 개체군



<그림 8> 교배 후 개체군

교배구간 밖의 개발인력을 교배구간 내의 개발인력과 중복을 피하기 위하여 조정배치가 필요하다. 교배구간 밖의 중복도 다른 프로젝트와 기능들도 같은 방식으로 재조정한다.

- 돌연변이(Mutation)

본 연구에서는 유전자 알고리즘의 돌연변이를 객체지향 매트릭스 조직 스케줄링의 특성에 따라 처리시간이 짧은 프로젝트의 개발인력을 임계경로(Critical Path; CP)에 해당하는 프로젝트의 기능을 지원하도록 하여 총 처리시간을 단축시킨다. CP는 프로젝트의 전체 일정에 가장 영향을 크게 미치는 주요 관리대상의 경로이다.

CP 프로젝트의 기능 중 처리시간이 다른 프로젝트들의 평균 처리시간 보다 큰 기능을 찾아내어 빠른 프로젝트에서 추출한 자원을 배치하도록 한다. 돌연변이 후에 CP 프로젝트를 재검색 하여 다시 돌연변이를 적용하였고 만약 CP 프로젝트의 처리시간이 돌연변이 전 CP 프로젝트의 처리시간 보다 길어지는 회수가 일정수 이상이면 돌연변이 연산을 중지하였다. 기본적인 연산과정은 다음과 같다.

Pset: CP를 구성하는 프로젝트들의 집합
pmax: 처리시간이 가장 긴 CP프로젝트
t(pmax): pmax의 처리시간
s(pmax): pmax의 마지막 기능
t(Pi): 프로젝트 i 의 처리시간
t(sj): 기능j의 평균 처리시간
t(bpmax): 돌연변이 전 pmax 처리시간
t(apmax): 돌연변이 후 pmax 처리시간
mk: 개발인력 k
min_t(Pi): 처리시간 빠른 프로젝트
min(mk): min_t(Pi)에서 참여율이 가장 낮은 인력

```
BEGIN
   COMPUTE t(Pi) for i = 1, 2, ..
   COMPUTE t(sj) for j = 1, 2, ..
   DO
      SEARCH Pset
      SEARCH min_t(Pi)
      t(bpmax) ← t(pmax)
      min(mk) ← s(pmax)
      m*k = min(mk),
      DELETE m*k from min_t(Pi)
      INSERT m*k  into t(pmax)
      COMPUTE t(Pi)
      t(apmax) ← t(pmax)
    WHILE ( t(apmax) < t(bpmax) )
END
```

가장 먼저 완료한 프로젝트의 기능인력을 CP 프로젝트의 기능에 추가하고 연산 후에는 CP 프로젝트가 바뀔 수 있으므로 처리시간을 재계산하여 새로운 CP 프로젝트를 찾아내어 연산을 반복하게 한다. 돌연변이 연산의 전 후를 비교하여 표현하면 <그림 9>와 같이 표현된다.

- 적합도 평가(Fitness Check)와 종결

유전연산 적용 후 생성되는 개체군은 프로젝트 납기일을 준수하여야 한다. 유전연산 후 개체군의 처리시간을 비교하여 처리시간이 짧을수록 적합도가 높은 것으로 판단하고, 적합도가 높은 엘리트 개체군을 보존하여

유전연산 수행 시 생성되는 개체군과 비교한다.

생성된 객체군의 적합도가 엘리트 집단의 적합도 보다 높은 경우 엘리트 집단의 개체군과 교환하여 최적의 엘리트 개체군을 보존한다. 적합도 평가 과정을 표현하면 다음과 같다.

Step 1. 프로젝트 납기일 준수여부를 확인

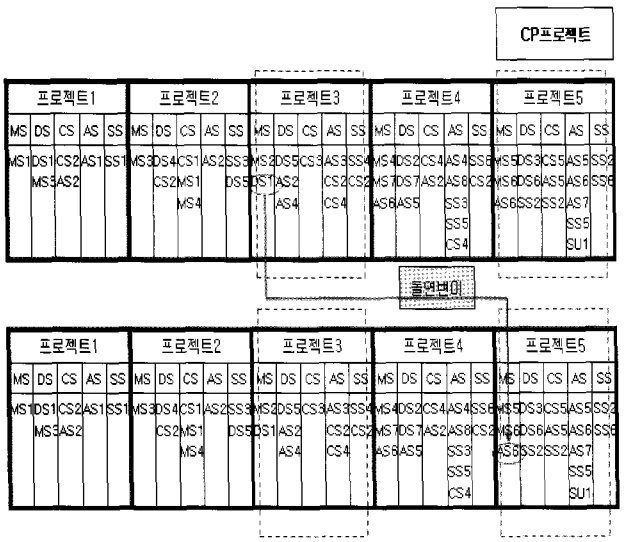
Step 2. 유전연산 후 개체군의 수행기간을 계산

Step 3. 엘리트 집단 및 유전연산 후 개체군의 수행기간과 비교

Step 4. 새로운 개체군이 기존 개체군보다 우수하면 엘리트 집단에 삽입, 가장 열성의 개체군 제거

교배연산과 돌연변이의 연산과정을 객체지향 모델링에서 정의된 Revised PERT 차트에 Iteration의 수만큼 설정된 반복수 내에서 수행하도록 하여 최적의 해를 찾도록 한다.

유전연산의 종결은 교배연산과 돌연변이 연산을 사용자의 정한 반복수 내에서 반복 수행하여 최적의 해를 찾게 된다.



<그림 9> 돌연변이 연산

5. 연구결과 및 분석

논문의 알고리즘은 Intel(R) Pentium(R)4 CPU 2.40 2.40GHz CPU 2.4 GHz AT/AT COMPATIBLE 261,616KB RAM의 컴퓨터에서 Java언어로 구현하여 테스트 하였다.

약 20가지의 문제를 무작위로 생성하여 테스트를 수행하였고 프로젝트의 크기는 테스트 환경의 문제로 100여개 정도로 제한하였다. 각 프로젝트별로 개발인력의 구성을 프로젝트 수에 비례하게 생성하고 프로젝트의 기능별 처리시간은 균등분포에 의한 구간 [40, 60]에서 무작위 추출하여 생성하였다. 소속팀 개발인력의 처리시간은 균등분포에 의한 구간 [6, 9]에서 생성하였고 지원팀 개발인력의 처리시간은 소속팀의 개발인력보다는 프로젝트 수행 시간이 더 오래 걸린다는 것으로 판단하고, 균등분포 구간을 [4, 5]에서 생성하였다.

<표 8> 개발인력의 주 업무와 지원 가능 업무

자원	소속팀		지원 기능팀 1		지원 기능팀 2		지원 기능팀 3		지원 기능팀 4	
	소속팀	시간	지원팀	시간	지원팀	시간	지원팀	시간	지원팀	시간
MS1	MS	8	DS	9	CS	9			SS	9
MS2	MS	6			CS	7	AS	8		
MS3	MS	7	DS	8			AS	9	SS	8
MS4	MS	10			CS	11				
MS5	MS	8			CS	9			SS	9
MS6	MS	12	DS	13	CS	14				
MS7	MS	10			CS	12	AS	13		
DS1	DS	10	MS	11					SS	13
DS2	DS	11			CS	13	AS	12	SS	14
DS3	DS	11	MS	15	CS	13				
DS4	DS	13			CS	12	AS	16		
DS5	DS	8	MS	12					SS	10
DS6	DS	9			CS	10			SS	11
DS7	DS	8			CS	11	AS	14	SS	10
CS1	CS	9	MS	10			AS	13		
CS2	CS	8			DS	9	AS	11	SS	9
CS3	CS	8	MS	9			AS	10		
CS4	CS	10	MS	11			AS	12		
CS5	CS	11	MS	9	DS	11				
CS6	CS	8			DS	9	AS	9	SS	10
AS1	AS	10							SS	11
AS2	AS	11	MS	12	DS	12	CS	13		
AS3	AS	13	MS	14					SS	14
AS4	AS	9			DS	11				
AS5	AS	8			DS	9	CS	10	SS	15
AS6	AS	10	MS	11					SS	15
AS7	AS	9			DS	12	CS	12	SS	15
AS8	AS	7	MS	9	DS	8	CS	8	SS	9
SS1	SS	5	MS	6						
SS2	SS	9			DS	14	CS	10		
SS3	SS	8	MS	11					AS	15
SS4	SS	10					CS	12		
SS5	SS	11	MS	11	DS	13			AS	12
SS6	SS	10					CS	11	AS	14

팀별 개발인력의 생성 데이터는 <표 8>과 같다. 팀별 개발인력의 수는 해당 프로젝트의 수에서 기능팀 수의 배수 내의 균등분포에서 무작위로 생성하였다. 문제 유형별 프로젝트 수 및 팀별 인원구성은 기능팀과 같다. 엘리트 개체군의 해를 평가하기 위하여 best solution과 개체군과의 편차는 다음과 같다.

$$편차 = \sqrt{(best\ solution - 개체군처리시간)^2}$$

best solution = $\min_{i \in p}\{t_i,\ i = 1, 2, \cdots n\}$

t_i : 엘리트 i의 makespan

p : 유전연산 종료 후 개체군의 엘리트 집합

n : 개체군의 크기 (population size)

<표 9> 문제의 정보와 편차

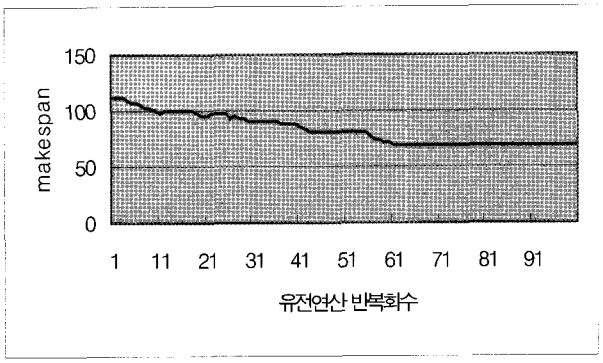
문제	프로젝트수	별 자원수					자원수	best solution	편차
		MS	DS	CS	AS	SS			
1	5	7	7	6	8	6	34	265	21.00
2	9	9	10	9	11	10	49	295	27.56
3	12	18	18	15	16	17	84	312	25.22
4	25	35	29	31	33	32	189	435	22.89
5	16	20	19	18	21	22	119	321	15.45
6	55	65	62	61	60	59	365	555	23.89
7	58	61	63	65	59	64	375	565	21.45
8	66	72	70	69	72	69	421	686	32.70
9	30	39	38	42	41	37	236	347	15.90
10	75	85	81	88	90	91	514	461	11.77
11	41	55	56	52	49	48	306	543	19.78
12	59	69	70	71	72	69	419	569	19.56
13	23	30	33	29	39	28	186	333	20.31
14	44	65	66	62	50	52	349	549	17.75
15	49	59	52	56	54	58	336	553	20.75
16	26	33	35	36	37	41	214	340	14.78
17	63	73	69	68	71	68	418	696	26.37
18	98	93	91	92	94	92	560	965	32.44
19	13	16	19	23	25	23	124	299	17.77
20	18	25	19	21	22	29	141	327	25.45

<표 9>와 같이 문제를 생성하여 유전 연산의 수행하여 결과를 얻었다. 유전연산의 반복수가 증가 될수록 프로젝트 처리시간이 단축됨을 확인 할 수 있었다. 반복의 수가 일정구간에 다다르면 시간이 더 이상 감소되지 않는 해를 best solution으로 간주하였다.

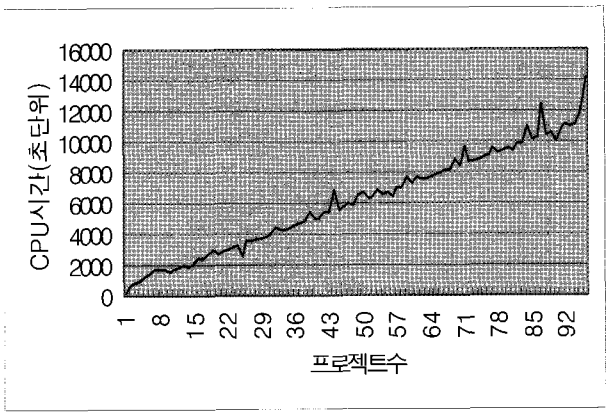
유전연산을 반복 할수록 총 처리시간이 짧은 엘리트 개체군이 생성되고 있다. 제한된 자원으로 정해진 납기일 내에 유휴인력을 적절히 배치하여 전체 프로젝트의 처리시간을 단축하는 스케줄링 얻을 수 있었고 유전연산의 반복수를 증가 시킬수록 best solution과 가용한 개체군의 해와의 편차도 감소되어 전반적으로 우수한 해를 얻을 수 있었다 <그림 10>.

하지만 어느 반복수 이상의 반복은 더 이상의 해의 개선이 이루어 지지 않음 알 수 있다. 유전연산의 반복을 지속하면 우수한 엘리트 객체군이 오랫동안 생성되어 분포되고 더욱 더 우수한 해를 생성할 있다는 점을 알 수 있다.

종결조건을 동등하게 했을 경우 프로젝트의 수가 증가하면 연산시간도 증가하고 있다 <그림 11>.



<그림 10> 유전 연산의 반복과 makespan



<그림 11> 프로젝트수와 연산시간

6. 결 론

객체지향 개발의 스케줄링은 복잡하고 어려워 프로젝트 매니저의 경험과 직감에 의존하여 왔다.

복잡한 다수의 프로젝트를 동시에 수행해야하는 상황에서는 인적자원의 활용에 한계가 있다. 다수의 프로젝트를 위한 매트릭스 조직에서 인력배치를 하면서 총 처리 시간을 최소화하는 문제는 문제의 크기가 작은 문제일지라도 최적해를 구하는 매우 어렵고 불가능한 경우도 있다.

특히 문제의 크기가 큰 문제의 경우는 유전자 알고

리즘을 이용하여 효과적으로 최적해에 가까운 해를 찾아갈 수 있을 것이다.

본 연구에서 제시한 객체지향 개발의 스케줄링의 방법을 성공적으로 적용하기 위해서는 개발자의 단계별 개발능력에 대한 객관적인 평가, 각 프로젝트 팀별 Iteration의 적절한 예측, 프로젝트 팀 간 혹은 단계별 기능팀 간의 인력공유에 대한 이해와 협조, 매트릭스 개발조직의 자원 효율화 및 생산성 측면에서의 올바른 인식 등이 선행 되어야 할 것이다.

7. 참 고 문 헌

[1] M. Hapke, A. Jaszkiewicz and R. Slowinski, "Fuzzy project scheduling system for software development", Fuzzy Sets and Systems 67 (1994) : 101-117

[2] S.-R. Kim, "Fuzzy PERT Applications for System Development Scheduling", 북악정보기술논집, 9 (2003) : 1-14

[3] 여한구, 이종태, "PERT/CPM에서의 프로젝트 완료시간 예측과 주경로 파악 및 통제를 위한 퍼지 기법의 응용", 산업기술논문집, 13 (1999) : 149-160

[4] 김정자, 공명달, "자원제약하의 복수 프로젝트 일정계획을 위한 휴리스틱 알고리즘", 대한산업공학회지, 13 (1987) : 110-119

[5] 이화기, 정제원, "Constraint Programming 을 이용한 자원제약 동적 다중프로젝트 일정계획" , 산업공학, 12 (1999) : 362-373

[6] 양미나, 이건호, "유전자 알고리즘을 이용한 매트릭스조직의 소프트웨어 개발 스케줄링", 경영과학, 23 (2006) : 187-198

[7] M. Qiu, "Prioritizing and scheduling road projects by genetic algorithm", Mathematics and Computers in Simulation, 43 (1997) : 569-574

[8] S.-S. Leu, An-Ting Chen and Chung-Huei Yang, "A GA-based fuzzy optimal model for construction time-cost trade-off", International Journal of Project Management, 19 (2001) : 47-58

[9] L. Ozdamar, "A Genetic Algorithm Approach to a General Category Project Scheduling Problem", IEEE Transactions. on Systems Man, and Cybernetics Part C: Applications and Reviws, 29 (1999) : 44-59

[10] K. Kim, Y. Yun, J. Yoon, M. Gen, G. Yamazaki, "Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling", Computers in Industry, 56 (2005) : 143-160

[11] M. Mori, C. C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem", European Journal of Operational Research, 100 (1997) : 134-141

[12] A. Azaron, C. Perkgoz, M. Sakawa, "A genetic algorithm approach for the time-cost trade-off in PERT networks", Applied Mathematics and Computation, 170 (2005) 761-780

[13] 박수진, "중소규모 프로젝트 적용을 위한 객체지향 소프트웨어 개발방법론의 테일러링에 관한 연구", 서강대학교 정보통신대학원, (1999)

[14] 정승렬, "객체지향 시스템 개발에서의 사용사례 활용실태에 관한 연구", 정보기술연구, 9 (2003)

[15] B. N. Joergensen, "Dynamic Scheduling of Object Invocations in Distributed Object-Oriented Real-Time Systems" Lecture Notes In Artificial Intelligence, 1543 (1998) : 503-506

[16] C. S. Chong, A. I. Sivakumar, R. Gay, "Design, Development and Application of an Object Oriented Simulation Toolkit for Real-Time Semiconductor Manufacturing Scheduling," In Proc. of the Winter Simulation Conference, (2002) : 1849-1856

[17] B. Wade "Mississippi Lignite Co. adopts Chronos scheduling software" Coal Age, 104, (1999) : 39-40

[18] A. R. Cockburn. "The impact of object-orientation on application development" IBM Systems Journal, 38 (1999) : 308-332

[19] J. H. Holland, "Adaptation in Natural and Artificial Systems", MIT press, (1992)

[20] B. Bruegge, A. Dutoit, "Object-Oriented Software Engineering : Conquering Complex and Changing Systems," Prentice Hall, (1999)

[21] R. Pooley, P. Stevens, "Using UML : Software Engineering with Objects and Components," Addison-Wesley, (2000)

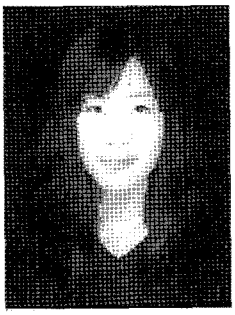
[22] M. Cantor, "Object-Oriented Project management with UML," Wiley, (1998)

[23] M. Fayad, W. Tsai and M. Fulghum, "Transition to object oriented software development", CACM, 39 (1996)

[24] 허진선, 최시원, 김수동, "Use-Case 기반 객체지향 프로젝트 스케줄링 기법," 정보과학회논문지: 소프트웨어 및 응용, 30 (2003) : 293-307

저 자 소 개

김 은 진

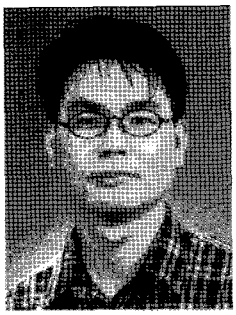


김은진은 2006. 02 년 숭실대학교 산업기술정보대학원 산업정보시스템공학과에서 석사학위를 받았다. 현재 우리금융 전산시스템 근무 중이다.

금융정보시스템, e-비지니스 시스템, 결제 시스템, 보안시스템 등의 요구분석, 설계, 개발, 품질관리, 개발관리를 해왔으며 관심분야는 객체지향 소프트웨어개발, 금융시스템 개발 프로젝트관리 등 이다.

주소: 서울시 동작구 상도5동 1-1 숭실대학교 산업정보시스템공학과

이 건 호



이 박사는 1996년 U. of Iowa, 에서 Industrial Eng.전공으로 박사학위를 받았다. 현재 숭실대학교 산업정보시스템공학과에 부교수로 재직 중이며 중앙정부 및 지방정부, 기업경영 자문과 국가공인자격증 시험 출제위원 등의 활동을 하고 있다. 소프트웨공학, 인공지능, 데이터 마이닝 등의 연구를 수행했으며 수십여 편의 연구논문을 국내 및 해외 유명저널에 소개하고 있다. 소개 페이지: http://software.ssu.ac.kr/ghlee.htm

주소: 서울시 동작구 상도5동 1-1 숭실대학교 산업정보시스템공학과