한국정밀공학회지 제 16 권 제 10 호 (1999 년 10 월)
Journal of the Korean Society of Precision Engineering, Vol. 16, No. 10, October 1999.

협동설계 효율화를 위한 설계순서작성 및 공유시스템 개발

한진택*, 이수홍**, 박삼진***

A Development of the Process Capturing and Sharing System for an Effective Collaborative Design

Jinteck Han*, Soo-Hong Lee**, and Sam-Jin Park***

ABSTRACT

This paper describes an approach to collaborative design which focuses on the effects of individual activities on the overall design process. We utilize a new process modeling tool to define the process and then analyze and refine the process based on critical paths. This information is then shared over the Internet with all participants. The goal of this system is to detect critical errors at initial design stage and guide the designers to make better decisions based on the knowledge of the overall process. This system enables participating designers to publish his local process through an Internet bulletin board. Other members of the team can then provide feedback based on how the proposed process impacts their activities. The system provides a context-rich, persistent forum for collecting, preserving, and refining corporate expertise of the team. For example, designers can select any process from the bulletin board and use it as a template for his current project and then use it to maintain his own design history. This paper is based on the process modeling concepts of Design Roadmap[2] and describe several key extensions in the area of CPM calculations and collaborative interfaces.

Key Words : Process Planning(공정계획), Process scheduling(일정계획), Design Roadmap[2](디자인로드맵이론) Collaborative Design(협동설계), Critical path method(취약경로산출법), WWW(월드와이드웹)

1. 서론

협동설계 환경에서 설계자간의 의사 소통과 아이디어의 보존 및 공유는 작업의 효율성과 직결되는 문제이다. 설계는 최종 결과가 나올 때까지 지속적으로 변경되기 마련이므로 각 설계자들이 빈번히 변하는 상황에 신속히 대처하면서 일단 기각되었던 아이디어가 다시 채택될 경우를 대비하여 아이디어를 효율적으로 보존, 공유할 수 있어야 하기 때문이다.

컴퓨터를 이용하여 이러한 요구를 지원하는 시스템은 설계자의 아이디어를 효과적으로 취합해야 하고 누구나 쉽게 설계자의 의도를 파악할 있도록 해야 한다. 나아가 단편적으로 취합된 설계안을 분석하여 다른 설계자에게 필요한 정보를 제공할 수 있어야 하며, 취합된 설계안이 다른 설계아이디어에 채택되는 경우 손쉽게 재사용할 수 있도록 해야 한다. 또한 초기의 설계안들과 새로운 정보들을 통합 관리하는 기능도 필요하다. 이를 정리하면 다음과 같다.

* 연세대학교 기계공학과 대학원
** 연세대학교 기계공학과
*** 선문대학교 공과대학 기계 및 제어공학부

- 설계안의 손쉬운 취합
- 설계자 모두가 이해할 수 있는 단순한 표현
- 설계안의 효율적인 분석도구
- 설계의 재사용과 참조 지원
- 전체적인 프로젝트를 관리하는 도구

과거의 경우와 달리 현재의 협동, 분산 설계 환경에서는 설계자들이 지리적으로 상당한 거리를 두고 있는 경우가 많다. PACT[3], SHARE[4] 등은 협동 작업 환경에서 의사 소통을 원활히 이룰 수 있도록 개발된 시스템들이다. 각 분산 시스템간의 작업 효율을 높이기 위해 에이전트 모델이 제시되었고, 이러한 에이전트 기반 작업에서의 의사 전달을 원활히 하기 위한 연구도 활발히 진행되어서 KQML[5], JATLite[10]와 같은 모델들이 제시되었다. 본 연구에서 개발한 시스템구성은 다음과 같다.

- 설계안작성-Macroscap[12]사의 IdeaStorm™
- 인터넷을 통해 설계안을 열람하고 분석하는 알고리즘을 사용하여 개발한 Java Applet.
- 인터넷을 통한 설계 의사 교환과 설계안의 열람/재사용이 가능하도록 데이터베이스와 연동되게 개발된 BBS

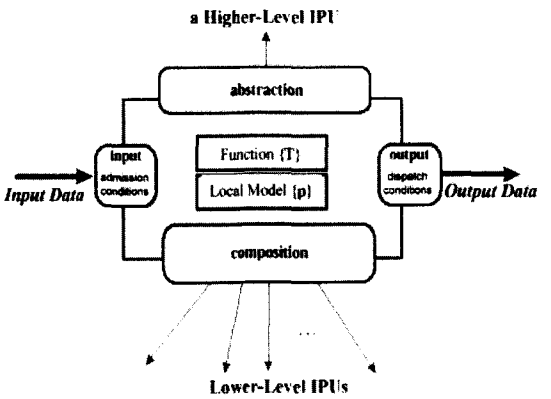

Fig. 1 Information Processing Unit(ICU)*[2]

설계안 작성 도구로서 사용한 IdeaStorm™ 은 협동 설계 Process 관리를 위하여 제시된 Design Roadmap[2] 이론을 사용하였다. 이 이론은 협동 설계 상황에서 흔히 발생하는 제약조건의 관리하며 간결하고 알기 쉬운 Process 표현법을 제시하였다

2. 본론

2.1 이론

2.1.1 Design Roadmap 이론의 모델 표현 요소

Design Roadmap 이론의 가장 기본적인 모델의 표현 요소의 단위는 Information Processing Unit(이하 IPU 라 칭함)이다. Fig. 1 에서 보는 바와 같이 IPU 는 정의된 작업을 처리하는 최소 단위의 표현 요소이다. IPU 는 자신의 기능에 대한 설명과 기능의 수행에 필요한 자원에 대한 요구 사항을 가지고 있으며 이전 IPU 와 다음 IPU 와의 연결을 위한 소켓과 상위 IPU 와 하위 IPU 와의 계층 구조의 정의를 위한 소켓을 가지고 있다. IPU 는 노드로 구체화되어 사용되며 연결(Link)을 이용하여 서로 연결되어 IPU 간의 관계를 형성하게 된다.

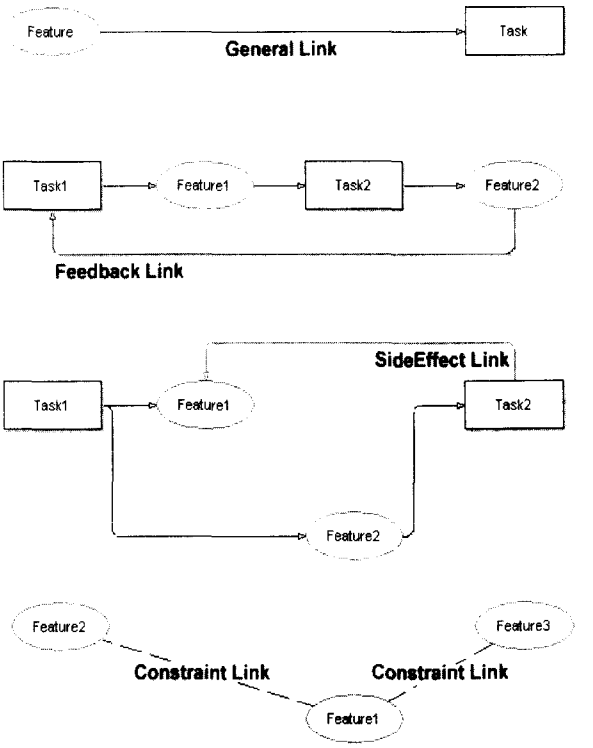

Fig. 2 Expression of Links*[2]

Fig. 2 에서 보는 바와 같이 노드는 크게 Feature 노드와 Task 노드로 나뉘며 Feature 노드는 Task 노드를 수행하기 위한 입력이나 Task 노드에서 출력된 결과를 표현하는 요소이며 Task 노드는 기능이 할당되어 있는 작업 수행을 의미하는 표현 요소가 된다. 다른 하나의 노드는 어떠한 Task 노드의 수행을 맡은 인력이나 도구를 의미하는 표현 요소인 Agent 노드이다. 연결(Link)은 일반적인 작업의 순서를 나타내는 표현 요소로의 기능을 가지고 있다. 부가적 기능을 가진 연결(Link)로는 Task 노드에서 작업의 수행 결과로 출력된 Feature 노드가 다시 Task 노드에 영향을 미치는 것을 의미하는 표현 요소인 피드백 연결(Link)과 Task 노드에서의 작업 결과가 Feature 노드에 영향을 미치는 것을 의미하는 표현 요소인 작업기능효과(Side Effect) 연결(Link)이 있다. 또한 Feature 노드 간의 제약 조건을 의미하는 표현 요소인 제약(Constraint)연결(Link)과 Task 노드를 수행하는 역할을 맡은 Agent 노드를 그 Task 노드와 연결하는 표현 요소인 할당(Assignment) 연결(Link)이 존재한다.

2.1.2 Design Roadmap 의 설계 과정 표현

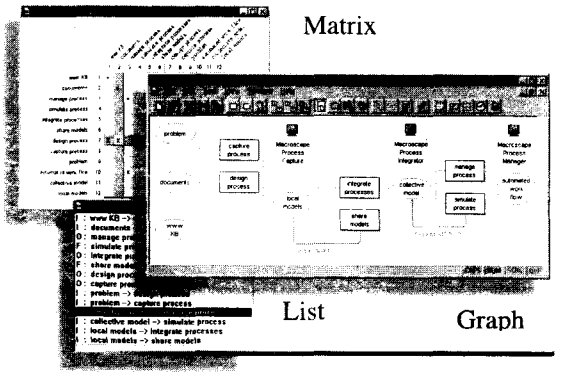

Fig. 3 Expression of design process

Design Roadmap 이론에서는 설계 과정을 표현하는 방법으로 기본적으로 그래프 보기, 리스트 보기, Matrix 보기를 제시하였고 각 보기에 대해서 원하는 표현 요소만을 보여줄 수 있도록 단순화된 보기를 추가로 제시하였다. 리스트 보기나 Matrix 보기는 각 설계 요소간의 내용이나 연관성을 알아보기에는 매우 바람직하였으나 본 연구에서 관심을 가지는 설계 순서의 결정을 위한 보기로는 적합하지 못하였다. List 보기는 사용되어진 표현 요소를 일목요연하게 보기에 유리하다. Matrix 보기는 사용된 각 요소들 사이에 연관 관계를 한 눈에 볼 수 있도록 도와준다. 본 연구에서는 2.1 절에서 논의한 모델의 표현 요소를 이용하여 설계의 순서가 명확히 판별되는 그래프 보기를 분석의 대상으로 하였다. 그래프 보기에서는 피드백과 작업기능효과(Side Effect)를 제외하고는 순환 경로를 형성하는 것을 금하고 있다. 각 표현 요소들은 그래프 보기 상태에서 손쉽게 제거되고 추가되고 다시 정의될 수 있도록 프로그램 될 것이며 그 결과 설계자에게 가장 친근하고 편리하며 전반적인 설계 과정을 이해하기 쉬운 보기로 여겨진다. 각 보기에 대한 표현의 예는 Fig. 3 에 나타나 있다.

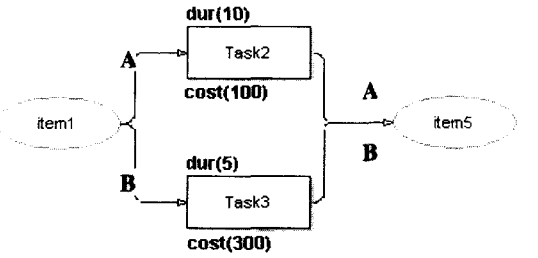

Fig. 4 Critical path(1)

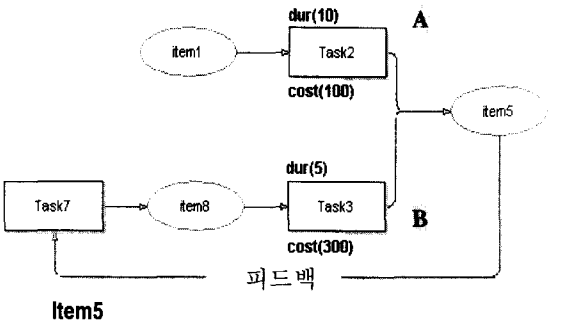

Fig. 5 Critical path(2)

2.1.3. 취약 경로(Critical Path)

취약경로(Critical Path[1])는 어떠한 작업을 하는 여러 가지 경로 중에서 정해진 하나의 자원에 대

하여 그 자원을 가장 많이 요구하는 경로이다. Fig. 4 에 나타난 바와 같이 경로 A 는 시간의 측면에서 가장 많은 투입 기간이 요구되는 경로이며 경로 B 의 경우는 비용의 측면에서 가장 많은 비용이 소모되는 경로이다. 따라서 시간의 측면에서 경로 A 상의 투입 시간이 현재보다 증가한다면 이것은 그 작업 전체의 완료 시간의 증가를 초래하게 될 것이다. 비용의 측면에서 경로 B 상의 투입 비용이 현재보다 증가한다면 이것은 그 작업 전체의 소요 비용의 증가를 가져오게 되는 결과를 가져온다. 취약경로는 같은 경로가 될 수도 있지만 일반적으로는 서로 다르다고 보았다. 이와 같이 취약 경로 위에 놓여진 작업들은 만일 그 작업이 요구하는 자원이 증가한다면 그가 속한 전체 작업의 자원 증가를 요구하게 됨으로 작업 관리자의 측면에서 가장 신경을 써서 집중적인 관리를 할 필요가 있는 작업 경로들이다. 본 연구에는 이러한 취약경로를 설계자가 구성한 작업 구상을 분석하여 설계 초기에 제시함으로써 설계자에게 이를 숙지 시켜 전체적인 작업 진행에 대한 이해를 도우며 참고 자료로서 활용하여 설계 작업 순서를 결정하는데 효율을 증대 시키고자 하였다.

일반적으로 그래프 표현에서 취약경로를 구하기 위해서는 그래프는 비 순환 그래프(Acyclic)라는 가정을 한다. 그러나 본 연구가 대상으로 삼는 Design Roadmap 이론에서는 전체적인 측면에서는 비순환 그래프이지만 부분적으로 피드백과 작업기능효과(Side Effect)라는 순환의 특성을 가지는 연결(Link)이 존재함으로 이에 대한 처리를 할 알고리즘이 필요하였다. 본 연구에서는 그러한 순환 마디를 감지하고 그 순환 마디를 순환 횟수에 비례하게 선형화 하여 비용이나 시간 값을 순환 횟수의 증가에 따라 비례하여 증가 시키는 방법으로 이를 해결하였다. 이것은 과거의 취약경로를 구하는 알고리즘에서 제외한 feedback 마디에 대해 순환의 최대횟수에 제약을 주어 의미상의 취약경로를 구하기 위함이다. 순환의 최대횟수는 열람을 하면서 동적으로 조정할 수 있으므로 취약 경로를 고려하는 사용자는 어느 정도까지 순환을 허용할 것인가에 대한 정보를 구할 수 있다.

동일한 작업 경로에 대하여 순환이 존재하지 않는 경우나 순환 횟수가 낮은 경우 취약경로가 아니었다가 순환이 발생하거나 그 횟수가 증가함에 따라 취약경로가 되는 경우가 발생하므로 이러

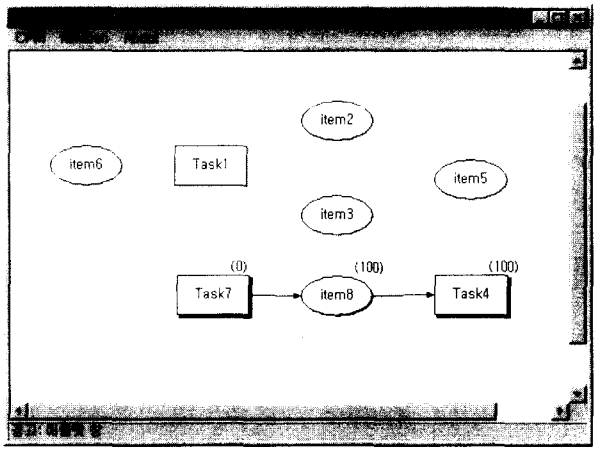

Fig. 6-1 Critical path – low iteration number

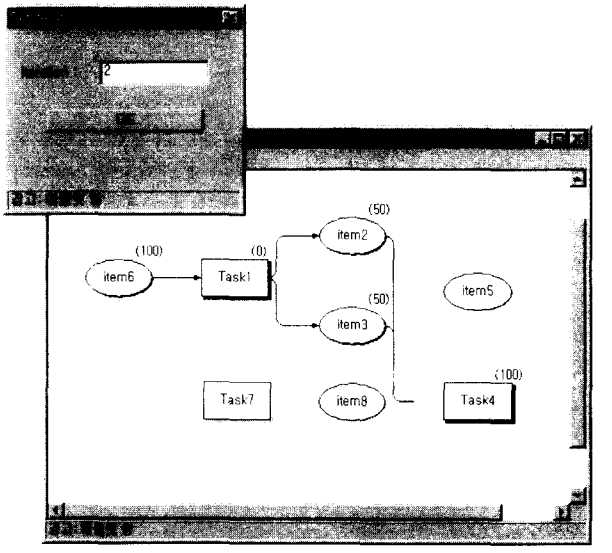

Fig. 6-2 Critical path – high iteration number

한 현상에 대비하는 알고리즘도 개발하였다. 이에 대해서는 Fig. 5, Fig. 6-1, Fig. 6-2 를 참고한다. Fig. 5 의 경우, 초기에는 윗쪽의 A 경로가 시간에 대한 취약경로 였으나 피드백에 의해 반복이 거듭되는 경우 경로 B 가 취약경로가 될 수가 있다. 이러한 경우의 처리를 위하여 피드백에 반복에 대한 제약을 둘 필요가 있다. Fig. 6-1 에서 Task4 로 오는 시간에 대한 취약경로는 Feedback 이 발생하지 않은 경우(가장 순조롭게 과정이 진행된 경우라고 가정)는 Task7-Item8-Task4 를 거치는 경로이다. 그

러나 Task4 에서 Item6 으로의 Feedback 이 발생한다면 시간이 50 일이 소요되는 Task1 을 2 번 지날 수 있는 가능성이 발생하게 된다.이 경우 Task1 을 지나는 경로와 Task7 을 지나는 경로는 모두 동일하게 100 의 시간이 요구된다고 생각한다. 만일 feedback 의 발생이 2 번이라면 Task1 은 총 3 번 거칠 수 있으므로 이 경우 취약경로는 아래의 그림 Fig. 6-2 같이 이동하게 된다.

경우에 따라서 설계자는 임의의 노드부터 시작하여 다른 임의의 노드로 구성되는 구간에 대하여서만 취약경로를 구할 필요가 있다. 이를 위하여 시스템에서 정의된 구간을 별도로 인식하여 그 구간에 한정된 취약경로를 산출할 알고리즘이 필요하였다. 이것은 그래프의 순 추적과 역 추적을 반복하여 원하는 구간을 산정하고 그 구간에 대해서만 결과를 산출하는 알고리즘을 적용하여 해결하였다.

2.1.4. 설계 계층 관리

복잡한 계층을 이루는 설계 과정의 경우는 이를 통합한 상위 계층의 설계안에서 하위 계층의 설계안을 연결하여 다룰 수 있다. 이 경우 각각의 취약 경로는 독립적으로 구해지지만 계층을 분화하여 작업함으로서 작업자는 보다 효율적이고 체계적으로 전 과정을 정리하여 작업할 수 있다.

예를 들어 Fig. 7 과 같은 전체의 설계 과정에 하위 설계 과정이 연결되어 있는 경우를 생각하여 보자. "mk_Mechanical Part" 노드와 "mk_Electrical Part" 노드에 아이콘이 붙어있는데 이는 이곳에 추가된 무엇인가가 있다는 것을 의미한다. 추가된 요소는 일반적으로는 그 노드의 수행에 필요한 Application File 이지만 필요에 따라 이곳에 하부 설계 과정을 연결시켜 둘 수 있다. 만일 "mk_Mechanical Part" 노드를 더블 클릭하면 Fig. 7-1 그림과 같은 정보 창이 나타나게 되고 여기서 "Linked files" 클릭하여 활성화시킨 후 "View File" 을 클릭하면 하부 설계 과정이 표시되게 된다. 이 하부 설계안도 다시 그 하부 설계안을 가질 수 있다. Fig. 7-2 에서 보듯이 "Assemble Part" 노드는 그 하부 설계안을 가지고 있다. 즉 위에서 표시된 예제는 총 3 단계의 계층을 가지는 설계안의 다루고 있으며 각각에 대한 분석작업을 시행할 수 있다.

2.2 구현

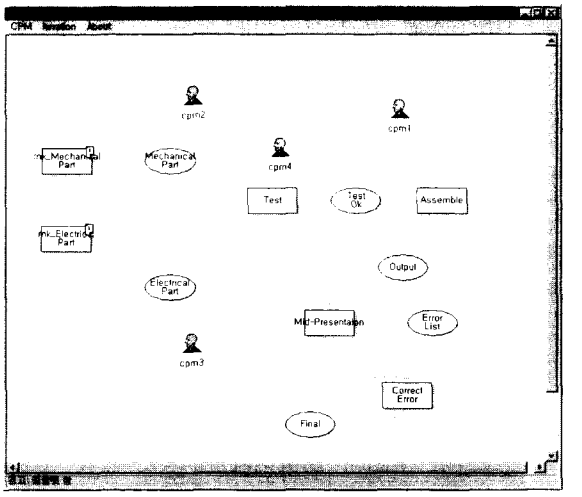

Fig. 7 Top-level design process

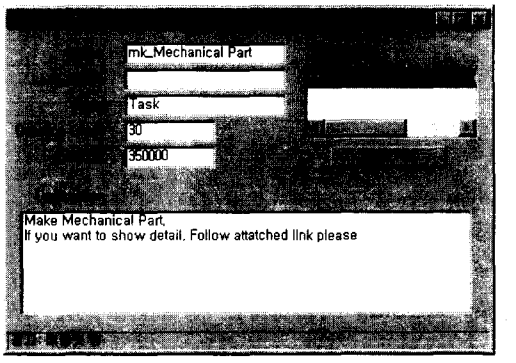

Fig. 7-1 Information window

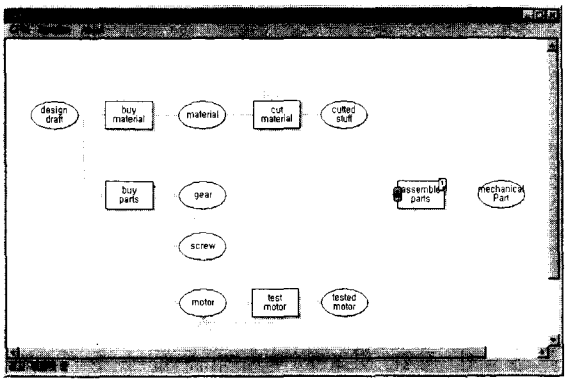

Fig. 7-2 Sub-level design process

2.2.1. 취약경로 산출 알고리즘 - 노드와 연결(Link)의 표현

하나의 노드는 독립된 하나의 객체로서 자신을 표현하는 모든 정보를 가지고 있다. 이 정보 모델의 기본은 Design Roadmap 의 IPU(Information Processing Unit) 모델이 기본을 두고 있다. 이 노드의 종류는 Task, Feature, 에이전트가 있으며 취약경로를 생성하는 데는 Task 와 Feature 노드만이 참가한다.

연결(Link)은 노드와 노드를 이어주는 역할을 하게 되며 일반적으로 작업의 순서를 지칭하는 의미를 가진다. 그러나 필요에 따라 해당 노드를 담당하는 에이전트를 연결하는 작업 할당 연결(Link)의 구실이나 Feature 간의 제약 조건을 관리하는 제약 연결(Link)의 구실을 하기도 하며 피드백, 작업기능 효과(Side Effect)의 표현을 나타내기도 한다. 취약 경로를 형성하는 데는 일반적으로 순서 지칭 연결을 사용하고 필요에 따라 확장되어 피드백, 작업기능 효과(Side Effect) 노드도 참가한다.

물리적 파일로부터 정보를 입력 받은 후 배열을 생성한 뒤 각 노드의 In-Degree(이하 IDE 라 칭함) 및 Out-Degree(이하 ODE 라 칭함)를 생성한다. IDE 란 그래프 보기에 있어서 어떠한 하나의 노드가 몇 개의 바로 전 노드를 가지고 있는 가를 나타내는 정수이며 ODE 는 반대로 하나의 노드가 몇 개의 바로 다음 노드를 가지고 있는 가를 나타내는 정수이다. 프로그램은 초기에 형성된 연결(Link) 배열을 읽어 들이고 이어서 전체 연결(Link)의 목록에 접근하여 연결(Link)의 시작 고유번호에 해당하는 노드의 IDE 를 하나씩 증가하고 연결(Link)의 끝 고유번호에 해당하는 노드에 ODE 를 하나씩 증가 시키는 방법을 이용하여 IDE 와 ODE 를 배당하게 된다.

그 다음은 각 노드에 대해 Earliest Event Time(이하 EET 라 칭함)을 부여한다. EET 란 그 노드가 가장 빨리 시작되는 경우 작업의 시작으로부터 얼마만큼 시간이 흐른 뒤에 해당 작업이 시작이 될 수 있는 가를 나타내는 수치이다.

EET 값을 정하는 경우 앞 노드의 작업이 끝난 시점이 자신의 노드가 시작되는 시점으로 본다. 일반적으로 EET(i)=0 인 최초의 노드가 존재하고 만일 어떤 노드 j 가 이전 노드 i 를 가지고 있다면 그 노드 j 의 EET 는 다음과 같은 식에 의해서 얻을 수 있다.

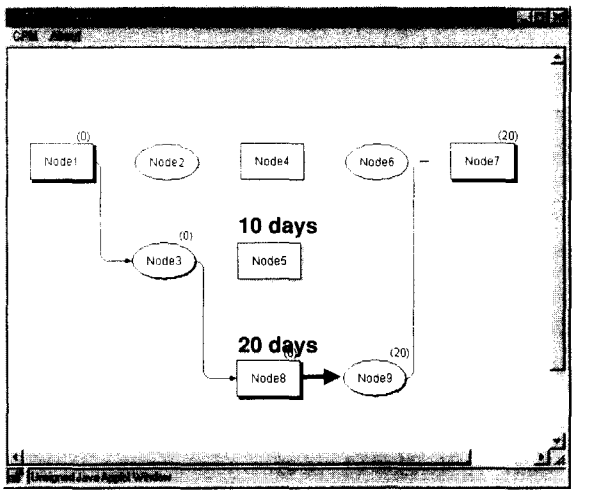

Fig. 8 Critical path – whole process

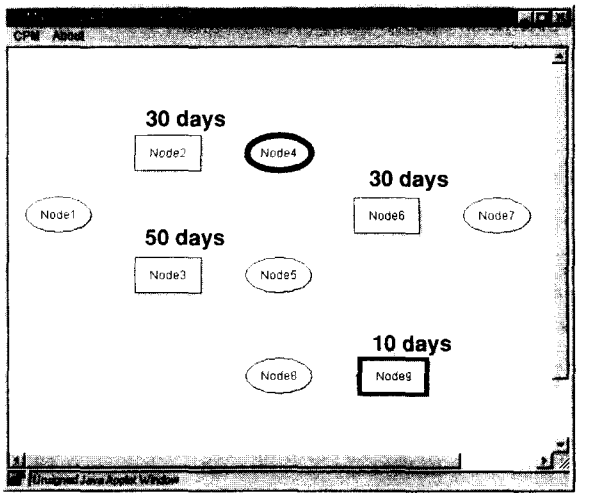

Fig. 9 Finding local critical path

Maximum value of EET(i) + duration of <i,j>

이 식을 이용하여 각 Node 에 대한 EET 의 계산을 마친 후 각 노드에 대해 Latest Event Time(이하 LET 라 칭함)을 부여한다. LET 는 그 노드가 작업이 시작된 후 최대한 얼마 이후에 시작될 수 있는 가를 나타내는 수치이다. 어떤 노드 i 가 다음 노드 j 를 가지고 있다면 그 노드의 LET 는 아래

의 식과 같이 정의한다. 이러한 계산은 그래프를 역추적하면서 이루어 지는데 그래프의 순 추적에서 사용한 방법을 동일하게 적용한다. 즉 검색이 끝난 노드의 ODE 를 하나 낮추고 ODE 가 -1 인 노드는 고려의 대상에서 제외한다.

Minimum value of LET(j) – duration of <i,j>

이러한 노드의 LET 를 결정하는 작업은 ODE 가 0 인 노드에서 시작한다. ODE 가 0 인 노드는 가장 마지막 작업 노드이므로 그 노드의 EET 는 LET 와 같다.

2.2.2. 취약 경로의 생성

앞 절에서 기술된 과정을 거쳐 모든 노드에 대해서 IDE, ODE, EET 와 LET 를 부여하는 계산이 끝나면 각 노드에 대해 아래와 같이 정의된 SLACK 수치들을 구해나간다.

SLACK (i) = LET(i) –EET(i)

취약 경로 위의 노드는 시간의 지연이 발생하면 안되는 노드들이므로 EET 와 LET 가 같다. 따라서 그들의 SLACK 값은 모두 0 를 가지게 된다. 이렇게 정의된 기본 알고리즘을 이용하여 비용에 대한 취약 경로나 각 부분 마디에 대해서도 취약 경로를 산출할 수 있다. 지역 모델이 아닌 전체 그래프의 노드에 대해서 취약 경로를 산출한 결과 (Fig. 8)가 일반적으로 사용되어 왔다. 지역 취약 경로(Local Critical Path)의 생성은 임의의 노드에서 또 하나의 임의의 노드로의 작업 경로 사이에서 발생하는 취약 경로를 찾는 것이 목적이다. 그러므로 검색을 시작하기 전에 선택한 2 개의 노드가 의미상 취약 경로를 구할 수 있는 경로를 형성하여야 한다. 아래에서 제시된 Fig. 9 에서 Node9 와 Node4 는 취약 경로를 찾기 위해 고려할 만한 직접적인 경로를 형성하지 않는다. 따라서 만일 이러한 2 개의 노드가 선택되어진다면 적절한 경고창을 보여주며 새로운 2 개의 노드를 선택하게 하여야 한다.

Fig. 10 에서 만일 Node9 과 Node3 사이의 경로를 형성하고자 하는 경우 표시된 바와 같이 Node9 를 끝으로 하고 IDE 가 0 인 Node1 까지의 가능한 전체 경로를 Node9 부터 역추적하여 설정

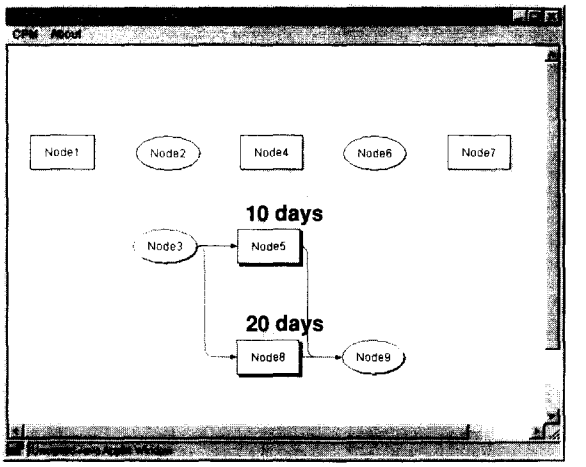

Fig. 10 Finding local path(1)

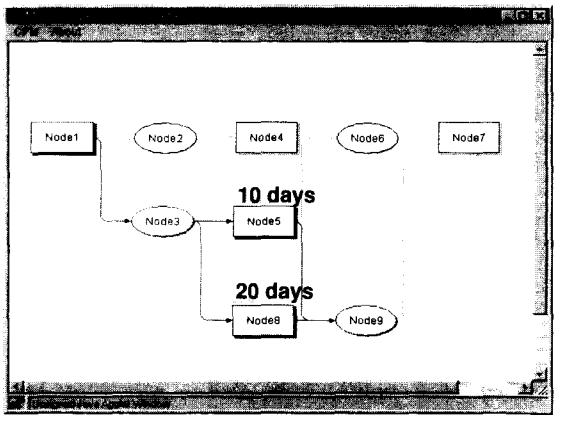

Fig. 11 Finding local critical path(2)

한다. 이후 Fig. 11 에 나타난 바와 같이 Node1 을 시작으로 하고 Node9 를 끝으로 하는 가능한 모든 경로를 형성하며 처음 과정에서 설정된 것과 중복되는 노드만을 고려의 대상으로 삼는다.

이러한 설정이 이루어지면 Node3 에서 Node9 사이의 작은 지역 모델이 하나 생성되고 이 지역 모델에 대해 앞에서 논의된 알고리즘을 사용하여 취약 경로를 구해나간다. Node3 에서 Node9 사이의 시간에 대한 취약 경로를 구한 결과는 다음의 Fig. 12 와 같다

2.2.3. 피드백과 작업기능효과(Side Effect) 연

결을 다루는 알고리즘

일반적으로 사용되는 순서를 결정하는 연결(Link) 이외에 본 연구에서 관심을 두어 해결하고 싶은 문제는 피드백과 작업 기능 효과(Side Effect)와 같이 순환 마디를 형성하는 연결(Link)을 처리하여 취약 경로를 형성하는 문제이다. 기존의 연구에서 제시된 취약 경로를 형성하는 알고리즘에는 이러한 순환 마디가 형성되는 경우를 배제하고 있으며 본 연구에서 설계안 캡쳐 도구로 사용하고 있는 Design Roadmap 이론이 적용된 응용 프로그램의 경우에도 이러한 순환 마디가 형성되는 경우에 위상 우선 순위(Topological ordering) 방법을 사용하여 그 순환 마디의 존재 여부를 감지할 뿐 이에 대한 분석 도구는 존재하지 않는다.

피드백이나 작업 기능 효과(Side Effect) 연결(Link)의 경우 이를 일반적으로 다루기는 상당히 까다롭다. 따라서 본 연구에서는 이러한 순환 마디가 형성되는 경우, 그 반복 횟수를 입력 받는 방법을 선택하였다. 즉 모든 일반적인 경우의 피드백 연결을 다루기에 앞서 반복의 횟수를 지정한 경우를 먼저 해결해보고자 하였다. 피드백이나 작업 기능 효과(Side Effect)와 같은 순환 마디를 형성하는 연결의 경우는 반복 횟수나 피드백 조건에 대한 최소한의 제약이 가해지지 않는다면 취약 경로가 가변적으로 움직이게 되므로 취약 경로 문제의 대상이 될 수 없다. Fig. 13 의 경우, Item1 부터 Task7 까지 비용에 대한 취약 경로를 설정할 때, 만일 Task7 에서 Item1 로의 피드백 연결의 존재를 고려하지 않는다면 Fig. 13 에 나타난 것과 같은 일반적인 취약 경로를 항상 형성할 것이다.

그러나 Task7 에서 Item1 로의 피드백의 횟수가 증가함에 따라서 취약 경로가 변화하는 경우가 발생할 수 있다. Fig. 13 의 경우, 가장 처음 노드인 Item1 노드에서 시작하여 비용에 대한 취약 경로를 Task7 까지 구하고자 할 때, 비용에 대한 내용을 가지고 있는 노드는 Item5($20) 과 Item6($10) 이 있다. 만일 피드백 연결의 존재를 고려하지 않는다면 취약 경로는 Fig. 13 와 같이 Item5 노드를 통과하여야 한다. 그러나 피드백이 발생함을 고려하는 경우에 그 반복 횟수를 한 번으로 제한한다면 최악의 경우 Item6 을 2 번 지날 수 있으므로 이 경우 Item5 와 Item6 은 모두 취약 경로 위에 위치하여야 한다. 만일 반복 횟수가 2 회 이상이

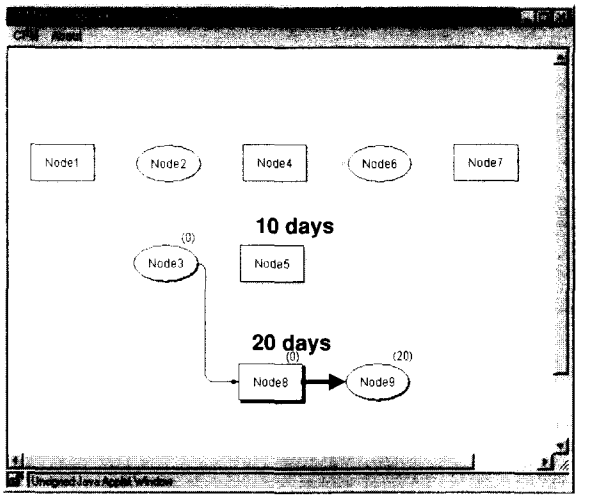

Fig. 12 Local critical path

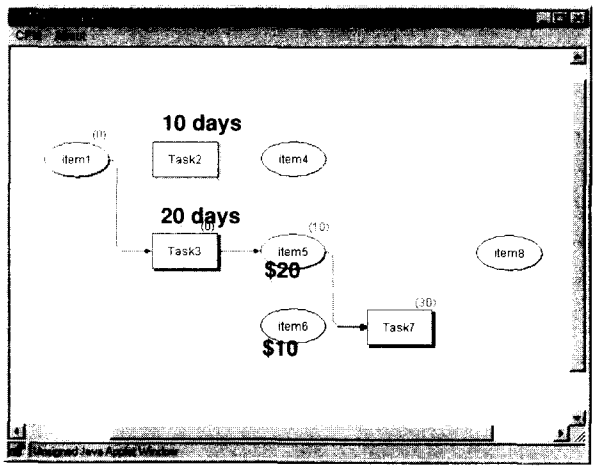

Fig. 13 Critical path of design process with feedback link. With no iteration

된다면 Item6 은 최소 3 번 이상 통과할 가능성을 가지고 있으므로 취약 경로는 Item6 을 통과하는 새로운 경로에 놓이게 된다.

위에서 논의된 문제를 처리하는 알고리즘은 앞에서 논의된 지역 취약 경로 형성에서 사용된 알고리즘을 사용하여 피드백이 시작되는 노드와 피드백이 끝나는 노드를 감지한다. 피드백의 시작과 끝을 알게 되면 그 사이에 있는 전체 노드에 대해 반복 횟수만큼 비용이나 지연시간 값을 증가시킨다. 이 경우 제시되는 각 노드의 비용이나 지연 시간은 최악의 경우를 고려하여 제시한 값이

된다. 따라서 같은 피드백 내에서도 경우에 따라 피드백이 발생할 수 있고 그렇지 않을 수도 있다. 하지만 본 연구는 사전에 최악의 경우를 제시하고 대상 작업의 반복 횟수가 증가함에 따라서 취약 경로를 형성할 수 있다는 것을 미리 알아볼 수 있는 기능을 제시, 전체 작업에 지장을 주지않고 반복할 수 있는 최대한의 반복 횟수를 제한함으로써 관리에 안전을 기울일 수 있는 방안을 제시하는데 의의를 두고 있다.

2.2.4. 설계공유

현재의 협동 설계 작업은 대개 그 업무를 분담하는 것에서 시작하여 자신의 작업 결과와 다른 그룹의 작업 결과를 취합하여 작업을 완성시켜나간다. 이 과정에서 작업 공정상에서의 자신의 위치를 명확히 인식해야 하는 문제가 발생한다. 더구나 설계자 자신은 일반적으로 1가지 설계작업에 전념하는 것이 아니라 2~3가지의 업무를 동시에 추진하게 되므로 작업의 우선 순위를 정하여 요구된 일정에 맞추어 작업을 진행시켜야 할 필요가 있다. 설계 작업을 관리하는 관리자의 입장에서는 자신이 관리하는 설계 작업들 중에서 전체 공정상 일정이나 비용이 초과되어서는 안될 작업 단위를 명확하게 인식하여 그 부분에 대한 집중적인 관리를 해야 할 필요가 있다.

협동 작업에서는 자신들의 작업 결과를 그 이력별로 관리, 공유하고 필요한 경우 제안을 둘 수 있는 장치가 필요하다. 만일 각 작업 그룹의 물리적 거리가 매우 먼 경우라면 그 필요성은 더욱 증가한다.

이상에서 설명된 필요성으로 개발된 알고리즘을 적용하여 설계 과정 취합 도구에 의해 모여진 결과에 대하여 WWW를 통한 설계 공유를 위하여 자바 언어를 이용한 애플릿을 제작하였다. 이 애플릿은 설계 게시판의 하나의 모듈로 작동하며 게시된 설계 과정과 그 과정에서의 출력물에 대해 WWW 상에서 즉각적인 열람이나 다운로드가 가능하다. 아울러 관리자의 권한을 가진 사람이라면 취약경로의 산출이 가능하다. 이러한 시스템을 통하여 설계자들은 자신의 노트나 회의실의 칠판에 설계 아이디어를 기록하는 대신 자신의 작업 컴퓨터에서 설계 아이디어를 메모하고 자신이 맡은 과정에서 산출된 결과물을 이를 게시판에 전송하여 둠으로써 다른 그룹 설계자들과 아이디어와 결과를 공유할 수 있게 되며 즉각적인 설계의 취약경로의 산출이 가능하므로 앞으로의 설계의 구현에 대하여 특별히 관리할 설계 과정에 대한 정보를 참고할 수 있게 된다. 아울러 관리자의 특별한 변경이 없는 한 이전의 설계 정보들이 그대로 보관되며 차후에 다시 재사용 할 수 있으므로 설계 이력을 관리할 수 있도록 해 준다.

2.3. 중소형 금형 설계 팀에의 적용사례*[11]

본 논문에서 제시된 시스템을 금형 설계를 담당하는 중소형 설계팀에 적용하여 보았다. 금형설계의 시작은 수주로부터 시작된다. 영업을 담당한 직원이 수주 자료를 작성하여 자신의 해당 작업 노드에 연결하여 두며 이 작업을 통하여 작업이 시작된다. 설계팀의 관리자는 설계에 필요한 공정의 개요를 정의하고 필요한 인원을 배정하고 각 인원이 수행해야 할 작업 노드에 연결하여둔다. 동시에 각 단위 노드에 필요한 작업 시간과 비용을 명시한다. 이러한 작업은 기존에는 표의 형식으로 문서 작성기를 이용하여 작성하였지만 개발된 시스템을 이용하여 작성하게 되고 작성자가 게시하는 경우 웹을 통한 공유가 이루어지게 된다. 또한 시스템을 작성하는 경우 공정의 흐름과 상호 연관 관계를 명확하게 알아볼 수 있고 후에 공정을 수정하거나 재사용할 수 있다. 설계자의 의사 교환은 주로 문서와 도면으로 이루어 지게 되며 여기서 가장 중요한 것은 도면이다. 도면 File만을 공유하게 된다면 그 도면이 어떠한 공정에서 만들어 진 것이며 이것이 다른 요소와 어떠한 연관이

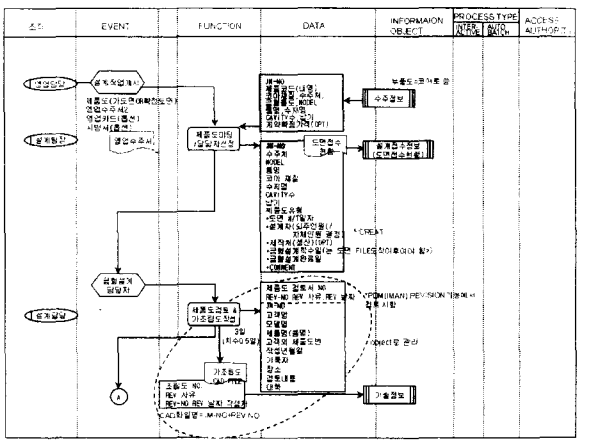

Fig. 14-1 Traditional expression of design process

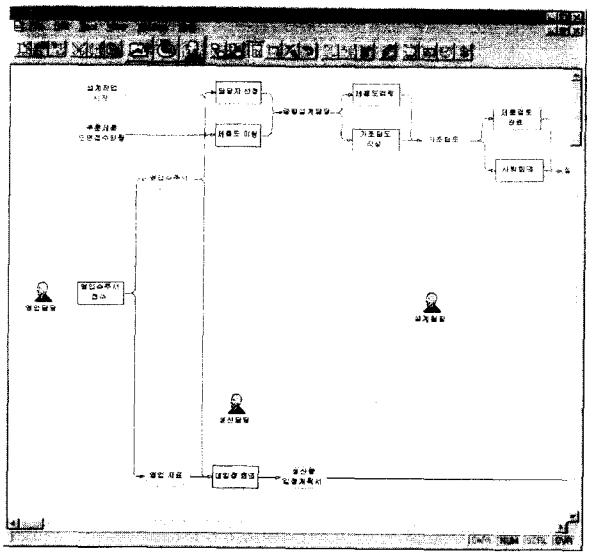

Fig. 14-2 Expression of design process in Fig. 16-1

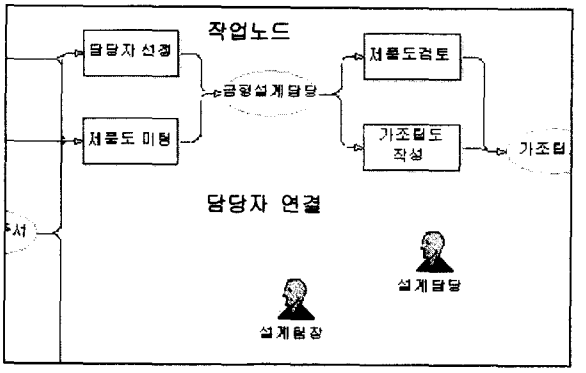

Fig. 14-3 : Node and assignment

나 제약을 가지는지에 대한 정보가 누락될 수 있다. 시스템을 사용하는 경우 공정의 특정 위치에 위치한 작업노드에서 도면을 열람할 수 있다. 또한 각 작업 노드에 대한 정보는 추가 정보의 형태로 일목요연하게 정리되어 열람할 수 있다. 작업을 진행하면서 과거의 설계 이력이 필요한 경우가 발생한다. 도면의 사소한 변경에도 그 이력은 시스템에 기록되어 있으므로 설계자는 도면에서 수정해야 하는 요소와 과거에 이루어졌던 유사 작업에 대한 정보를 손쉽게 얻어낼 수 있다. 이에 대하서는 Fig. 14-1 ~ Fig. 14-7 을 참고한다.

각 작업자들은 이러한 공정 레이아웃을 Web

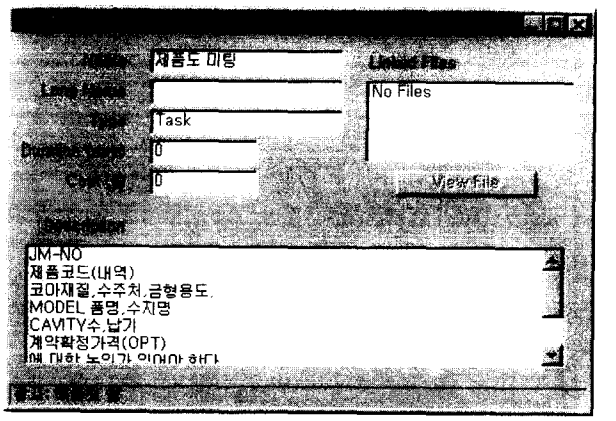

Fig. 14-4 Information window of node. The information includes node name, node type, processing time(duration), cost, detail description, linked files and etc.

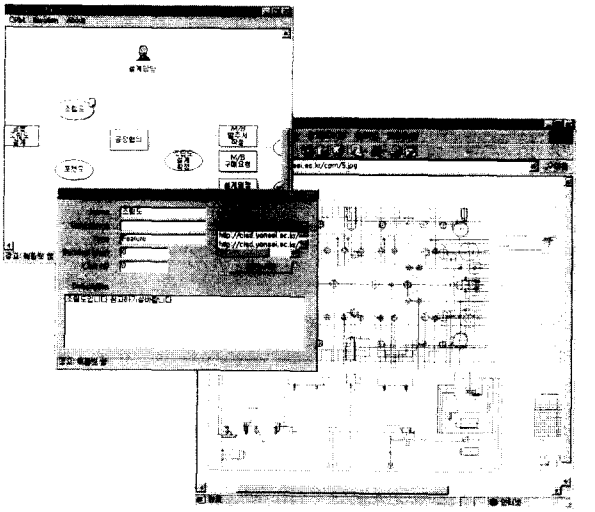

Fig. 14.5 Browsing linked files via Web. If designer click "View File" button on information window, they can view or download the linked file to their workstation.

을 통하여 검색하고 무리한 작업요소에 대한 수정안을 제출한다. 만일 이상이 없다면 작업을 속행한다. 이러한 작업은 대개의 경우 Web 에서 처리되며 일정이 확정된다면 설계 작업 관리자는 취약

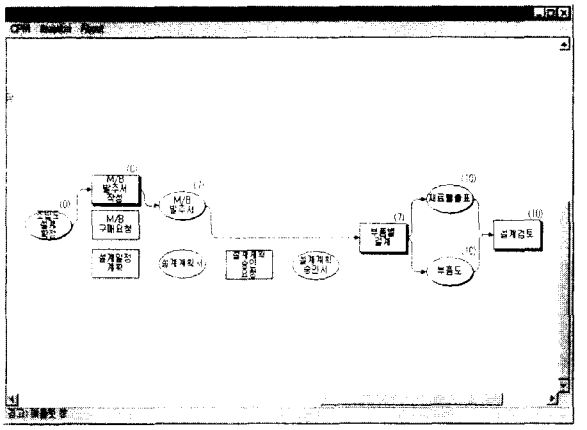

Fig. 14-6 Finding critical path on Web. Designer choose starting node, ending node and analyzing method. Java applet shows the critical path of the selected part of the process on Web.

Fig. 14-7 Web based design BBS system. This BBS has whole design history. Designer can browse citable design process of old projects. This system is connected with SQL database system

경로 산출 모듈을 가동하여 게시하고 해당 노드를 담당한 직원에게 통보한 후 작업을 진행하게 된다. 협동 설계 작업 전반에 대해 의사교환과 공정에서의 자신의 위치를 파악하면서 작업을 진행하므로 작업자 스스로 작업의 우선 순위를 파악할 수 있으며 비효율적인 회의로 작업 시간을 낭비하는 것을 방지할 수 있다.

3. 결론 및 향후 연구 과제

본 연구에서는 협동 설계에서 발생할 수 있는 비효율적인 요소를 해결하기 위한 개선 방안으로 공정 표현과 공유를 위한 도구와 분석을 수행하는데 요구되는 알고리즘 및 이를 통합한 시스템을 을 제시하였다.

적용 사례에서 제시한 바와 같이 기존의 공정 표현은 흐름도의 형태를 이용하였고 공정자료의 공유는 주로 회의를 통한 문서의 교환이었다. 흐름도에서는 각 구성 요소의 연관 관계나 그것들에 관련된 자료들의 연관성을 파악하는데 어려움이 있었다. 그러나 IdeaStrom 시스템으로 표현된 공정은 제약이나 연관 자료들을 손쉽게 검사하고 열람할 수 있었다. 작성된 공정자료를 본 연구에서 개발된 Web 을 통한 공유 시스템을 이용하여 공유함으로 협동작업을 하는 구성원 전체가 일괄적인 흐름을 인식하며 작업을 수행하였다. 이것은 전체적인 공정에 대한 이해를 향상시킬 수 있었다. 공정자료는 본 연구에서 제시된 시스템에 설치된 설계 이력 데이터베이스에 저장된다. 이것은 차후 발생하는 유사작업에서의 잘 정의된 작업 지침으로의 역할을 수행하였다. 시간적으로 중첩된 작업을 수행하는 작업자의 경우는 공유 시스템에서의 열람을 통해 전체 공정에서의 자신의 위치를 명확히 파악함으로써 작업의 우선 순위를 결정하는데 편리하였다. 본 연구에서 제시된 시스템은 IdeaStorm 과 설계 공유 시스템을 통합 운영하는 형태를 가지고 있으며 직접적인 설계 도구라고 볼 수는 없으나 이러한 설계 주변 상황을 조직화하여 설계 작업을 도울 수 있는 제반 환경을 제공한다.

본 연구에서 제시된 시스템에 사용되기 위해 개발된 알고리즘 중 순환 마디를 다루는 경우에는 최대 반복 횟수를 사용자가 입력해 나가면서 자료를 처리, 분석한다. 반복 횟수가 증가함에 따라 취약 경로의 변화를 알아보는 기능을 수행하며 취약 경로의 변화를 가져오는 반복 횟수 이상의 반복을 허용하지 않는다는 결정을 내리기 위함이다.

그러나 상황이 가변적으로 변하면서 설계 공정의 침삭이 계속적으로 발생하는 작업 상황에 대해 이러한 제한된 최대 반복 횟수를 사용해 공정을 분석하고 공정의 침삭이 발생한 경우 다시 반복횟수를 조정하며 작업을 검토하는 것은 경우에 따라서 작업의 효율성을 감소시킬 수 있다. 최대 반복 횟수가 정해지지 않은 순환 마디에 대한 분석은 일반적인 분석의 대상이 될 수 없으므로, 향후 최대 반복 횟수가 제한되는 경우에 작업 효율의 현저한 감소나 심각한 오류가 발생하는 경우에 대한 분류가 필요하고 각 분류에 대한 문제 해결 알고리즘을 보충할 필요가 있다. 또한 적용 대상을 확대하여 본 연구에서 제시된 시스템을 적용한 경우 협동 설계 작업의 효율성 증가에 대한 보다 구체적이고 객관적인 자료를 산출하여 제시할 필요가 있다.

참고문헌

1. Weiss, Mark Allen. "Data structure and algorithm analysis in C 2nd edition," Addison Wesley Longman, Inc. Chap. 9.3.4~9.3.4. 1997.
2. Park, Hisup. "Modeling of collaborative design process for agent-assisted product design," Ph.D thesis. Stanford Univ. Chap. 3.2, 4.1 March. 1995.
3. Cutkosky, M., Genesereth, M., et al., "PACT : An Experiment in Integrating Concurrent Engineering Systems," *IEEE Computer Special Issue on Computer Supported Concurrent Engineering.* Chap.2, Jan. 1993.
4. Toye, G.., Cutkosky, M., Leifer, Larry J. "SHARE: A Methodology and environment for collaborative product development" Introduction, Chap. "The Developing SHARE environment," *W.E.T.I.C.E.* April, 1993.
5. Finin, T., McKay, D. and Fritzson, R., "An Overview of KQML: A Knowledge Query and Manipulation Language," Technical Report, Computer Science Department. University of Maryland. Mar. 1992.
6. Peng, Y., Finin, T., Labrou, Y., Chu, B., Long, J., Tolone, W.J., Boughhannam, A. "A Multi-Agent system for enterprise integration," *The international conference on agile manufacturing.* Chap. 2, Chap 3.2 1997.
7. Schlenoff, C., Knutilla, Amy. Ray, Steven. "Requirements for modeling manufacturing process: a new perspective.". *ASME, DET97/CIE-4279.* Chap 4, Chap. 5. 1997.
8. Andrew, D. Christian, Kamala, J. Grasso, Warren, P. Seering. "Validation studies of an information-flow model of design," *AMSE, 96-DETC/DTM-1306.* Chap. "Modeling and Simulation" 1997.
9. Sohall, M. Qureshi, Jami J. Shah, Susan D. Urban. "Integration model to support archival of design history in databases,". *ASME, DETC97/DTM-3876,* Chap.2.1, Chap 2.2. 1997.
10. "JATLite", http://java.stanford.edu/.
11. http://cisd.yonsei.ac.kr/scripts/bbs-scripts/index_bbs.asp (로그인 사용자 : cpm1 ~ cpm4, 로그인 암호 : 각 사용자 이름과 동일.)
12. Macroscape, Inc., http://www.macroscape.net