The Journal of Korea
Institute of CALS/EC
Vol. 2, No. 1, June. 1997

분산 환경하의 설계 및 제조활동을 위한 프로세스관리기법 연구

박 화규*, 김 현*, 오 치재*, 정 문정**

A Process Management Framework for Design and Manufacturing Activities in a Distributed Environment

Hwa-Gyoo Park, Hyun Kim, Chi-Jae Oh, Moon-Jung Chung

Abstract

As the complexity in design and manufacturing activities of distributed virtual enterprises rapidly increases, the issue of process management becomes more critical to shorten the time-to-market, reduce the manufacturing cost and improve the product quality. This paper proposes a unified framework to manage design and manufacturing processes in a distributed environment. We present a methodology which utilizes process flow graphs to depict the hierarchical structure of workflows and process grammars to represent various design processes and design tools. To implement the proposed concept, we develop a process management system which mainly consists of a cockpit and manager programs, and we finally address a preliminary implementation procedure based on the Object Modeling Technique. Since the proposed framework can be a formal approach to the process management by providing formalism, parallelism, reusability, and flexibility, it can be effectively applied to further application domains of distributed virtual enterprises.

***Keyword**: Process Management, Cockpit Program, Manager Program, Non-Terminal Task Node, Process Flow Graph, Production, Specification Node, System Integrator.*

---

* 시스템공학연구소 (Systems Engineering Research Institute)

**Department of Computer Science, Michigan State University, USA

1. 서 론

1.1 개요

기업활동의 단위 시스템이 복잡해 짐에 따라 생산성 향상이 중요한 문제로 부각되고 있다. 엔지니어링 부문에 있어서도 CAD/CAM/CAE/CAPP 등에 관한 기법 및 전산 시스템의 발달로 제품의 설계, 개발 및 제조에 관련한 생산성은 급속도로 향상되고 있으나 이에 따라 산출되는 엔지니어링 Data 의 흐름이 Bottleneck 으로 작용하여 원격지의 타 부서와 Communication 이 이루어지지 못하고 있으며 작업흐름 관리가 원활하게 지원되지 못하고 있다. 이에 따라 Process 관리의 개혁 요구가 대두되고 있으며 이를 위한 솔루션으로 Process Management System 이 등장하게 되었다. 이전의 생산관리는 MRP (Manufacturing Resource Planning)와 제품관리 중심이었으나 90 년대 중반에 들어 오면서 Process Cycle Time 을 단축시키고자 하는 노력이 계속되고 있다. 이는 생산 현장 중심의 체계에서 제품 설계로 촛점이 옮아가고 있음을 의미하며, 설계/개발 기간의 단축을 위해서는 체계적인 Data 의 관리가 필요하다는 것을 인지하게 된 것이다.

1.2 관련 연구

현재 설계 Process 관리에 대한 활발한 연구가 진행 중에 있고 일부 상용 소프트웨어가 발표되고 있으나 아직은 더 많은 연구가 필요한 상태이다. 지금까지 설계 Process 관리에 관해 연구되고 있는 내용을 살펴보면 설계 지원 Tool 로서 Di Janni [Di Janni, 86]는 확장된 Petri Nets 를 이용해 고정된 (Fixed) Workflow 를 모델링하는 방법을 제시했고, VOV 시스템 [Casotto et al., 90]은 설계자가 Tool 을 실행할 때 그 Sequence 를 기록할 수 있게 했다. 이 시스템은 입력 파일이 수정되면 수정 이전 입력 파일의 무효화된 출력 파일 혹은 이전의 Tool 실행을 통해 Data 일관성 (Data Consistency)을 유지하게 하였다. 이 외에도 설계 Process 중에 실행될 Tool 을 선정하는 시스템으로 Forward Chaining 방법을 이용해 계획그래프 (Plan Graph)를 산출하는 ADAMS [Knapp, 92] 가 있고 설계 Process 계획산출을 위해 계층적 전략방법을 이용하는 Minerva [Jacome, 92]와 OCT Task Manager [Chiueh, 90]가 있다.

또한 설계 Process 의 지식 처리를 위해 유관 지식들 간의 협조나 상충성을 해결하기 위한 연구로써 Case-Based Reasoning, Agent-Based Approach 및 Blackboard Approach [Corkill et al., 87] 등의 연구가 활발히 이루어지고 있다. 일본의 경우도 최근 설계에서 Artificial Intelligence 기법의 연구와 설계방법론에 대한 연구가 활발해짐에 따라 1991 년에 일본 기계학회에서 설계 및 System 부문을 설치하여 컴퓨터를 이용한 설계 고도화에 관

한 연구를 수행 중에 있다.

이러한 많은 시스템들이 거 개발 혹은 진행 중에 있으나 전반적으로 다음과 같은 문제점을 내포하고 있다.

- Formalism의 부족: 전체 Workflow의 시스템 Behavior 예측이 어려우며 특정한 Property의 만족도 분석이 용이하지 않다.
- Process 표현력의 부족: 대부분의 기존 시스템은 정보의 완전한 기술을 전제로 하기 때문에 전체의 이해도가 낮아져 정보의 상호 공유가 어려우며 계속적인 Process 향상과 Sub-Process의 Dynamic Behavior 모델링이 어렵다.
- 설계 Data와 Process 간의 통합 기능 부족: Process는 여러 Format 형태의 Data를 산출하게 되는데 현존 시스템의 Framework들은 이를 위한 설계 Data와 Process 간의 관계 및 일관성 (Consistency) 유지가 어렵다.
- 비 유연성: Run Time시에 Process 흐름의 재 정의와 예외성 처리 (Exception Handling)가 어렵다. 그러나 대부분 Workflow 내에는 많은 예외성이 존재하고 Run Time시 흐름의 재 정의가 불가피하다.

1.3 제안 시스템

공학설계에 있어서, 주어진 설계 사양으로 부터 원하는 설계를 얻기 위해서는 실제 설계 해를 얻기 위한 공학 기술적 의사결정도 매우 중요하지만 어떠한 설계 과정을 통해서 효율적으로 설계를 진행시키는가 하는 설계 프로세스의 관리 역시 매우 중요하다. 특히 자동차, 항공기, 선박 등의 복잡한 설계를 진행하기 위해서는 설계 프로세스를 모델링하고 이를 통해 설계 과정에서 발생되는 Event를 관리하는 Process 관리 체계의 구축이 반드시 필요하다. 본 연구에서 제안된 Framework은 설계 활동에 공통적으로 적용 가능한 Process의 Formal Representation에 중점을 두었으며 이는 Process의 Behavior를 분석하고 예측을 가능하게 한다. 이를 위해 Process Management System에서는 설계 Process를 표현하기 위한 Process Grammar를 개발하고 관련 Data간의 일관성을 유지하면서 설계 Process에서 발생되는 Activity들의 관리 통제 기능을 개발하였다. 구체적인 개발 내용은 다음과 같다.

- Process Grammar: 설계 Process를 정의하기 위한 Grammar를 개발하고 이를 통해 설계의 계층적인 Decomposition 과정 및 설계 Data와 설계 업무 간의 종속성을 표현하였다.
- Process Browser: 설계 Process의 관리 및 통제를 위한 Browser를 개발하였다. 이는 Task Library, Production Library, Design Data Library 및 Process Library를 포함하며 각 Library는 Generalization/Specification 계층 구조로 구성하였다.
- Graphical Editor: Process를 편집하기 위한

Graphical Editor를 개발하였다. 이는 Event를 식별하거나 Process를 동적으로 편집하거나 또는 Process를 저장하기 위해 이용될 수 있다.

Figure 1은 기업 통합을 위한 Process 정보 Infrastructure를 나타낸다. 예시된 바와 같이 제시된 Framework은 설계 외에 타 부문으로 확장 가능하나 본 연구에서의 범위는 설계 부문을 위한 Process 관리의 적용에 한정한다.

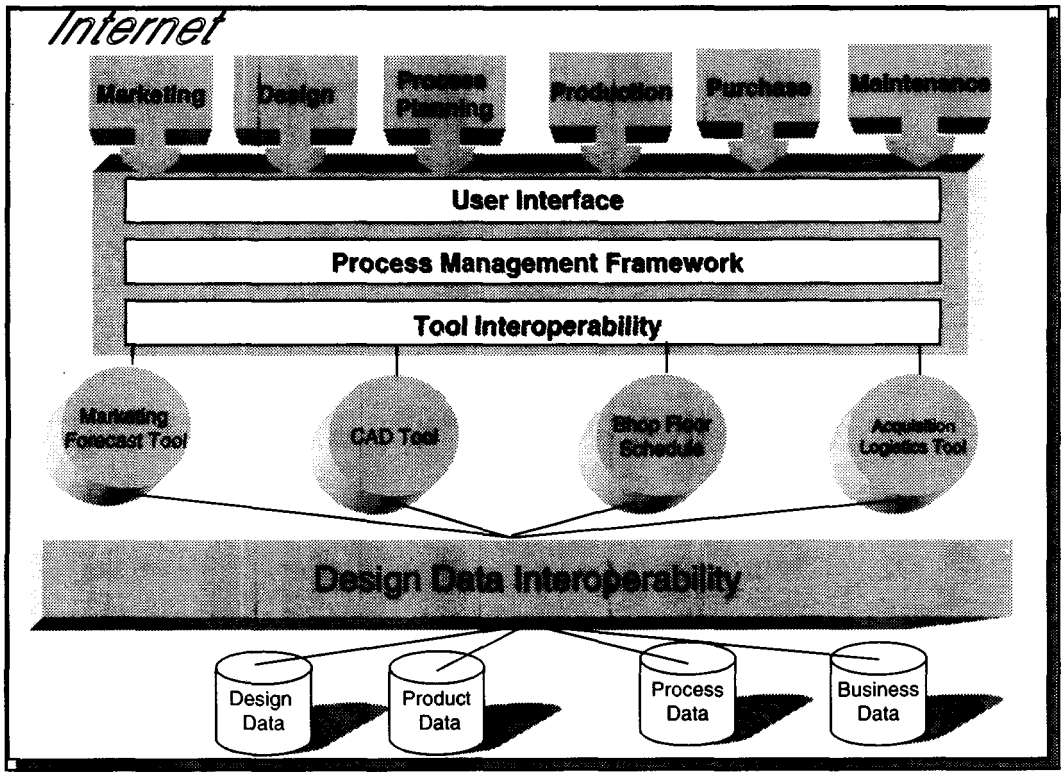

Figure 1. Information Infrastructure of Enterprise Integration

2. 방법론

본 제안 시스템은 Process Flow Graph와 설계 Process Grammar에 근거한 방법론을 사용한다. Process Flow Graph는 Process에서의 정보 흐름을 기술하며 설계 Process Grammar는 상위 Level의 Process Graph를 점차 세부적인 하위 Graph로 전환하는 수단이 된다.

2.1 프로세스 흐름 그래프 (Process Flow Graph)

Process Flow Graph는 Bipartite Acyclic Directed Graph로써 이 Graph는 두개의 Task Node와 Specification Node로 구성된다. 모든 Edge는 한 Type에서 다른 Type으로 연결하기위해 사용되며 자기 자신의 Node로 다시 유입되는 Edge는 존재하지 않는다. 또한 Task Node는 Terminal Node와 Non-Terminal Node로 분류된다. Terminal Node는 원격지 Tool 실행 (Invocation) Node로 Application Program을 실행하게 되며 이중 원형으로 표현된다. Non-Terminal Node는 소수개의 다른 Tool 조합으로 분할 되어지는 Abstract Task로서 단일 원으로 표현되어 진다. 각 Specification Node에서 Specification 업무가 표시되고 각 Specification Node는 한 개의 유입 연결부(Incoming Edge)를 갖는다. Incoming Edge를 갖지 않는 Specification Node는 Graph G에서의 초기 입력 값을 의미한다. T(G), S(G)와 E(G)는 각각 Graph G의 Task Node, Specification Node의 집합과 Edge를 표현한다.

Figure 2는 Rapid Prototyping을 위한 설계 Process를 표현한 것으로 Behavioral Description을 갖는 Specification Node가 Structural Description을 표현하는 Specification Node으로 변형되는 Process Flow Graph의 간단한 예를 나타낸다. Parent Node는 몇 개의 Child Node로 분할되면서 보다 구체화 된 세부 Specification으로 표현 된다.

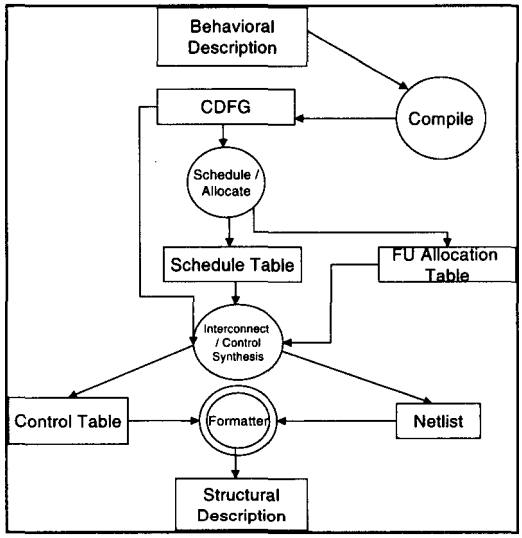

Figure 2. Sample Process Flow Graph for High Level Synthesis

Process Flow Graph는 다양한 내용의 상세 Level Process를 표현 할 수 있는데 Terminal Node가 어떠한 세부 Tool이 사용되어져야 하는가를 나타내는 구체적인 것이라면 Non-Terminal Node는 비 구체적인 상태로써 수행되야 하는 방법론만이 표현된다.

In(N) 이 Node N의 입력 Node 집합이면:

$In(N) = \{ M \mid (M, N) \in E \}$.

Out(N) 이 Node N의 출력 Node 집합이면:

$Out(N) = \{ M \mid (N, M) \in E \}$.

I(G)이 Graph G의 입력 Specification 집합이면:

$I(G) = \{N \in S(G) \mid In(N) = \emptyset\}$과 같이 표현 될 수 있다.

2.2 프로세스 문법 (Process Grammar)

설계자는 대략적인 흐름과 목표를 설정하기 위해 몇 개의 입 출력 Specification Node, Task Node 및 Edge 를 사용하여 Editor 상에서 상위 Level 의 초기 그래프(Initial Graph)를 설정하게 된다. 그러면 Process Grammar 에 의해 Non-Terminal Task Node 는 좀더 세부적인 Abstract Task 및 Intermediate Specification Node 들로 대치되며 출력 Specification Node 도 보다 자세한 Child Specification Type 을 갖는 Node 들로 전환된다.

Graph Grammar 의 Production 은 서로 Mapping 가능한 대안들의 집합으로 이를 통해 하나의 Sub-Graph 는 다른 세부 구성과 수순을 갖는 Sub-Graph 로 대치된다. 이러한 Production Rule 은 다음과 같은 Tuple 로 표현된다.

$P = (GLHS, GRHS, \sigma In, \sigma Out)$

여기서 GLHS 과 GRHS 과는 각각 좌측과 우측의 Process Flow Graph 를 나타낸다. 따라서 T(GLHS)은 변환 되어야 할 Single 의 Abstract Task 를 나타내며 Non-Terminal Node 이다. σIn 은 I(GRHS)에서 I(GLHS)의 Mapping 으로 입력 Specification 간의 정확하게 Match 되는 Correspondence 를 나타내며 σOut 은 S(GLHS)-I(GLHS)에서 S(GRHS)으로의 Mapping 으로 출력 Specification 간의 Correspondence 를 나타낸다.

Figure 3 은 "Compile, Schedule 및 Allocate" Task 들에 대한 Production 을 보여준다. Mapping 은 Specification Node 들 외에 숫자로도 표현될 수 있다. Figure 3a 는 다른 입력 Specification Type 에 대하여 동일한 출력 Specification Type 을 산출하는 경우이며, Figure 3b 는 Algorithm 에 의해 하나의 Sub-Graph 가 다른 Sub-Graph 로 대치 가능한 대안들을 나타내는 Production 을 의미한다.

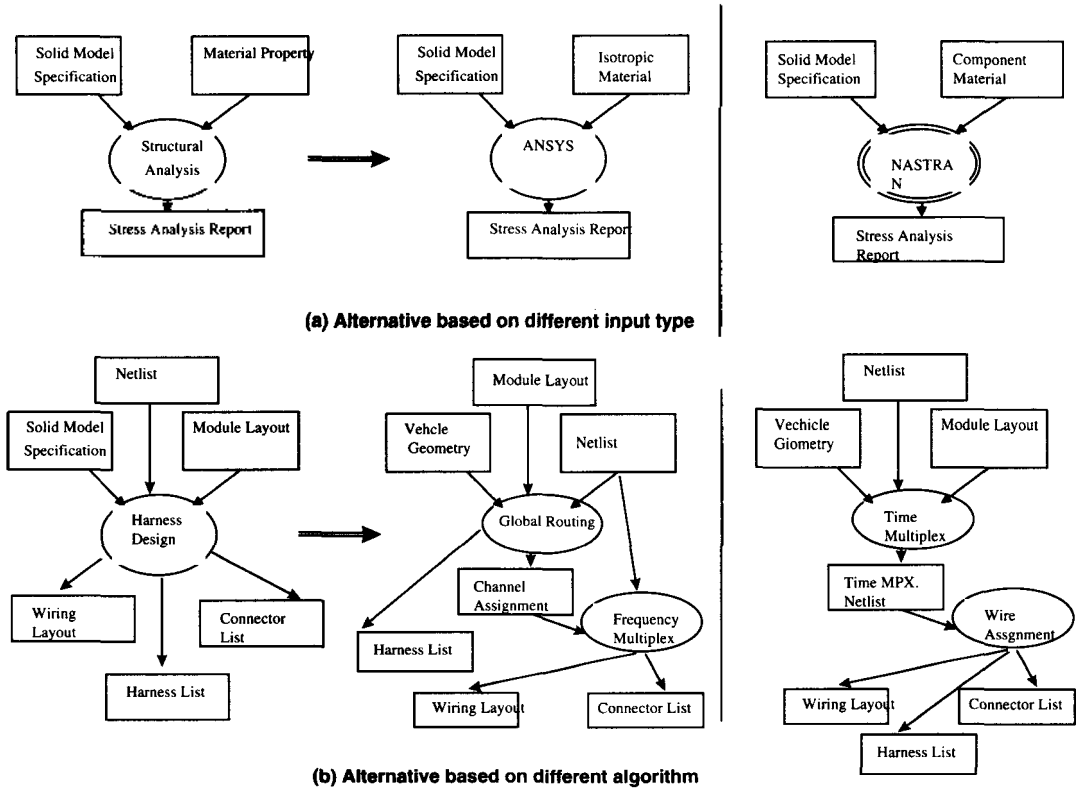

Figure 3. Two Types of Production

A 를 T(GLHS)의 Non-Terminal Task Node 라 하고 A'를 기존 Process Flow Graph G 의 Non-Terminal Task Node 라 하면 다음과 같은 조건이 만족될 경우 Production 에 의해 서로 Match 되어 전환될 수 있다.

- A' 가 A 와 같은 Task Label 을 갖는다.
- In(A)로부터 In(A')로의 Mapping ρIn 이 존재하며 모든 Node N ∈ In(A)에 대해 ρIn(N)이 동일 Type 또는 Sub-Type 의 N 을 갖는다.
- Out(A)로부터 Out(A')로의 Mapping ρOut 이 존재하며 모든 Node N ∈ Out (A)에 대해 ρOut(N)이 동일 Type 또는 Sub-Type 의 N 을 갖는다.

Mapping 은 새롭게 변형된 Sub-Graph 에서 Node 를 구성하는 방법을 결정하기 위해 사용된다. Graph G 에서 Match 가능성이 있으면 Production 은 다음과 같은 수순을 따라 적용된다.

- G 에 GRHS - I(GRHS)를 삽입한다
- I(GRHS)에 모든 N 과 GRHS 의 Edge(N, M)에 대해 (ρIn (σIn(N)), M) Mapping 을

Graph G 에 첨가한다.

- Out(A')에 모든 N 과 G 의 Edge(N, M)에 대해 Edge (N, M)을 Edge (σOut (ρOut (N)), M)로 대치한다.
- G 로부터 A'와 Out(A')를 제거한다.

Figure 4 는 Figure 3b 에서의 Production 을 이용해 전환된 예를 나타낸다.

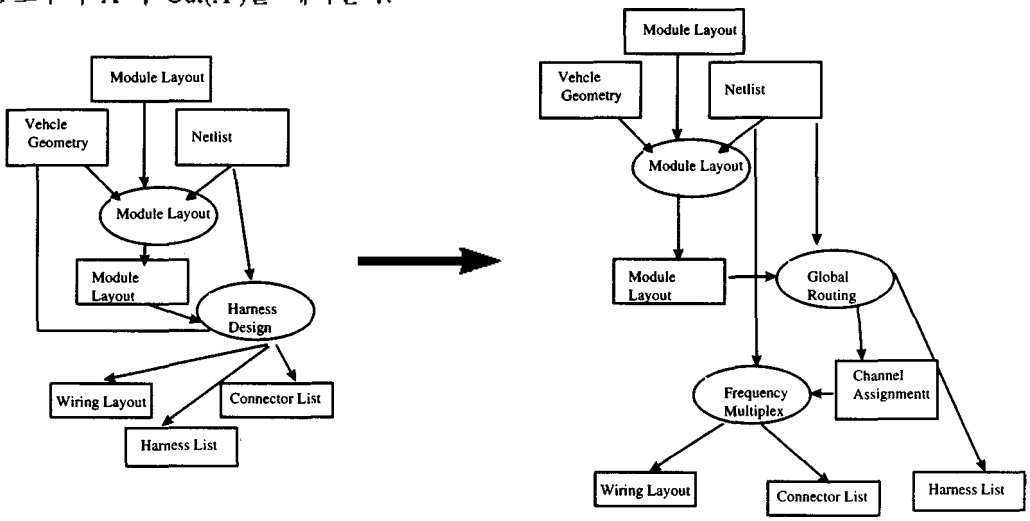

Figure 4. A Sample Graph Derivation

3. 시스템 구성 요소

분산된 조직을 갖는 가상기업에서의 실행 환경은 프랫폼에 독립적 (Platform Independent) 이어야 한다. 설계 Process 관리를 위한 Framework은 설계 작업을 Process의 정형화된 표현을 중심으로 Process Grammar를 통해 가능한 계층구조로 Decompose하고 이를 코드화 한다. 설계자는 설계 Process Grammar로부터 Process Flow Graph를 만들 수 있고 따라서 계층적인 설계 방법론을 얻어낼 수 있기 때문에 설계 공간 (Design Space)을 체계적으로 탐색할 수 있다.

설계자와 시스템과의 관계는 Cockpit이라고 하는 Program을 통하여 Interaction이 일어나도록 한다. 이 Cockpit은 현재 설계 상태를 계속적으로 추적하고 가능한 Action을 사용자에게 알려주며 설계자가 적절한 Action을 선택할 수 있도록 Manager Program과 서로 대화한다.

Manager Program들은 설계 Decomposition에 대한 평가를 하고 Software Tool을 Loading하면서 그 결과를 점검한다. Manager Program들은 어떤 특정 정보를 반영하기 위해 Tool Integrator라고 하는 것에 의해 Maintenance된다.

Tool Integrator는 Manager Program을 새롭게 만들때 시스템과 연계를 위한 Source Code를 자동적으로 만들기 위해 Tool Vendor에 의해 제공되는 정보를 이용한다. 또한 Tool Integrator에서 Cockpit에 입력 파일을 Maintenance시킬 수 있도록 하기 위해 Pre-Process 기능이 제공된다. 분산된 객체들간에 Communication은 개방형 산업 표준 (Industrial Open Standard)을 준수해야 한다.

Figure 5는 전술된 Grammar를 적용한 본 제안 시스템의 틀 구성을 나타낸다. 여기서 사용되어질 수 있는 Tool들에 대한 선정과 실행은 Manager Program에 의해 제안 되어 Cockpit Program에 전달 된다. 이때 Cockpit Program은 Manager Program과 설계자간에 상호작용을 계속적으로 관리 조정한다. 선택된 Tool Set과 실행 방법은 Tool이 설치 되어 있는 지역 Site와 시간에 따라 달라 질 수 있으므로 본 연구에서는 각 지역 Site에서 Tool-Dependent Code를 작성하고 Maintenance하는 System Integrator 역할을 하는 담당자가 있음을 전제로 하며 이를 위해 Tool-Independent Code 및 Templates를 지원한다.

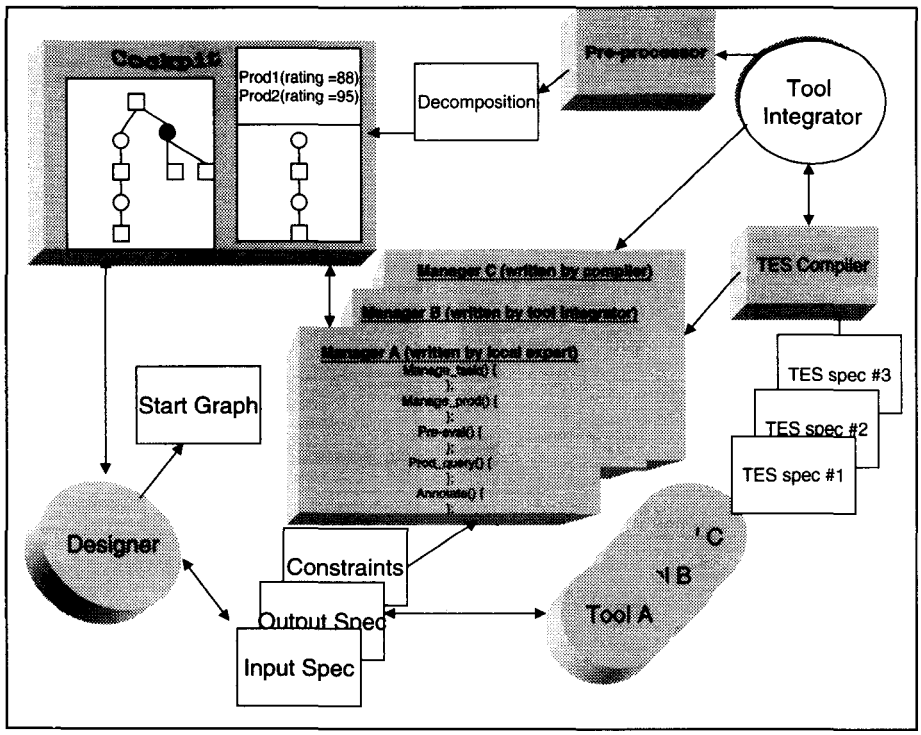

Figure 5. System Overview

3.1 Cockpit

Cockpit 은 설계 Process 의 현 상태를 추적 관리한다. 따라서 특정 Task 에 대한 Knowledge Source 를 갖는 Manager Program 과는 달리 Cockpit 은 Knowledge Source 가 내포되 있지 않다. 설계 Process 의 모든 정보는 Non-Terminal Node 에 대한 가능한 Task 와 Decomposition 의 집합을 나타내는 Process Grammar 에 따르는 입력 파일에 포함되어 있다. Cockpit 은 Text Editor 에서 산출된 입력 파일로부터 Initial Process Flow Graph 를 읽고 언제 Non-Terminal Node 로 Production 이 적용될 수 있는가를 계속적으로 나타내면서 Production Manager 에게 Constraints 에 근거한 Rating 을 요구하고 적절한 Production 이 적용되도록 한다.

설계자에게 Process Flow Graph 와 가능한 Production 에 Rating 이 보여지면 설계자는 Cockpit 이 Production 과 Task 를 특정 Node 에 적용하도록 한다. 그러면 Cockpit 은 Task Manager 에게 Message 를 보낸다. 이러한 절차를 따라서 Terminal Task Node 가 선택되면 원격지에 있는 해당 Tool 이 실행되며 Non-Terminal Task Node 의 경우에는 좀 더 상세한 Child Node 로 Decomposition 되기 위해 Task Manager 에 연계된 Production 이 적용된다. 즉 Cockpit 은 Production 을 적용하고 Production Manager 가 이를 실행하도록 한다.

다음은 Cockpit 의 Algorithm 을 나타낸다.

```
Initialization()
{
    Start graph is selected;
    Create initial daemon process and place tokens
    (send message);
}
Wait for message;
    IF the message is from execution process
THEN
    {
        IF the message is TO_ELABORATE THEN
        {
            Invoke pre-evaluation function;
            Select production based on pre-
            evaluation;
            Display expanded process flow;
            Send ELABORATE_THIS or FAILED
            message to execution process;
        }
        IF the Message is POST_EVAL THEN
        {
            Post-evaluation;
            Send result POST_EVAL_OK or _FAIL
            to execution process;
        }
        IF the message is FAIL_ELABORATE
        THEN
            Delete useless tokens;
        IF the message type is FAILED THEN
```

```
        Kill the child process;
    }
ELSE /* Message from daemon process */
  {
    IF the message is FAILED THEN
        Kill the child process;
    ELSE
    {
    Create a daemon from the subsequent task;
    Put the output token in the newly created in
    the newly created daemon's input place;
    }
}
END IF;
```

3.2 Manager Programs

Manager Programs 은 해당 Task에 대한 Knowledge Source를 기지며 새로운 Tool이 첨가되고 실행 중 얻은 경험을 일반화하여 지식을 Upgrade 한다. 각각의 Manager Program 은 다음과 같은 5 개의 기능을 갖는다.

- 사전 평가 (Pre-Evaluation)
- 도구 실행 (Tool Execution)
- 추상 작업의 실행 (Abstract Task Execution)
- 프로덕션 실행 (Production Execution)
- 질의 처리 (Query Handling)

Pre-Evaluation: Production Manager는 설계자에게 각 Production에 대해 Rating을 주고 Task Manager가 최적의 Production을 선정하도록 한다. Rating은 각 Production에 대해 성공 정도를 1과 100 사이의 숫자로 수치화 한 것으로 System Integrator에 의해 사전 정의된다. 설계자는 Production에 대한 변수를 정의하여 ASCII Format 형태로 표기한다. 다음은 Production Rating의 한 예를 보여준다.

```
Arch_Syn. PRE ./Preeval1
Arch_Syn. POST ./Hello
Arch_Syn. 0 Time 9 Pref 3 History 4
Arch_Syn. 1 Time 11 Pref 3 History 5
Arch_Syn. 2 Time 4 Pref 6 History 2
```

Production Arch_Syn 이 적용되면 사용자는 현재의 사용 Directory에 있는 Preeval1 이라는 Pre-Evaluation Function을 사용한다. 마지막 세 개 Line은 실제 변수와 사용자에 의해 지정된 값을 나타낸다. 설계자는 이 값을 이용해 Pre-Evaluation Function을 작성한다. 간단한 예는 다음에서와 같다.

```
#include <stdlib.h>
#include <stdio.h>
main (int argc, char **argv)
{
int t, p, h;
int score;
if (argc < 7)
```

```
    exit (-1)
t=atoi (argv[2]);
p=atoi (argv[4]);
h=atoi (argv[6]);
score = t*0.2 + p*0.1 + h*0.7;
exit(score);
}
```

여기서 T, P 및 H는 각각 해당 Production 종결 시점, 변환되어 지는 Input File Type에 대한 Penalty 값을 갖는 변수 및 해당 Production 이 사용 가능 했던 기간을 의미한다.

정적 (Static) Rating은 해당 Task Node에 비성공적인 전례가 있으면 조정될 수 있다. Rating은 질의 기능이나 입력 파일의 분석에서 얻어진 Parameter일 수 도 있다. Manager Program은 계속적으로 성공 조건을 나타내는 Process 행렬을 수집 분석할 수 있다.

Tool Execution: Terminal Task는 해당 Task Manager에 의해 Software Tool 실행을 하고 그 성공 여부를 판단한다. 대부분의 경우 정보는 사전 정의 되고 표준 Template으로 입력된다. 다른 경우는, Manager는 Task-Specific Knowledge를 검사하여 Tool Parameters를 결정하거나 Task-Specific Constraints을 확인하여 성공 여부를 결정한다.

Abstract Task Execution: Non-Terminal Task에 대해 Task Manager는 Abstract Task를 실행할 Production을 선택한다. Cockpit은 Task Manager에 사용 가능한 Production과 해당 Rating을 알려준다. Task Manager는 Cockpit이 한 개 이상의 Production을 적용 및 실행하도록 하거나 실패 여부를 알려준다. 만일 해당 Production 이 성공하면 Task Manager는 Constraints을 확인한 후 만족되면 성공 Message를 나타낸다.

Production Execution: Production Manager는 Production의 우측 (Right-Hand Side)에 해당하는 Task들을 실행한다. 만일 이들 중 하나의 Task가 구속조건에 위반되면 Backtracking이 된다. Production Manager는 특정 업무에 관련된 지식을 가지고 있으며 Production Manager가 실패(Failure)를 처리 할 수 없을 경우에는 이를 Cockpit에 알리고 보다 상위 Level의 Task Manager가 이를 처리한다.

Query Handling: Production Manager와 Task Manager들은 Query Mechanism에 관여되어 이들은 상위 Parent Manager에게 또는 하위 Child Manager에게 질의를 하게 된다.

4. 시스템의 객체 모델

제안된 Process Management System은 객체지향 방법론을 이용하여 Internet을 기반으로 하는 Network 환경에서 개발되었다. Process Flow, Process Grammar 및 Process 관련 지식을 모델

령하기 위해 OMT (Object Modeling Technique) 기법 [Rumbaugh et al., 91]이 이용되었으며 개발 언어는 Web Program을 위해 Java, Perl 및 Tcl/Tk가 이용되고 기타 모듈은 C++로 개발되었다.

Figure 6 은 OMT 로 표현된 Process Management System 의 상위 Level 객체 모델로서 각 객체간의 관계성을 나타낸다.

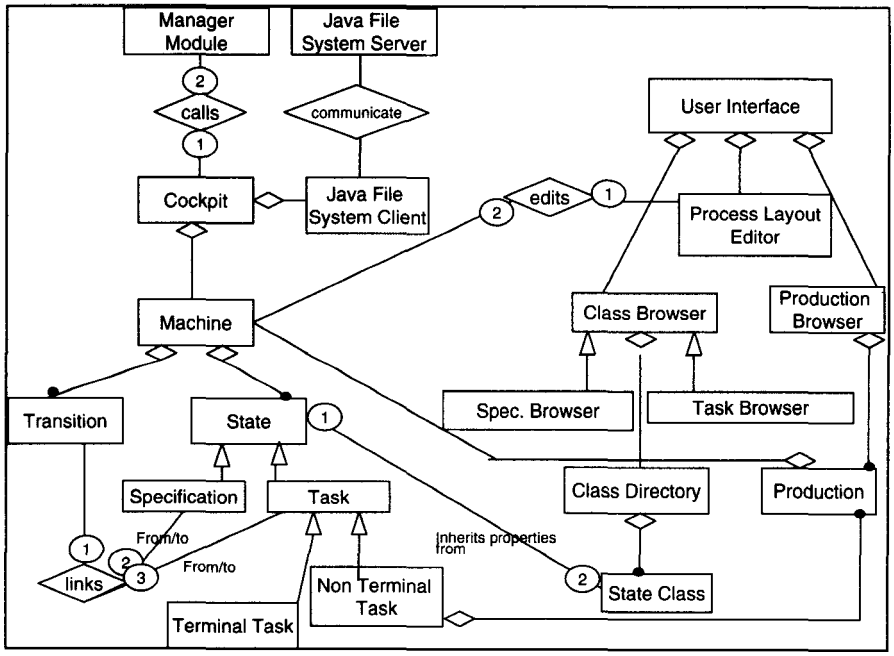

Figure 6. High-Level Object Model of Process Management System

Cockpit: Cockpit은 Class *COCKPIT* 으로 시작되고 Class *PRMAIN* 은 설계 Process Management System에 대한 Control Point이다. 시스템이 시작될 때 Cockpit은 Root Process Flow가 되는 Class *MACHINE* Instance를 산출한다. Cockpit의 가장 중요한 기능은 Process Simulation 제어로서 Cockpit은 *STATE* Object 들 Rollback 및 Pre- 및 Post-Evaluation을 담당하는 다양한 Manager Module과의 Communication을 관리 통제한다.

마지막으로 Java File System Server 와의 연계를 제어하여 특정 Process 가 저장 및 Loading 되도록 한다.

Manager Module: 이는 *COCKPIT* Class 와 밀접한 외부 Program 으로 특정 *NON-*

*TERMINAL-TASK*에 대해 모든 적용 가능한 *PRODUCTIONS*을 평가 하고 각 Production에 대한 Rating 값을 산출한다. (Pre-Evaluation 단계). 또한 Managers는 *PRODUCTION*의 출력 값을 평가하게 된다 (Post-Evaluation 단계).

Java File System: Java Applet에서 생성된 File은 원격지의 Java File System (JFS)에 저장하게 한다. *COCKPIT* Class는 실행되는 JFS Server와 Communication되는 JFS Client를 Maintenance 한다.

Machine: *MACHINE* Class는 설계 Process Management System의 가장 중심이 되는 객체 중 하나로 *TRANSITIONS* 과 *STATES* 의 Vector를 Maintenance함으로서 Process Flow를 표현 한다.

Transition: *TRANSITION* Class는 *STATE* Class와 논리적 연결 (Logical Link)을 가지며 *TRANSITION* Class의 Instantiation은 *SPECIFICATION*에서 *STATE*로 혹은 *STATE*에서 *SPECIFICATION*로 연계된다.

State: *STATE* Class는 *SPECIFICATION* 및 *TASK*의 상위 Class로 *MACHINE* 이 Finite Automaton로 표현 되는 반면 *STATE* 는 Finite Automaton에서 Generic Node를 나타낸다.

- *SPECIFICATION*은 *TASK* 의 입/출력 역할을 하는 *STATE*의 한 Type이다.
- *TASK*는 *TERMINAL-TASK*와 *NON-TERMINAL-TASK* 두 Sub-Type의 상위 Class이다.
- *NON-TERMINAL-TASK* Class는 Abstract Task를 나타내어 적어도 한 개 이상의 *PRODUCTION* Instance를 갖는다. *TERMINAL-TASK* 는 실제 Tool과 연계된다. *TERMINAL-TASK*가 Process Simulation이 될 경우 원격지의 Tool은 Invocation이 된다.

User Interface:

- **Process Layout Editor-** 몇 개의 Abstract Windowing Toolkit (AWT) Graph Class로 구성되어 있고 *MACHINE*을 구성하는 *STATE*와 *TRANSITION*로 조정 관리하기 위한 기능을 갖는다.
- **Class Browser-** 설계 Process Management System에서는 *TASK-BROWER*와 *SPECIFICATION-BROWSER*등의 두개 Class Browsers가 제공 되는데 이들은 AWT 구성요소와 동일하며 *STATE-CLASS* 구조를 Maintenance하기 위한 *CLASS-DIRECTORY* 의 Instance를 사용한다.
- **Production Browser-** Production Browser는 모든 *PRODUCTION*과 각 *PRODUCTION*이 포함되는 *NON-TERMINAL-TASK*의 Vector를 관리 하

며 사용자에게 Process Layout Editor를 이용하여 각 *PRODUCTION* 을 조정하기 위한 AWT 기능들을 제공한다.

Production: 모든 *NON-TERMINAL-TASK* 은 적어도 하나의 *PRODUCTION* Class에 Instance를 갖는다. 각 *PRODUCTION* 은 특정한 *NON-TERMINAL-TASK*에 대한 하나의 가능한 대안 Process Flow를 나타낸다. 이러한 Process Flow의 Maintenance는 *MACHINE* Class의 Instance를 통해 이루어진다.

Class Directory: Class Browsers는 *STATE-CLASS* 객체의 Vector를 Maintenance하기 위한 *STATE-CLASS-DIRECTORY* Class의 Instance를 사용한다.

5. 결론

본 논문에서는 기업 통합 구현을 위한 가장 중요한 요소 중의 하나인 설계 Process 관리를 위해 Client/Server 환경 하에서 다양한 설계 요구조건을 만족하여 최적 해를 얻을 수 있는 Framework 구축 방안을 제시하였다. 본 Process Management System에서는 설계 Process를 표현하기 위한 Process Grammar를 개발하고 관련 Data들 간의 일관성을 유지하면서 설계 Process에서 발생되는 Activity들을 관리 통제하는 기능을 포함하였다. 이러한 Framework의 개발을 통해 설계를 효율적으로 수행하고 설계 개발 기간을 크게 단축할 수 있다. 제안된 Framework은 다음과 같은 장점을 갖고 있다.

- Formalism: 설계 Process의 관리를 위한 이론적 기반을 구축하여 이를 통해 본 시스템이 다른 방법론과 함께 어떻게 활용되는지를 분석할 수 있다.
- Parallelism: 몇 개의 대안을 동시적으로 탐색할 수 있게 해 준다. 이는 설계자로 하여금 컴퓨터 자원을 보다 잘 활용할 수 있게 함으로써 설계 시간을 줄일 수 있다.
- Reusability: 설계 Process가 저장되고 수정 및 재사용될 수 있다. 재설계 과정과 이용되는 Tool은 이전 Component의 설계 Data에 크게 의존하며 이때 제시된 Process Grammar는 이러한 관련성을 얻고 재설계 활동에 대한 방향을 제시하는데 특히 유효하다.
- Flexibility: 제시된 Framework은 방법론과 수행환경을 명확하게 구분한다.
- User Friendliness: Framework은 Internet Web 상에서 GUI환경으로 사용자와 Interactive하게 설계 Process를 결정한다.

References

[Casotto et al., 90] Andre Casotto, A. Richard Newton, and Alberto Sangiovanni-Vincetelli. "Design Management based on Design Traces". 27th ACM/IEEE Design Automation Conference, pp 136-141, 1990.

[Chiueh., 90] Tzi-cker F. Chiueh and Randy H. Katz. "A History for Managing the VSLI Design Process. International Conference on Computer Aided Design", pp 358-161, 1990.

[Corkill et al., 87] Daniel D. Corkill, Kevin Q. Gallagher, and Philip M. Johnson, "Achieving Flexiblility, Efficiency, and Generality in Blackboard Architectures", Proceedings of the National Conference on Artificial Intelligence, 1987, pp18-23

[Daniell, 91] James Daniell and Steven W. Director. "An Object Oriented Approach to CAD Tool Control". IEEE Transactions on Computer-Aided Design, pp 698-713, 1991.

[Di Janni, 86] Alberto Di Janni. "A Monitor for Complex CAD Systems". Proceedings of the 23rd Design Automation Conference, pp 145-151, 1986.

[Fairbairn, 94] Douglas G. Fairbairn. 1994 Keynote Address. Proceedings of the 31the Design Automation Conference, pp xvi-xvii, 1994.

[Hsu, 96] Meichun Hsu and Charley Kleissner. "ObjectFlow: Towards a Process Management Infrastructure". Distributed and Parallel Databases, 4:169-194, 1996.

[Jacome, 92] Margarida F. Jacome and Stephen W. Director. "Design Process Management for CAD Frameworks". Proceedings of the 29th Design Automation Conference, pp 500-505, 1992.

[Knapp, 92] David Knapp and Alice Parker. "The ADAM Design Planning Engine". In AI in Design, Vol. II, pp 263-285. Academic Press, 1992.

[Rumbaugh et al., 91] James Rumbaugh et al. "Object-Oriented Modeling and Design". pp 57-79. Prentice Hall, 1991.

저자소개

박화규

동국대학교 산업공학과 (학사)

California State University 산업공학과 (석사)

Oklahoma State University 산업공학과 (박사과정)

현재 시스템공학연구소 시스템 통합연구부 연구원

관심분야 : CALS, Concurrent Engineering, PDM (Product Data Management), Machine Learning,

김 현

한양대학교 기계설계학과 (학사)

한양대학교 기계설계학과 (석사)

한양대학교 기계설계학과 (박사)

현재 시스템공학연구소 시스템 통합연구부 선임연구원

관심분야 : Concurrent Engineering, CAD/CAM/CAE, Design Automation

오치재

서울대학교 재료공학과 (학사)

Texas A&M University 산업공학과 (석사)

Auburn University 산업공학과 (박사)

현재 시스템공학연구소 시스템통합연구부 책임연구원

관심분야 : System Integration, CALS, Concurrent Engineering, PDM (Product Data Management)

성문정

서울대학교 공과대학 (학사)

한국과학기술원 전산학과 (석사)

Northwestern University 전산학과 (박사)

현재 Michigan State University 전산학과 부교수

관심분야 : Algorithms in Design Automation, Design Process Management, Parallel Algorithm