CIM 시스템 설계를 위한 기능 모형-프로세스 흐름 모형 변환에 관한 연구
Function-to-Process Flow Model Transformation for Design of CIM Systems

박찬권*, 신기태**, 박진우*
* 서울대학교, 산업공학과, ** 대진대학교 산업공학과

Abstract

Models from different perspectives including functional view, behavioural view and information view are necessary to implement complex systems such as CIM. It is difficult to integrate those models into a unified framework since they describe different aspects of the system for different purposes. However, modeling from different perspectives is often the case, and many workers are involved in developing a system. Thus, the transformation or interconnection mechanism is required to cope with the consistency problem between them and to save the efforts during development.

In this study, a methodology to thansform the IDEF0 model into a process flow model is proposed.

1. 서 론

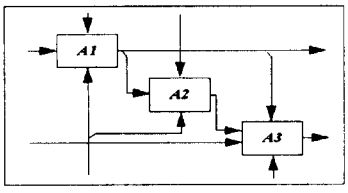
모형의 주된 사용 목적은 연구 대상 시스템을 연구 목적에 맞게 묘사하기 위해서 이다. 그런데 제조 시스템을 포함해서 모형화 대상으로서의 많은 실제 시스템은 범위, 수명 주기, 그리고 관점(perspective)과 같은 특성들을 가지는데, 이러한 특성들을 어떻게 묘사할 것인가에 따라 다양한 형태의 모형을 필요로 한다.

이들 모형들은 서로 다른 필요에 따라 시스템의 특정 관점을 묘사하고 있어서 하나의 단일 모형이나 모형화 방법론을 통해 시스템에 대한 통합적인 묘사가 불가능하다. 그러나 서로 다른 관점에서의 모형일지라도 하나의 시스템을 대상으로 하기 때문에, 설계 및 개발 과정에서 이들 모형들의 연계 관계가 규명되어야만 일관적인 시스템의 구축이 용이하다. 또한 모형의 개발 단계에서 특정 관점에서 개발된 모형과는 별도로 전혀 다른 관점에서 새롭게 정보 모형이나 프로세스 흐름 모형 등을 개발하는 것보다는 기존의 기능 모형과의 연계 관계를 바탕으로 진화시키는 것이 바람직하다. 더욱이 전체 모형들이 개발된 후에는 특정 관점에서의 모형 변경 사항이 여타 모형들에 적절히 반영될 수 있기 위해서는 이러한 연계 관계가 필수적이다. 특히 다양하고 복잡한 구성 요소들을 바탕으로 기업에 있어 서로 관련된 제반 기능들에 대한 유기적인 통합을 목적으로 하는 컴퓨터통합생산시스템은 시스템 제공업자로부터 일괄적으로 구매가 어려워 기업 상황에 맞는 고유의 시스템을 개발해야 하기 때문에, 개발 과정에 존재하는 조직 관리 구조, 기능 구조, 정보 구조, 그리고 컴퓨터 시스템 구조와 같은 기능 단위들간의 기능적 연계 관계가 반드시 존재해야 한다.

본 연구에서는 시스템 요구 정의(system requirement specification)를 위해 보편적으로 사용되고, 따라서 컴퓨터통합생산시스템의 초기 설계 단계에서 기능 모형화(function modeling)를 위해 사용하는 IDEF0 모형으로부터 시스템 동작 행태를 묘사하기 위한 프로세스 흐름 모형(process flow model)으로 변환하기 위한 방법을 제시함으로써 시스템에 대한 관점(perspective)의 연계를 위한 틀을 제공하고자 한다.

2. IDEF0 기능 모형화 방법론

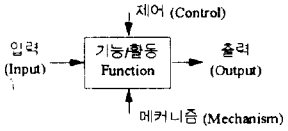
기능을 모형화 하는 도구로써 IDEF0는 제약 조건 아래서 입력을 출력으로 변환시키는 활동을 묘사하는데 사용된다. IDEF0의 가장 중요한 요소는 그래픽 표현과 계층적 구조이다. IDEF 모형은 [그림 1]과 같이 정의되는 도표(diagram)의 연속된 계층으로 구성되는데, 박스와 화살표를 통한 기능 및 기능간의 관계를 정의하고 부수적으로 글과 설명을 통한 보조 정보를 표현한다.



[그림 1] IDEF0의 계층적 분해에 의한 모형화

[그림 2]와 같이 정의되는 '기능'은 기본적으로 활동, 프로세스, 운용 또는 변환 작용을 나타내는 기능을 정의한다.

기능을 나타내는 박스와 관련되는 데이터 또는 객체를 표현하기 위해서 하나 또는 그 이상의 화살표를 사용한다. ICOM이라 불리는 이 화살표는 [그림 2]에서와 같이 박스에서의 상대적인 위치에 따라 입력, 제어, 메카니즘, 출력의 역할이 정해진다.



[그림 2] IDEF0의 ICOM

기능을 나타내는 박스를 연결하는 화살표는 한 박스의 출력이 다른 박스의 기능 수행에 필요한 입력, 제어, 또는 메커니즘으로 제공되는 것을 의미하게 된다. 화살표는 여러 박스로의 입력을 위해서 분기하기도 하고, 여러 박스에서의 입력을 위해 병합되기도 한다.

IDEF0는 시스템 설계에 있어 구조적인 방법론을 제공하는 등 여러 장점에도 불구하고, 여타 IDEF 방법론들과의 연관 관계가 취약하다는 단점이 있다. 즉, 기능 모형, 정보 모형, 그리고 동적 모형에 있어 구성 요소 상의 관계 확보가 어렵기 때문에 기능 모형에 제공하지 못하는 특성, 예컨대 시간의 추이에 따른 시스템의 동작 행태를 표현하거나, 기능 모형에 있어 활동의 시간적 순서 관계,

동시 수행 가능성 등을 파악하기가 어렵다는 것이다.

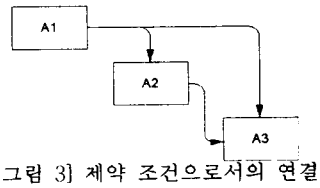
3. 프로세스 흐름 모형으로의 변환

IDEF0 기능 모형에는 활동 또는 기능의 시간적 순차 관계가 표현되지 않는다. IDEF0를 바탕으로 활동의 순차 관계를 표현할 수 있는 프로세스 흐름 모형으로의 변환은 기능 모형(function model)이 기능 또는 활동(activity) 위주로 표현되고, 프로세스 흐름 모형(process flow model) 또한 활동(activity) 위주의 묘사라는 점에서 출발한다. 즉, IDEF0 모형의 중심이 되는 기능(function)이 활동(activity), 의사 결정(decision making), 운용(operation), 행위(action), 또는 운영(operation) 등을 묘사하고, 이러한 기능이 프로세스 흐름 모형에서 그대로 단위 활동(unit of activity)으로 전환될 수 있기 때문이다.

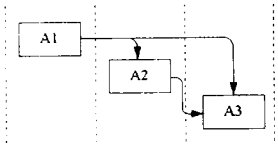
3.1 화살표에 의한 연결의 해석

IDEF0에서 화살표는 하나의 박스에서 또 다른 박스로의 입력이나 제어 또는 메커니즘의 제공을 나타냄으로써 두 개의 박스 즉, 기능간에 존재하는 관계를 표현하도록 한다. IDEF0에서 두 활동을 연결하는 화살표는 기본적으로 제약 조건(constraints)의 의미와 동시 실행(concurrency)의 의미로 해석될 수 있다.

먼저 [그림 3]은 한 기능에서의 출력이 또 다른 기능의 제약 조건(constraints)이 되는 경우이다. 즉, 기능 A1이 종료되고 A1에 의해 생성된 데이터나 객체가 A3에 제공되어야 기능 A3이 시작할 수 있음을 의미한다. 마찬가지로 A3의 수행에는 기능 A2로부터의 출력이 필요하다. 이 것은 IDEF0가 입력, 제어, 메커니즘의 역할이 제약 조건임을 명시하고 있기는 하지만, 의미상으로 기능들 사이에 순차 관계(precedence)가 존재하고 있다는 것을 보여준다.

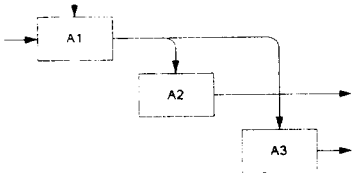


[그림 3] 제약 조건으로서의 연결



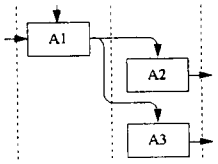
[그림 4] 순차 관계의 정의

따라서 선후 관계가 분명히 존재하는 경우는 [그림 4]와 같이 순차(precedence) 관계를 정의한다. 원래의 IDEF0 모형이 기능을 표현하는 박스를 대각선으로 배열하는 반면, 여기서 순차 관계로 정의되는 박스는 가로축을 기준으로 하여 후행하는 기능이 오른쪽에 위치하도록 배열한다.



[그림 5] 동시 실행의 표현

한편, [그림 5]와 같은 경우, IDEF0는 기능의 동시 실행(concurrency)을 의미하게 되는데, 기능 A1이 종료한 후 일단 출력으로 데이터 또는 객체가 제공되기만 하면, 그 출력을 제약 조건으로 하는 활동 A2, A3은 동시에 실행이 가능하다는 것이다. 이 경우에 있어서도 A1은 A2, A3 보다 먼저 실행 및 종료되어야 하는 순차 관계(precedence)를 내포하게 된다. 따라서 A1과 A2 또는 A1과 A3 사이에는 앞서의 [그림 4]와 같은 순차 관계가 정의되고, A2와 A3 사이에는 [그림 6]과 같은 병렬 수행 관계가 정의된다. 순차 관계와 마찬가지로, 동시 수행 관계에 해당하는 기능들은 세로축을 기준으로 상하에 위치하도록 배열한다.



[그림 6] 동시 수행 관계의 정의

(1) 출력-입력

출력으로부터 입력으로 제공될 수 있는 실체는 데이터나 자재가 있다. 하나의 기능으로부터 출력이 또 다른 기능의 입력으로 제공되기 위해서는 그 것이 데이터이든지 자재든지 두 기능 사이의 순차 관계를 정의한다고 볼 수 있다. 즉, 하나의 기능이 종료된 후 출력이 제공되어야 그 것을 입력으로 하는 기능이 시작될 수 있다.

그러므로 출력으로부터 입력으로의 연결 관계는 모두 [그림 4]와 같은 순차 관계로 정의된다.

(2) 출력-제어

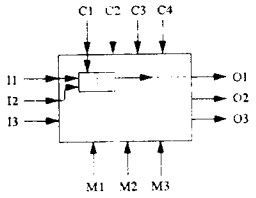
제어 가운데서 기능의 수행 환경이나 조건을 지정하는 제어와 사건(event) 제어는 입력과 마찬가지로 순차 관계를 정의한다.

한편 피이드 백의 경우는 반복 수행이 필요 없는 정 피이드 백(positive feed back)의 경우는 입력과 마찬가지로 순차 관계로 정의될 수 있지만, 반복 수행을 수반하는 교정 피이드 백(corrective feedback)은 단순 순차 관계로 정의되기 어렵다. 이 경우는 뒤에서 설명될 활성 모드(activation mode)의 분석에 따라 순차 관계가 정의된다.

동기 제어(synchronization control)의 경우는 기능간의 순차 관계를 직접 지정하고 있기 때문에 제어의 내용을 구체적으로 해석해서 순차 관계와 병렬 수행 관계를 정의한다.

(3) 출력-메커니즘

출력-메커니즘의 경우도, 출력-입력과 마찬가지로 순차 관계를 정의한다. 다만 출력-메커니즘은 분석하고자 하는 시스템의 경우에 따라 다르지만 자주 발생하는 일반적인 상황은 아니다.



[그림 7] 분해된 기능에서의 입력, 제어, 출력의 조합

3.2 활성 모드(activation mode)

IDEF0 모형은 하위 계층으로 내려감에 따라, 화살표의 존재가 어느 정도 순차 관계로 해석될 수 있는 가능성을 가지고 있음에도 불구하고 전체적으로는 순차 관계에 의한 프로세스 흐름을 표현하지 못하는 이유는 하나의 기능이 활성화(activation)되는데 모든 입력과 출력이 동시에 관여하지 않기 때문이다. 예를 들면, [그림 7]과 같이 기능이 정의되는 경우 출력 O1, O2, O3를 내놓기 위해서 입력 I1, I2, I3 및 제어 C1, C2, C3, C4 그리고 메커니즘 M1, M2, M3이 전부 필요한 것은 아니고, 그 중 일부의 조합(I1, I2, C1)들이 특정 출력(O1)을 제공하는데 관여할 수 있기 때문이다. 이 경우에는 입출력의 조합에 따라 다

중활성 모드를 정의할 수 있다.

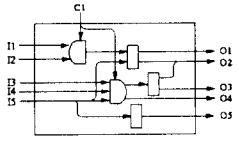
결국 IDEF0에는 이러한 다중 활성 모드가 존재함에 따라, 순차 관계에 의한 프로세스 흐름 분석이 불가능하게 되는 것이다. 따라서 다중 활성 모드가 존재하는 경우에는, 한 번의 수행을 통해 관여하게 되는 입력, 제어, 메커니즘 그리고 출력의 적절한 조합을 찾아냄으로써 프로세스 흐름 모형으로 변환이 가능하다.

다중 활성 모드는 하나의 기능에 관여하는 입력, 제어, 메커니즘, 그리고 출력의 수가 많지 않을 때는 직관적으로 쉽게 찾을 수 있지만, 상위 레벨과 같이 여러 개의 입력, 제어, 메커니즘, 그리고 출력이 있는 경우는 쉽지가 않다. 따라서 다음과 같은 절차를 이용한다. 즉, 기능/활동 A^z에 대해, l개의 입력, m개의 제어, n개의 메커니즘, 그리고 t개의 출력이 정의되어 있을 때, 가능한 출력 조합의 수(NOC)는 다음과 같다.

$$NOC = \text{Rank of } [CM^z] \ (\ i=1,\ldots,\ l,\ l+1,\ldots,\ l+m,\ l+m+1,\ \ldots,\ l+m+n,\ j=1,\ \ldots,\ t\ )$$

여기서 CM^z는 입력/제어/메카니즘과 출력 사이의 조합 관계를 표현한 행렬이고, cm_{ij}는 입력 또는 제어 i가 출력 j를 제공하는데 관여하는 횟수를 나타낸다. 그렇게 되면, 행렬 CM^z의 Rank는 서로 독립인 행 또는 열의 수를 나타내고, 이것은 출력의 조합을 제공하는데 필요한 입력 및 제어 조합으로 해석된다.

예를 들어, [그림 8]과 같은 기능/활동이 주어졌다. 여기서 입력과 제어, 그리고 출력의 논리적인 조합은 'D' 모양의 게이트로 표현했다. 즉, 'D'의 오른쪽 둥근 부분으로 들어가거나 나오는 화살표는 선택(selection) 분기(논리적 OR)로 해석하고, 왼쪽 수직한 부분으로 들어가거나 나오는 화살표는 다중 분기(논리적 AND)로 해석한다. 'D'가 뒤집어진 경우도 같은 방법에 의해 해석한다. 한편 입력이 들어가서 출력으로 변환되는 과정은 직사각형으로 표현한다 여기에 사용되는 제어의 종류는 선택 제어(selection control)이다.



.[그림 8] 다중 활성 모드의 예

이와 같은 조합이 주어 이루어질 수 있다면, 공헌 행렬(contribution matrix)은

$$CM = \begin{array}{c} \\ I1 \\ I2 \\ I3 \\ I4 \\ I5 \\ C1 \end{array} \begin{array}{c} \begin{array}{ccccc} O1 & O2 & O3 & O4 & O5 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

이 된다.

여기서, CM의 Rank는 3으로 세 개의 입력 및 제어와 출력의 조합이 존재한다. 즉, 입력 I1, I2, I5와 제어 C1이 출력 O1 및 O2를 생성하고, 입력 I3, I4, I5와 제어 C1이 출력 O2, O3, 그리고 O4의 생성에 관여한다. 마지막으로 입력 I5는 출력 O5를 내놓게 된다. 이 세 가지 조합 가운데 한 번의 기능 활성화에 의해 직접 나올 수 있는 출력의 조합들이 하나의 활성 모드를 결정하게 된다. 예컨대, 처음 활성화 때, 출력 조합 O1/O2와 O5가 생성될 수 있다면, O1, O2, O5는 하나의 활성 모드를 구성하게 된다.

4. 모형 변환 절차

활성 모드, 사이클, 출력으로부터 연결의 의미 등을 고려함으로써 IDEF0 기능 모형으로부터 프로세스 흐름 모형의 변환은 다음과 같은 과정을 통해 이루어진다.

4.1 활성 모드의 결정

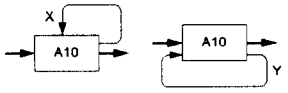
(1) 기능 A^z에 대하여, 공헌 행렬(contribution matrix) CM^z을 작성한다. 여기서 l개의 입력과 m개의 제어는 모두 행을 구성하고, n개의 출력은 열을 구성한다. cm^z_{ij}는 입력 또는 제어 i가 출력 j를 생성하는데 직접적으로 기여하는 횟수를 나타낸다.

(2) CM^z의 Rank를 구하면, 이것이 곧 기능 A^z에 대한 활성 모드의 개수가 된다.

(3) 그리고 최대 완전 이분할성 그래프 알고리즘을 적용해서 각 활성 모드에 관여하는 입력, 제어, 그리고 출력의 조합을 구한다.

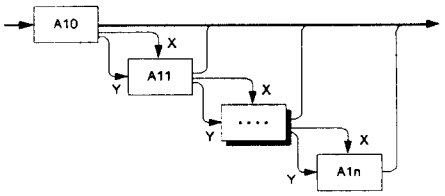
4.2 기능의 분해에 의한 사이클의 제거

앞서 언급한 바와 같이 IDEF0 기능 모형이 순차 관계의 표현이 불가능하고, 프로세스 흐름을 모형화 할 수 없는 이유는 다중 활성 모드에 있다. 이와 같은 다중 활성 모드는 기능이 여러 가지 하부 기능을 추상화해서 표현하는데도 원인이 있지만, 근본적으로는 피이드 백과 같은 사이클이 존재하기 때문이다. 따라서 앞서 구한 활성 모드의 수만큼 기능을 해체(split)해서, 이러한 사이클의 발생을 제거하도록 한다.



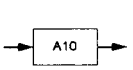
[그림 9] 순환 반복(iteration)

사이클은 반복 순환(iteration)과 피이드 백이 있다. [그림 9]와 같은 반복 순환은 활성 모드에 따라 기능을분해하면 [그림 10]과같이 표현될 수 있다. 이러한 단순 반복은 전체 프로세스의 흐름을 파악하는데 도움이 되지 않기 때문에, 특별히 필요한 경우가 아니면 [그림 11]과 같이 생략한다. 이러한 상황은 [그림 10]에서 굵은 실선으로 표시된 주 흐름을 분석해 보면 알 수 있다.



[그림 11] n회의 순환 반복 해체

[그림 12]와 같은 피이드 백의 경우는, [그림 13]과 같이 각 활성 모드에 관여하는 입력, 제어, 그리고 출력의 조합에 따라 기능 해체(split)해서 표현하면, 사이클을 제거할 수 있다.



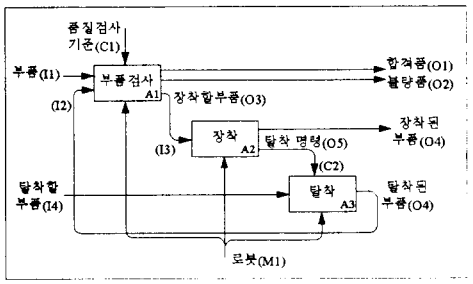
[그림 10] 순환 반복(iteration)의 생략

기능/활동 A1에 대해 공헌 행렬을 작성해 보면,

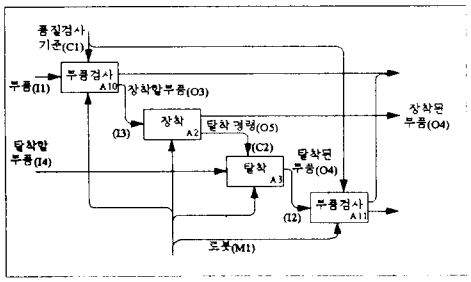
$$CM^1 = \begin{array}{c} \\ I1 \\ I2 \\ C1 \\ M1 \end{array} \begin{array}{c} \begin{array}{ccc} O1 & O2 & O3 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \end{array}$$

이 된다.

CM[1]의 Rank는 2로, 두 개의 출력 조합이 존재함을 알 수 있다. 즉, I1과 C1, 그리고 M1이 출력 O2 또는 O3를 생성하거나, I2와 C1, 그리고 M1이 O1 또는 O2를 생성한다. 여기서 두 개의 출력 조합은 A1이 수행된 후, 즉 한 번의 활성화로 제공될 수 없는 조합이므로 두 개의 활성 모드를 결정한다. 물론, 이 경우는 관여하는 입력, 제어, 그리고 출력 사이의 관계가 비교적 단순하기 때문에 애초에 기능 모형을 작성하는데 관여한 전문가가 곧바로 활성 모드의 수를 발견할 수 있다. 많은 수의 ICOM이 복잡하게 존재하는 경우는 공헌 행렬을 통해 활성 모드를 결정한다. 따라서 기능 A1은 [그림 13]처럼 A10과 A11로 해체될 수 있고, 결과적으로 사이클은 제거되었음을 발견할 수 있다.



[그림 12] 피이드 백에 의한 사이클이 있는 경우

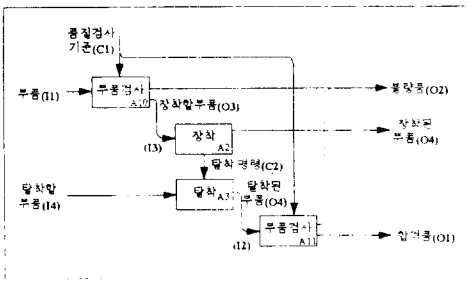


[그림 13] 사이클의 해체

4.3 화살표의 유형 분류

앞서 설명한 바와 같이 동기 제어(synchronization control)를 제외한 모든 화살표는 직접적인 선후 관계를 정의한다. 즉, 출력을 제공하는 기능 수행이 끝나고, 입력이나 제어 또는 메커니즘을 제공받는 기능이 시작되는데, 이 경우는 [그림 4]와 같이 기능을 배열한다. 동기 제어(synchronization control)는 직접적인 순차 관계를 선후 관계와 병렬 수행 관계로 나누어 선후 관계는 [그림 4], 동시 수행 관계는 [그림 6]과 같은 방법으로 배열하도록 한다. 특별히 동시 수행 관계를 표현하는 동기 제어는 점선으로 표현한다.

이렇게 해서 작성된 프로세스 흐름 모형은 [그림 14]와 같다.



[그림 14] 순차 관계가 반영된 프로세스 흐름 모형

여기서는 메커니즘을 생략하고 표현했다. 탈착 명령(C2)은 장착과 동시에 탈착의 수행을 알리는 동기 제어(synchronization control)이기 때문에 A2와 A3은 점선에 의해 연결되고, 상하로 위치한 관계에 의해 동시 수행이 가능한 기능으로 해석된다.

4.4 프로세스 흐름 모형의 작성

기능 모형에서 기능을 나타내는 박스는 좌상에서 우하의 대각선을 따라 위치하지만, 그 위치에 따른 시간적 선후 관계의 의미는 포함하고 있지 않다([그림 12]). 하지만 활성 모드에 따른 기능의 해체에 의해 사이클을 제거하고, 변환 규칙에 의해서 화살표 유형의 분류에 따라 순차 관계를 부여하므로써 도출한 프로세스 흐름 모형에서는 전체적으로 기능들간의 순차 관계를 고려해서 표현할 수 있다([그림 14]). 예컨대, 프로세스 흐름 모형에서 왼쪽에 위치한 기능이 오른쪽에 위치한 기능보다 먼저 수행된다는 선후 관계의 의미를 분명히 부여할 수 있고, 더불어 상하 방향으로 같은 자리에 위치한 기능은 개별 기능의 수행 시간에 따라 차이가 있을 수 있지만, 상대적으로 동시 수행이 가능한 관계로 해석할 수 있다. 따라서 앞서의 변환 규칙에 따라 순서가 부여된 개별 기능들을 전체적인 순차 관계를 고려해 재배치 할 수 있다. 먼저, 기능을 노드로 하고 변환된 순차 관계를 호로 하는 네트워크를 구성한다. 이렇게 구성된 네트워크는 이 상태에서 그 자체로 전체 프로세스의 흐름을 표현하기 때문에, 기능의 수행 순서에 따른 객체의 흐름이나 제어의 흐름을 파악할 수 있다.

5. 결론 및 추후 연구 방향

본 연구를 통해서 컴퓨터통합생산시스템 설계에 있어 가장 널리 이용되고 있는 IDEF0 기능 모형을 바탕으로 프로세스 흐름 모형을 도출할 수 있는 방법을 제시했다.

도출된 프로세스 흐름 모형은 객체의 흐름과 더불어 시스템에서 수행되는 기능의 순서 관계가 묘사되고 있어, 주요 프로세스(critical process)의 정의에 의한 병목 프로세스의 개선이나 기능 모형과의 일관성 확보 등 기대할 수 있다.

한편, 프로세스 흐름 모형에 표현되는 객체의 흐름은 그 자체로 객체 상태 전이를 묘사하고 있기 때문에, 이를 바탕으로 소프트웨어의 사양 설계와 정보 모형으로의 연계 방안에 관한 연구가 요구된다.

참고문헌

[1] Boucher, T. O. and M. A. Jafari, "Design of a Factory Floor Sequence Controller from a High Level System Specification," J. of Mfg. Sys., V11, No1, pp 401-417, 1992

[2] Busby, J. S. and G. M. Williams, "The Value and Limitations of Using Process Models to Describe the Manufacturing Organization," Int. J. of Prod. Res., V31, No9, pp2179-2194, 1993

[3] Curtis, B., M. I. Kellner and J. Over, "Process Modeling," Communications of the ACM, V35, No, pp75-90, 1992

[4] Draft Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling(IDEF0), FIPS Waiver Decisions, NIST, Dec. 1993

[5] Engelke, H. et al., "Integrated Manufacturing Modeling System," IBM J. Res. Develop, V29, No4, pp343-355, 1985

[6] Liang, G. R. and H. M. Hong, "Hierarchy Transformation Method for Repetitive Manufacturing System Specification, Design, Verification and Implementation," CIMS, V7, No3, pp191-205, 1994

[7] 신기태, 제조 데이터베이스 구축을 위한 개념설계 지원 시스템에 관한 연구, 서울대학교, 박사학위논문, 1995