객체지향 생산시스템 시뮬레이터 설계 및 평가

김재만*, 김성식*

Design and Evaluation of Object-Oriented Simulator for Manufacturing System

Jae M. Kim*, Sung S, Kim*

*고려대학교 산업공학과

Abstract

생산시스템 관련 연구에서 시뮬레이션은 중요한 도구로 폭넓게 활용되어 왔다. 생산시스템을 분석하기 위한 시뮬레이션 모델은 그 기본구조가 정형화되어 있기 때문에 생산시스템 시뮬레이터 개발에 관련된 사례연구가 적지 않다. 최근 들어서는 객체지향 접근방법을 이용한 시뮬레이터가 상용화되는 추세이다.

본 연구에서는 객체지향 생산시스템 시뮬레이터를 설계 및 구현하는데 있어서 객체지향 접근방법의 잇점인 재사용성 및 확장성을 제고시킬 수 있는 방안으로 Pattern을 이용한 설계기법 및 S/W Metrics를 이용한 설계평가 방법에 대해서 논의한다. 또한 현재 개발중인 생산시스템 시뮬레이터 KU-FASIM에 이런 개념들이 실제 적용된 결과에 대해서 검토해 본다.

1. 서 론

생산시스템 관련 연구에서 시뮬레이션은 중요한 도구로 폭넓게 활용되어 왔다. 생산시스템을 분석하기 위한 시뮬레이션 모델은 기본구조는 정형화되어 있지만 실제 대상이 되는 생산시스템이나 문제의 특성에 따른 적지않은 변화요인을 가지고 있다. 이와 같은 이유로 생산시스템 시뮬레이터 개발에 객체지향 개념을 적용한 사례연구도 적지 않다.

객체지향 프로그래밍(Object-Oriented Prog-ramming; OOP) 기법은 개발된 소프트웨어의 재사용성과 유지성, 확장성 등을 제고시킬 수 있는 방안으로 최근 들어 더욱 각광받고 있다. 그러나 단순히 객체지향 언어를 사용하여 개발된 시스템이 모두 이와 같은 특성을 가지는 것은 아니다. 효과적인 객체지향 시스템의 개발은 시스템의 분석 및 설계단계에서부터 객체지향적인 개념의 적용을 필요로 한다. 그간 많은 연구자들이 다양한 객체지향 시스템 개발 방법론을 제안해 왔다. 그러나 이들 방법론들 대부분이 설계나 설계분석시 정성적인 접근방법의 비중이 높아서 개발자의 주관적인 측면이 차지하는 비중이 높았다.

최근 들어 객체지향 S/W에서 해당 S/W의 재사용성과 확장성등을 제고시킬 수 있는 방안으로 Pattern 중심의 설계기법 및 S/W Metrics를 이용한 설계평가에 대한 연구가 활발히 진행되고 있다. 이중 Pattern 중심의 설계기법은 객체지향 시스템의 설계단계에서 적용될 수 있고, S/W Metrics는 시스템의 설계가 완료된 상황에서 해당 설계의 문제점을 파악하는 단계에서 적용될 수 있다.

Pattern 중심의 설계기법은 사용중인 기존의 객체 지향 시스템에서 그 효용성이 증명된 클래스들 혹은 오브젝트들간의 관계 형태를 새로 개발하고자 하는 시스템의 설계시 그대로 적용하는 것이다. 이러한 Pattern 중심의 설계기법에 관련된 연구는 Gamma등[3]의 연구가 대표적이다.

S/W Metrics는 특정 소프트웨어의 여러가지

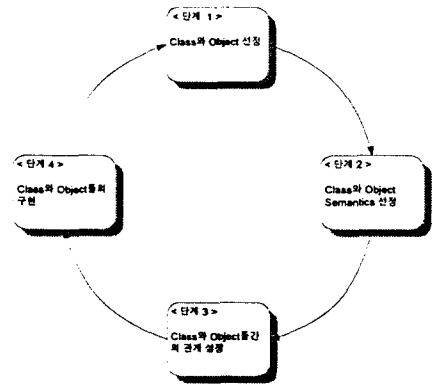
내부 특성치(interanl attribute)를 수치화된 값으로 측정하는 측정단위이다. S/W Metric은 시스템 개발자가 관심을 가지는 소프트웨어의 복잡성, 신뢰성(Reliability) 등과 같은 외부 특성치(external attribute)를 추정 혹은 평가하는 목적으로 이용된다. 소프트웨어의 재사용성 혹은 유지성은 해당 소프트웨어의 복잡성이라는 특성치와 직접 연관된다. 물론 지금까지 여러 연구자에 의해 제안된 S/W Metrics들이 대상 소프트웨어의 복잡성에 대한 완벽한 척도로서의 역할은 못했지만, 전통적인 소프트웨어의 구조적 설계 및 분석(Structured Design and Analysis; SD/A) 기법에서는 시스템의 재사용성, 유지성등을 평가하는 Coupling / Cohesion이라는 S/W Metrics가 현실적으로 폭넓게 이용되어 왔다. 최근 들어 객체지향 환경에 적합한 S/W Metrics에 대한 연구가 활발히 진행되고 있으며, 객체지향 환경에 적합한 S/W Metrics에 대한 연구는 Chidamber와 Kemerer [2]가 대표적이다.

본 연구에서는 현재 개발중인 KU-FASIM이라는 객체지향 생산시스템 시뮬레이터의 기본 설계에 적용된 Gamma등의 설계 Pattern 들과 Chidamber와 Kemerer의 S/W Metrics를 적용한 평가 결과를 제시하고 그 적용결과를 검토하여 보았다. KU-FASIM은 대상 생산시스템에서 주문의 납기결정, 주문투입, 작업순서 결정 등의 방식들의 효과를 분석하기 위한 객체지향 시뮬레이터이며, 현재 기본적인 Class Library가 구현되어 있다.

2. 객체지향 시스템 설계 및 구현

시스템의 개발은 크게 분석(Analysis)과 설계 및 구현(Design and Implementation)의 두 단계로 이루어 진다. 이중 분석은 개발 대상 시스템을 정의하고 필요기능을 파악하는 단계이며, 설계 및 구현은 분석단계에서 정의된 시스템 기능을 실제 개발환경에 적합한 구조로 변환하여 실제 물리적인 시스템으로 개발하는 단계이다. Pattern중심의 설계기법 및 시스템의 복잡성을 평가하기 위한 S/W Metric은 이중 설계 및 구현 단계에 적용되게 된다.

다양한 객체지향 시스템 설계 방법론들의 효시가 된 Booch[2]의 방법론은 객체지향 시스템 설계 및 구현을 다음 그림 1과 같은 4단계의 반복적인 과정으로 정의하였다.



[그림 1] 객체지향 시스템 설계및 구현 절차

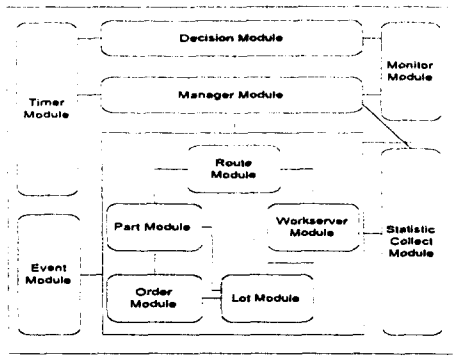
이중 Pattern 중심의 설계기법은 단계 2와 단계 3에서 주로 적용되게 된다.

대부분의 객체지향 시스템 설계 방법론들과 마찬가지로 그림 1의 모든 4가지 단계에서는 시스템 개발자의 주관이 많이 반영되게 된다. 따라서 각 단계의 결과물을 평가하기 위한 척도가 필요하다. 위의 4단계중 본 연구의 대상이 되는 S/W Metrics를 주된 적용대상이 되는 것은 단계 3의 클래스와 오브젝트간의 관계설정 단계이다. 단계 3의 결과로 Class Diagram과 Object Diagrm이 결정되는데, 이들 도표는 객체지향 시스템의 상세설계서가 된다. Booch[2]는 Coupling과 Cohesion 등을 단계 3의 주된 평가항목으로 제안하였으나 그 구체적인 평가방법은 언급하지 않았다.

3. 객체지향 생산시스템 시뮬레이터 - KU-FASIM

현재 개발중인 객체지향 생산시스템 시뮬레이터 KU-FASIM은 생산시스템을 대상으로 각종 생산계획 및 관리절차의 효과를 측정하기 위한 목적으로 개발중인 객체지향 시뮬레이터이다. 현재 KU-FASIM은 기본 Class의 설계 및 구현이 완료되었으며 통합개발환경은 현재 개발중이다. 그림 2는 KU-FASIM의 개념적인 클래스 모듈 구성도이다.

그림 2의 Decision Module에는 생산시스템에



[그림 2] KU-FASIM의 개념적인 클래스모듈구성

서의 각종 의사결정 관련 Class의 표준 인터페이스가 정의되어 있고, 현재 많은 Algorithm이 Class 라이브러리화되어 있다. KU-FASIM에서 대상으로 하는 의사결정 내용은 다음의 6가지이다.

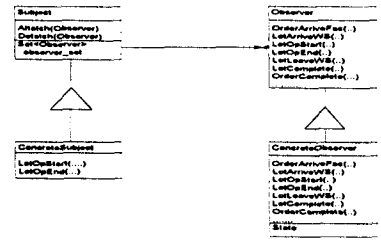
- 주문의 내부납기 결정
- 로트 크기 결정
- 로트 현장 투입 결정
- Dispathcing
- 대체경로의 결정
- 외주 및 잔업 결정

생산 시스템 시뮬레이션 모델링 작업중 대상 시스템에 따라 가장 변동이 심한 부분중 하나가 의사결정 관련 알고리즘과 연관된 부분이다. 의사결정 관련 알고리즘이 변경됨에 따라 동일한 공장이라 할지라도 시뮬레이션 모델의 재코딩이 요구될 수 있는데 그 변경되는 부분은 크게

1) 해당 알고리즘에 필요한 정보를 수집하는 부분과

2) 알고리즘 자체의 두가지이다.

이중 특히 해당 알고리즘에 필요한 정보를 수집하는 부분이 변경될 경우 다른 클래스 모듈에도 파급효과를 미치게 된다. KU-FASIM에서는 이와 같은 알고리즘 변화에 따른 파급효과를 최소화하기 위하여 Gamma 등[3]의 Observer - Subject 설계 Pattern을 변형시켜 적용함으로써 알고리즘 변경이나 추후 확장에 대비하였다. 알고리즘에 필요한 정보를 수집하는 클래스(생산현장 관련 클래스)와 정보를 제공하는 클래스(Monitoring 클래스)의 인터페이스를 표준화하여 알고리즘의 변경시 다른 클래스 오브젝트의 영향을 최소화하였다. KU-FASIM에서 다음 그림 3은 이와 같은 Observer-Subject의 Class Diagram을 도시한 것이다.



[그림 3] Observer-Subject Class Diagram

실제 KU-FASIM 개발과정에서 각종 의사결정 알고리즘에 관련된 Class 개발시 그림 3의 Observer-Subject Pattern이 새로운 알고리즘 도입에 따른 변화의 파급범위를 최소화하는데 유용하게 이용되었다.

4. 객체지향 S/W Metrics

- Chidamber & Kemerer's OO Metrics Suite

전통적인 소프트웨어의 구조적 설계 및 분석 기법에서는 Coupling/Cohesion이 설계 혹은 개발된 S/W의 모듈 분할(Decomposition)의 적합성을 평가하는 방법으로 폭넓게 사용되었고, 개발자들은 모듈간의 낮은 Coupling과 높은 Cohesion을 가지는 소프트웨어를 재사용성이나 유지성이 높다고 평가하는게 일반적이었다. 전통적인 접근방법에서는 전체 시스템을 개별 모듈로 분할하는 방법에 대한 뚜렷한 지침이 없는데 반해 객체지향 환경에서는 클래스와 오브젝트라는 명확한 하부객체가 존재한다. 이와 같은 이유로 그간 Coupling과 Cohesion을 대표되는 기존의 S/W Metrics이 객체지향 환경에서 직접 적용되지는 못했다.

Chidamber와 Kemerer[2]는 대부분의 S/W Metrics들이 이론적인 근거없이 제안되었음을 지적하고, Weyuker[7]가 제시한 6가지 기준을 통해 검증된 객체지향 S/W Metrics를 제안하였다. Chidamber와 Kemerer[2]가 제안한 객체지향 S/W Metrics는 각 개별 클래스 단위로 측정

하게 되어 있는 6가지의 Metrics로 구성된다. 이들 Metrics는 Booch의 Class Diagram과 같은 정적인 설계(static design)를 대상으로 한다. 개별 Metric의 정의는 다음과 같다.

1) WMC(Weighted Methods per Class)
임의의 클래스 C에 정의된 Method의 가중치의 합

2) DIT(Depth of Inheritance)
임의의 클래스 C의 최대 상위 클래스(super class)의 수

3) NOC(Number of Children)
임의의 클래스 C를 상속받는 인접한 하부 클래스(sub class)의 수

4) CBO(Coupling Between Object class)
임의의 클래스 C와 연결된 다른 Class의 수

5) RFC(Response For a Class)
임의의 클래스 C에 정의된 Method와 이들 Method에서 호출되는 Method 합집합의 Cardinality

6) LCOM(Lack of Cohesion in Method)
임의의 클래스 C에서 Instance Variable을 공유하는 Method들을 원소로 하는 서로 소인 집합의 수

이중 6번째 Metric인 LCOM은 Hitz와 Montazeri[5]가 제안한 수정된 정의를 이용하였다. 위의 6가지 Metrics중 CBO와 LCOM은 과거의 Coupling /Cohesion을 객체지향 환경에 적합하도록 수정한 것이고, 다른 4가지는 소프트웨어의 복잡성과의 연관을 직관적으로 연관지을 수 있는 측정치이다. 본연구에서는 위의 6가지 Metrics를 현재 개발중인 객체지향 생산시스템 시뮬레이터 KU-FASIM에 적용하였다.

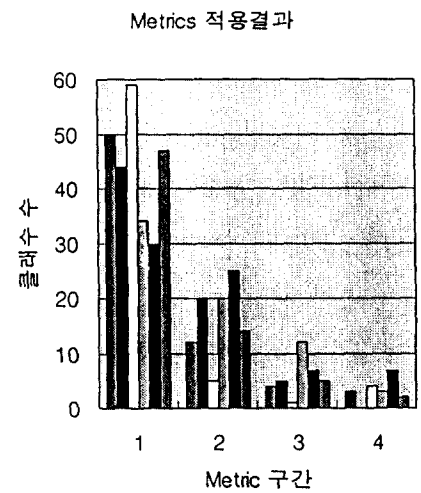
5. S/W Metrics 적용 결과 및 분석

그림 4는 Chidamber와 Kemerer의 S/W Metrics를 KU-FASIM에 적용한 결과를 요약한 것이다. 그림 4에서 볼 수 있듯이 6가지 S/W Metrics에 대해 대부분의 클래스들이 낮은 값을 가진다.

KU-FASIM에서는 클래스간의 관계 설정시 상속보다는 포함(Containment)관계를 많이 이용하였기 때문에 DIT와 NOC 등 상속 관련 Metric의 값이 낮게 나왔다.

LCOM의 경우는 측정치가 1을 초과하는 클래스가 적지 않다. 높은 LCOM 값을 가지는 클래스는 Jabocson등[6]이 정의한 개체형 오브젝트(Entity Object)에 해당되는 클래스에서 흔히 발견되었다. 개체형 오브젝트에서는 새로운 하부 클래스를 추가시키기 전에는 LCOM 수치를 낮출 방법이 없다. 새로운 하부 클래스를 추가시키면 대신 CBO와 RFC가 증가된다.

CBO와 RFC는 비슷한 결과를 나타냈고, 각 클래스의 수치를 비교하면 높은 상관관계를 가진다. Hitz와 Montazari[4]는 클래스간의 Coupling을 표현할 때 Coupling의 강도를 함께 고려해야 한다고 지적하였다. 이런 관점에서 RFC는 독립적으로 쓰이기 보다는 CBO를 보완하는 수치로 이용되는 것이 바람직하다. CBO와 RFC가 높게 나온 클래스는 대부분 의사결정 관련 클래스와 모니터링 관련 클래스이다.



[그림 4] S/W Metrics의 적용결과

6. 결 론

본 연구에서는 Pattern 중심의 설계기법 및 객체지향 환경의 대표적인 S/W Metrics인 Chidamber와 Kemerer의 Metrics를 현재 개발중이 생산시스템 시뮬레이터 KU-FASIM에 적용하여 보았다.

KU-FASIM의 기본 설계에 적용된 Gamma등의

Subject-Observer Pattern은 KU-FASIM의 개발 과정에서 다양한 알고리즘의 구현시 다른 클래스에 대한 변화의 파급효과를 최소화하는데 유용하게 이용되었다.

Chidamber와 Kemerer의 S/W Metrics는 객체지향 시스템이 가져야할 일반적인 특성을 비교적 잘 반영하고 있으나 아직까지는 상세한 수준의 설계평가에는 부적합하다. 더욱이 이들 Metrics중 CBO와 LCOM의 비중이 상당히 큰데 실제 적용해 본 결과 LCOM과 CBO의 경우는 수정 및 보완이 필요하다.

또한 현재 객체지향 환경의 S/W Metrics는 대부분 개별 클래스를 대상으로 측정하는데 클래스가 과거 SD/A에서 Coupling과 Cohesion 측정단위였던 모듈과는 차이가 있다. 이점은 LCOM과 CBO를 보완하는데 있어서 반영되어야 할 것이다.

참 고 문 헌

[1] Booch,G., *"Object-Oriented Analysis and Design with Applications"*, 2nd Ed., 1994, Addison-Wesley

[2] Chidamber, S.R., Kemerer, C.F., "A Metrics Suite for Object Oriented Design", *IEEE Tran. on Software Engineering*, Vol.20, No. 6, 1994

[3] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *"Design Patterns"*, 1995, Addison-Wesley

[4] Hitz. M., Montazeri, B., "Measuring Coupling and Cohesion in Object-Oriented Systems", *Proc. Int. Symposium on Applied Corporate Computing*, 1995

[5] Hitz. M., Montazeri, B., "Chidamber & Kemerer's Metrics Suite: A Measurement Theory Perspective", *IEEE Tans. on Software Engineering*, Vol.22, No. 4, 1996

[6] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., *"Object-Oriented Softwa-re Engineering"*, 1992, Addison-Wesley

[7] Weyuker, E.J., "Evaluating Software Complexity Measures", *IEEE Trans. on Software Engineering*, Vol. 14, No. 9, 1988