제조 공정의 설계 자동화 시스템을 시스템 이론에 의거한 소프트웨어 설계 방법론 적용

안영숙*·조대호*
Ahn, Young Sook and Cho,Tae Ho

Abstract

인터페이스 시스템은 많은 장치들을 빈번히 제어하고, 비동기적으로 사건의 입력 스트림을 보내야 하며, 사용자의 행동과 시스템 반응 사이에 지체감이 없도록 보장해야 한다는 요구 사항을 만족해야 하는 특징이 있다. 결과적으로 인터페이스 시스템은 빈번한 기능 추가 또는 변경에 의한 수정 때문에 자주 프로토타입으로 만들어지고 반복적으로 수정되어되야한다. 본 논문은 제조공정의 설계 자동화 시스템인 GDS(Grating automatic Drawing System)를 계발함에 있어서 인터페이스 시스템 설계 및 구현 과정에서 있을수 있는 설계 변경 및 이에 따른 다른 변경 요인들을 정확하게 파악하고, 구현상의 변경으로 인한 전체 시스템의 영향 등을 체계적으로 정립한 소프트웨어 설계 방법론을 적용함으로 해서 소프트웨어 설계 시 또는 설계 변경 후 인터페이스 시스템의 중요한 동적 특성을 미리 파악할수 있다.

1. 서 론

소프트웨어 공학에서 인터페이스는 사용자와 응용 시스템간의 상호작용을 위한 대화의 수단으로 정의 되어 사용되고 있다. 사용자가 응용 시스템에 입력을 주고, 응용 프로그램이 사용자에게 결과를 보여준다.

소프트웨어 시스템의 기능이 복잡해지고, 사용자의 계층이 다양해짐에 따라, 사용자 인터페이스의 중요성이 크게 강조되고 있다. 소프트웨어 개발시 좋은 사용자 인터페이스를 설계하는 것은 어려운 작업이다. 인터페이스 소프트웨어는 많은 장치들을 빈번히 제어해야 하고, 비동기적으로 사건의 입력스트림들을 보내야 하며 사용자의 행동과 시스템 반응 사이에 지체감이 없도록 보장해야 한다는 요구 사항을 만족해야 하는 특성 때문에 자주 프로토타입으로 만들어지고 반복적으로 수정되어야 한다[6]. 그러나 비교적 수행 절차에 필요한 알고리즘이 단순하고 처리 과정이 복잡하지 않으므로 사상 이론을 적용하여 자동 시스템을 구현하는 것이 용이하다.

그러므로 본 논문에서는 제조공정의 설계 자동

화 시스템인 GDS를 계발함에 있어서 인터페이스 시스템의 설계 및 구현 과정에서 있을 수 있는 설계 변경 및 이에 따른 다른 변경 요인들을 정확하게 파악하고, 구현상의 변경으로 인한 전체 시스템의 영향 등을 체계적으로 정립한 소프트웨어 설계 방법론으로의 적용 방법을 제시하였다.

2. 배경 이론

이 장에서는 시스템 명세의 계층 구조를 설계하는데 적용되는 시스템 이론, DEVS 방법론, 객체지향 프로그래밍에 대한 기본 개념을 살펴보면 다음과 같다.

- 객체지향형 프로그래밍 :
프로그램을 객체로 분해하고, 그들 사이의 관계를 설정하는 가정으로 객체를 중심으로 모듈을 분해하는 기법이다[8].

- 시스템 명세(System Specification) :
비형식적인 모델(informal model)을 컴포넌트(component)와 컴포넌트간의 상호작용(component interaction), 영향도(influence diagram)등에 의해 형식적인 5계층으로 표현한다[1]. 5계층 중에서 2,3 계층을 적용하여 설계하였다.

- 시스템 구조관계(System Morphism) :
같은 레벨에서 한 시스템과 다른 시스템과의 관계를 정의한다[1].

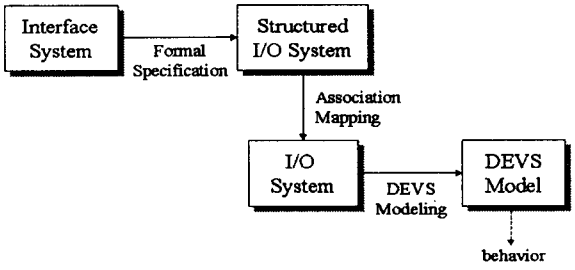
- DEVS(Discrete EVent system Specification) 방법론 :
시뮬레이션 모델링은 Zeigler에 의해 정립되어 이산사건 모델들의 계층 구조적 모듈화 방법을 제공해주는 형식론인 DEVS (Discrete Event System Specification)에 따른다. DEVS는 시스템이 일반적으로 갖는 특성들을 정의하여 시스템을 모델링할 수 있는 프레임웍(Framework)을 제공하며 형식론을 채택하여 시스템을 관찰하고자 하는 초점으로 추상화하여 모델링함으로써 모델과 실제 문제와의 관련성을 높일 수 있다. DEVS의 기본(Basic) 모델은 다음과 같은 항들로 명세할 수 있다[3,4,5] :

$$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : 입력 사건(Event)들의 집합
S : 사건의 변화에 따라 가질 수 있는 상태들의 집합
Y : 출력 사건들의 집합
δ_{int} : $S \rightarrow S$, 내부 상태전이(State transition) 함수
δ_{ext} : $Q \times X \rightarrow S$, 외부 상태전이 함수
λ : $S \rightarrow Y$, 출력 함수
ta : $S \rightarrow R^{+\infty}_{0}$, 시간 진행 함수
where $Q = \{(s,e) \mid s \in S, 0 \le e \le ta(s)\}$
e: 최근의 상태 전이 이후로 경과된 시간

3. 시스템 명세의 계층 구조

인터페이스 시스템을 <그림 1>과 같이 시스템 이론에서 정의하는 입출력 시스템 레벨(I/O System Level)의 요소들로 표현하고 다시 입출력 시스템 레벨을 구조적 입출력 레벨(Structured I/O System)로 변환한다. 이를 다시 DEVS 모델로 재구성하여 DEVS 시뮬레이션 환경에서 제공하는 시뮬레이터를 통하여 대상 시스템의 중요한 동적 특성을 소프트웨어 설계 시 또는 설계 변경 후 미리 파악할 수 있도록 한다.



<그림 1> 시스템 명세와 구조 관계

<그림 1>과 같은 4가지 시스템들의 사상을 통해 상호 의존성을 추적하는 것이 이 시스템 설계의 목표이다.

3.1 시스템 명세

본 논문에서는 시스템 명세를 인터페이스 시스

템, 구조적 입출력 시스템, 입출력 시스템, DEVS 모델 등 4가지 시스템으로 구분하였다. 각각의 내용과 적용된 예를 살펴보면 다음과 같다

- 실시스템((Real System) : 인터페이스 시스템으로서 소프트웨어 공학적인 토대와 객체지향형 프로그래밍 사고틀에 의해서 구성되어 있고, 모델에 적용된 예는 <표 1>와 같다.

- 구조적 입출력 시스템(Structured I/O System) : 인터페이스 시스템을 원시 집합과 함수로 추상화하여 표현한 구조적 모델[1]로서 정의 및 적용예는<표2>과 같다.

- 입출력 시스템(I/O System) :
입력 세그먼트에 의한 상태전이와 특정 상태에서 발생하는 출력 함수 등의 추상화된 집합과 함수로 구성된 시스템[1,2] 으로서 정의 및 적용예는 <표 3>과 같다.

- DEVS 모델(DEVS Model) :
이산사건 시스템을 모듈화하고 계층적으로 잘 기술하여 동적특성을 DEVS 시뮬레이션 환경에서 시뮬레이션할 수 있는 모델로서 정의 및 적용예는 <표 4>와 같다.

3.2 구조 관계

3.1절에서 살펴본 각 시스템간의 관계를 정의해 주는 것이 구조 관계이다. 형식 명세에 의해 인터페이스 시스템을 구조적 입출력 시스템의 요소들로 표현하고, 이를 관계 사상에 의해 입출력 시스템으로 변환한 다음 DEVS 모델링에 의해 DEVS 모델로 재구성한다. 각각의 내용과 적용된 예를 살펴보면 다음과 같다.

3.2.1 형식 명세 (Formal Specification)

인터페이스 시스템을 구조적 입출력 시스템의 요소들로 추상화하여 표현한다. <표 5>을 보면 인터페이스 시스템과 구조적 입출력 시스템의 구조 관계를 알 수 있다.

3.2.2 관계 사상 (Association Mapping)

구조적 입출력 시스템에서 입출력 시스템으로의 상태전이와 상태전이함수를 <표 6>에서 볼 수 있다.

3.2.3 DEVS 모델링 (DEVS Modeling)

입출력 시스템에서 <표 7>와 같이 DEVS 형식론을 적용하여 DEVS 모델로 모델링하였다.

<표 1> 인터페이스 시스템의 클래스와 멤버함수

클래스	멤버함수 (생성자, 소멸자 함수는 생략)
GDSApp	OnInitialize(), InitAction() ...
InitDWGDlg	OnInitialUpdate(), OnPrint(), OnEdit(),OnDelete(), OnDblSpred(), OnOK ....,
CFrameGenView	OnInitialUpdate(),ColunmnLocate(), CreateDWGInfo(), RecogSupport(), OnDelete(), OnOK(), ...
GratGenView	OnInitialUpdate(),CircularLocate(), RectanglularLocate(),RecGAInfo(), CirGAInfo(), OnOK(), ...
DraDataGenView	OnInitialUpdate(), EXcuteID(), ExcuteIPD(), EXcuteBML(), OnOK(),
...	...

<표 4> DEVS 모델의 정의 및 적용 예

정의	M (model) = <X,S,Y, δ_{int}, δ_{ext}, λ ,ta> where X: set of external (input) event types S: sequential state set Y: output set δ_{int} : S→S, the internal transition function δ_{ext} : Q×X→S, the external transition function ta : S→$R^{+\infty}_{0}$, the time advance function λ : S→Y, the output function where Q={(s,e)|s∈S, 0≤e≤ta(s)}
적용	M = <X,S,Y, δ_{int}, δ_{ext}, λ ,ta> where X = {DWGInfo, MatType, ColumnGen, ..., ReDWGInfo, ReFrame, ReGrating, Appl}, x∈X S = {Passive,DIA-1,DIA-2,DFA-1, ..., Processed}, s∈S Y = {DWGInfo, MatType, ColumnGen, ..., ReDWGInfo, ReFrame, ReGrating, Appl}, y∈Y δ_{int}(s e x) = s + x λ(s) = { $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_7$ }

<표 2> 구조적 입출력시스템의 정의 및 적용예

정의	M (machine) = < Q, δ > where Q : the cross product of the component states sets δ : the cross product of component functions structure of M : (D,{Q_α},{I_α},{δ_α}) where D (coordinates) {$Q_\alpha \mid \alpha \in D$} (range sets) {$I_\alpha \mid \alpha \in D, I_\alpha \subseteq D$} (influencers) {$\delta_\alpha \mid \alpha \in D, \delta_\alpha : \times_{\beta \in I_\alpha} Q_\beta \rightarrow Q_\alpha$} (local transition function) such that $Q \subseteq \times_{\alpha \in D} Q_\alpha$, and $\delta = \times_{\alpha \in D} \delta_\alpha$ ° proj I_α restricted to Q
적용	M = < Q, δ > where Q = { P_{O1}, P_{O2}, P_{O3}, ... , P_{O7} } δ = { O1, O2, O3, ... , O7 } (D,{Q_α},{I_α},{δ_α}) where D = { O1, O2, O3, ... , O7 } {Q_α} = { P_{O1}, P_{O2}, P_{O3}, ... , P_{O7}} such that P_{O1} = ($P_{O1\text{-}1}$, $P_{O1\text{-}2}$, $P_{O1\text{-}3}$), P_{O2} = ($P_{O2\text{-}1}$, $P_{O2\text{-}2}$), P_{O3} = ($P_{O3\text{-}1}$, $P_{O3\text{-}2}$), ... , and P_{O7} = ($P_{O7\text{-}0}$, $P_{O7\text{-}1}$, $P_{O7\text{-}2}$, ... , $P_{O7\text{-}17}$) {I_α} = { I_{O1}, I_{O2}, I_{O3} , ... , I_{O7} } such that I_{O1} = {O1,O7}, I_{O2} = {O1,O2,O7}, I_{O3} = {O1,O2,O3,O7}, ... , and I_{O7} = {O7} {δ_α} = { δ_{O1}, δ_{O2}, δ_{O3}, ... , δ_{O7} } such that δ_{O1} : $P_{O1} \times P_{O7} \rightarrow P_{O1}$, δ_{O2} : $P_{O1} \times P_{O2} \times P_{O7} \rightarrow P_{O2}$, δ_{O3} : $P_{O2} \times P_{O2} \times P_{O3} \times P_{O7} \rightarrow P_{O3}$, ... , and δ_{O7} : $P_{O7} \rightarrow P_{O7}$

<표 5> 형식 명세의 정의 및 적용 예

정의	객체지향 프로그래밍에 의해 구성된 인터페이스 시스템과 구조적 입출력 시스템 (D,{Q_α},{I_α},{δ_α})과의 구조 관계를 보면, - 인터페이스 시스템의 각 클래스마다 오브젝트를 생성하여 구조적 입출력 시스템의 D(coordinate)로 정의한다. - 각 클래스는 상태변수에 의해 오브젝트의 상태집합 Q_α로 표현된다.
적용	D : InitDWGDlg × FrameGenView × ... × GDSApp ⇒ O1 × O2 × ... × O7 Q : InitDWGDlg.p1 × FrameGenView.p2 × ... × GDSApp.p7 ⇒ P_{O1} × P_{O2} × P_{O3} × ... × P_{O7}

<표 3> 입출력 시스템의 정의 및 적용 예

정의	S (system) = < T,X,Ω,Q,Y,δ,λ > where T: time base X : the input value set Ω: the input segment set (a subset of (X,T)) Q: the state set Y: the output value set δ: the state transition function λ: the output function
적용	S = < T,X,Ω,Q,Y,δ,λ > where T = Integer, λ = { λ_1, λ_2, λ_3, ... , λ_7 } X = {DWGInfo, MatType, ColumnGen, ..., ReDWGInfo, ReFrame, ReGrating, Appl} Ω = { $\omega \mid \omega$:<0,1> → X } Q = {q_0, q_1, q_2, q_3, ... , q_{17}} such that q_0=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}0}$) q_1=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,.. ,$P_{O7\text{-}1}$) q_2=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}2}$) ... , and q_{25}=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$, ...,$P_{O7\text{-}0}$) Y = {DWGInfo, MatType, ColumnGen, ..., ReDWGInfo, ReFrame, ReGrating, Appl} δ(q)= (δ_{O1} ° proj I_{O1}(q), δ_{O2} ° proj I_{O2}(q), δ_{O3} ° proj I_{O3}(q), ... , δ_{O7} ° proj I_{O7}(q)), q ∈ { q_0, q_1, q_2, q_3, ... , q_{31} }

<표 6> 관계 사상의 정의 및 적용 예

정의	M = <Q, δ >은 다음과 같은 구조를 갖는다. D (coordinates) {$Q\alpha \mid \alpha \in D$}(range sets) {$I\alpha \mid \alpha \in D, I\alpha \subseteq D$}(influencers), {$\delta\alpha \mid \alpha \in D, \delta\alpha$: $\times_{\beta \in I\alpha} Q\beta \rightarrow Q\alpha$ } (local transition functions) 구조적 입출력 시스템과의 관계는 다음과 같이 정의 된다. $Q \subseteq \times_{\alpha \in D} Q\alpha$, $\delta = \times_{\alpha \in D} \delta\alpha$ ° proj$I\alpha$ restricted to Q
적용	Q: {P_{O1}, P_{O2}, P_{O3}, ... , P_{O7}} ⇒ {q0, q1, q2, q3, ... , q_{17}} such that q_0=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}0}$,) q_1=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}1}$) q_2=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}2}$), ... , and q_{25}=($P_{O1\text{-}1}$,$P_{O2\text{-}1}$,$P_{O3\text{-}1}$,$P_{O4\text{-}1}$,$P_{O5\text{-}1}$,$P_{O6\text{-}1}$,$P_{O7\text{-}0}$) δ: δ(q)= (δ_{O1} ° proj I_{O1}(q), δ_{O2} ° proj I_{O2}(q), δ_{O3} ° proj I_{O3}(q), ... , δ_{O7} ° proj I_{O7}(q)) , q ∈ { q_0, q_1, q_2, q_3, ... , q_{17} }

<표 7> DEVS 모델링의 정의 및 적용 예

정의	M (model) = <X_M, S_M, Y_M, δ_{int}, δ_{ext}, λ_M, ta> where X_M = X' S_M = S' Y_M = Y' δ_{int} : $Q_M \times \phi$ -> Q_M δ_{ext} : $Q_M \times \Omega'$ -> Q_M, where Q_M ={(s,e)|s∈S_M, 0≤e≤ta(s)} ta : S->Real λ_M : λ'
적용	S : { q_0, q_1, q_2, q_3, ... , q_{25} } ⇒ {PASSIVE,DIA-1,DIA-2,DIA-3, ... ,PROC} δ : δ(q) ⇒ δ_{ext}, q ∈ { q_0, q_1, q_2, q_3, ... , q_{25} }

4. 결 론

인터페이스 시스템은 빈번한 기능 추가 또는 변경에 의한 수정 때문에 자주 프로토타입으로 만들어지고 반복적으로 수정되어야 한다. 그러므로 소프트웨어의 규모가 커지고 복잡해질수록 수정하는 일이 어려워진다[6,9,13]. 그러나 비교적 수행절차에 필요한 알고리즘이 단순하고 처리 과정이 복잡하지 않으므로 사상 이론을 적용하여 자동 시스템을 구현하는 것이 용이하다는 특성을 이용하여 시스템 이론에서 정의하는 구조관계를 통해 구조적 입출력 시스템 레벨로 변환한 다음 이를 다시 DEVS 모델로 재구성하였다.

추후 연구되어야 할 과제는 DEVS 시뮬레이터를 통해 소프트웨어 시스템을 개발할 때마다 반복적으로 되풀이되는 수정에 의한 번거로움을 덜어줄 뿐만 아니라, 전체 시스템의 구조나 흐름을 파악하여 변경된 부분을 자동으로 추적하고, 그 역의 기능도 가능하도록 구현하는 것이다. 그리고 현재는 시간 개념을 배제한 인터페이스 시스템이라는 특정 도메인을 대상으로 설계 방법론을 제시하였으나, 향후 시간 의존적인 인터페이스 시스템, 더 나아가서 인터페이스 시스템뿐만 아니라 확장된 일반 소프트웨어에도 적용 가능한 체계적인 소프트웨어 설계 방법론을 정립하기 위한 연구가 이루어wu야 할 것이다.

참고 문헌

[1] Bernard.P.Zeigler, Theory of Modelling and Simulation, John Wiley, NY, USA, 1976

[2] L.Padulo and M.A.Arvbib, System Theory, Suders, Philadelpia, Pa, 1974

[3] Bernard.P.Zeigler, Multifacetted Modelling and Discrete Event Simulation Orlando, FL, USA: Academic, 1984

[4] Bernard.P.Zeigler, Object-Oriented Simulation with Hierarchical, Modular Models, San Diego, CA, USA: Academic Press, 1990

[5] 조대호, 이칠기, "계층의 구조를 갖는 시뮬레이션 모델에 있어서 단계적 접근을 위한 모델 연결 방법론과 그 적용 예", 한국시뮬레이션학회 논문지, Vol 5, No 2, 1996

[6] 김상근, 객체지향 기술에 의한 사용자 인터페이스 구축기 개발, 중앙대학교 컴퓨터공학과 소프트웨어 엔지니어링 석사학위 청구논문, 1995.12

[7] 김덕현, 박성주, "확장된 객체지향 데이터 모델을 이용한 소프트웨어 변경 관리 시스템", 정보과학회 논문지, Vol 22, No 2, pp.249-260, 1995

[8] 허계범, 최영근, 객체지향 소프트웨어 공학, 한국실리콘, 1995

[9] 우치수외, 사용자 인터페이스, 영지문화사, 1994

[10] 이경환외, 소프트웨어공학, 청문각, 1993

[11] 이경환외, 소프트웨어 재이용 시스템 개발(II), 연구보고서, 과학기술처, 1993

[12] G. Booch, Object-Oriented Design, Bengamin/Cummings Publishing, 1989

[13] R. Keller, M. Cameron, R. Taylor, D. Troup, "User Interface Development and Software Environments: The Chiron-1 System", Proceedings of the 12th ICSE, pp. 208-218, 1991

[14] A. Marcus, Graphic Design for Electronic Documents and User Interfaces, Addison-Wesley Publishing Company Inc., 1992

[15] B. Myers, "User-Interface Tools: Introduction and Survey", IEEE Software, pp. 15-23, January 1989

[16] J. Raumbaugh et al., Object-Orented Modeling and Design Prentice-hall, Inc., 1991

[17] S. Rimmer, Graphical User Interface Programming, WINDCREST/McGRAW-HILL, 1991

[18] S. Shaler and S. Meller, Object-Oriented Systems Analysis: Modeling the World in Data, Englewood, Cliffs, NJ: Yourdon Press, 1988