내장형 시스템의 명세를 위한 컴포넌트 지원 Statechart 도구 구현

박홍진[0] 김남규 천경아 김영찬
중앙대학교 컴퓨터공학과
{hjpark, ssgyu, amie, yckim}@sslab.cse.cau.ac.kr

The implementation of component based statechart tool for specification of embedded system

Hong-Jin Park[0] Nam-Gyu Kim Kyung-Ah Chun Young-Chan Kim
Dept. of Computer Science and Engineering, Chung-Ang University

요 약

최근의 내장형 시스템은 점점 대형화되고 복잡성이 증가하고 있다. 또한 시스템의 생명주기는 매우 빠른 속도로 단축되고 있는 추세이다. 이에 따라 발생될 수 있는 에러율이 증가하고 있으며 이를 해결하기 위해 많은 비용과 시간에 소비될 수 있다. 따라서, 시스템 설계단계부터 신뢰성과 안정성을 보장하며 신속하게 명세를 표현할 수 있는 정형 명세 기법이 필요하다.

본 논문은 신속하면서 견고한 정형 명세 개발을 위해 컴포넌트를 지원하는 Statechart 도구를 구현한다. 본 논문에서 개발된 도구를 이용하면 이미 작성된 정형 명세를 최대한 재사용함으로 새로운 시스템 설계의 개발시간을 단축시킬 수 있는 장점을 지닌다.

1. 서론

가전제품, 마이크로 프로세서 및 주문형 반도체 등에는 매우 정밀한 내장형 시스템이 사용되고 있다. 내장형 시스템을 이용한 응용 제품의 생명 주기는 매우 짧아지고 있고 있으며, 새로운 제품 생산 속도는 30%이상 빨라지고 있다. 시스템의 복잡성도 3년마다 2배정도 증가하고 있다[1]. 복잡성이 증가함에 따라 그에 따른 에러도 점점 증가하며 단순한 에러라도 이를 해결하기 위해서는 많은 비용과 시간에 소비될 수 있다(예를 들어, Intel Pentium FDIV Problem).

복잡성이 존재하는 내장형 시스템을 신뢰할 수 있고 안정성을 보장받기 위해 시스템 설계단계부터 보다 정확한 기능적 명세가 무엇보다도 필요하며 현재 이러한 연구는 활발히 진행 중이다. Statechart는 다른 명세 기법과 달리 그림으로써 시스템을 명세하기 때문에 보다 명확하고, 쉽게 이해될 수 있으며, 내장형 시스템 명세와 가상 프로토타입핑에서 널리 사용되고 있는 명세 기법이다[2]. 또한, Statechart는 국제 표준화 기구인 OMG(Object Management Group)에서 객체지향 모델링 언어(UML)의 일부분으로 이용되고 있다[3].

그러나, 기존의 Statechart 지원하는 도구는 이미 Statechart로 작성된 정형명세를 재사용 할 수 있는 방법을 효율적으로 제공하지 못하고 있어서 새로운 시스템 정형 명세 개발에 있어 불필요한 투자와 시간이 소요될 수 있다. 시장적기진입이 새로운 제품의 성패를 좌우하는 내장형 시스템에서는 무엇보다도 빠른 시간 내에 개발 될 수 있도록 새로운 정형 명세 도구가 필요하다.

본 논문은 목적은 신속하면서 견고한 정형 명세를 개발하기 위해 컴포넌트를 지원하는 Statechart 도구 구현이다 . 본 논문의 구성은 다음과 같다. 2장에서는 기반 연구로 Statechart와 컴포넌트에 대해 기술한다. 3장에서는 컴포넌트 지원 Statechart 도구를 설계하며, 4장에서는 예제를 통해 구현된 도구를 설명한다. 마지막으로 5장에서는 결론을 맺는다.

2. 기반 연구

2.1 Statechart

Statechart는 1986년 David Harel에 의해 소개가 된 것으로 이벤트(event) 지향적인 복잡한 반응 시스템(reactive systems)의 행위를 명세하기 위한 시작적 정형 명세 언어이다[4][5].

Statechart는 상태 전이 다이어그램(state transition diagram, STD)에 계층성(hierarchy), 동시성(concurrency), broadcast communication등의 개념이 추가되어, 기존의 상태를 중심으로 하는 다이어그램 방식 언어들이 갖는 시스템 표현의 단점을 극복한 기법이다. 객체지향 모델링 언어의 국제 표준인 UML에서 일부분으로 이용되고 있는 Statechart는 내장형 시스템 설계나 가상 프로토타입핑에서 널리 사용되고 있는 명세 기법이다. 그러나, 기존의 Statechart를 지원하는 도구에서는 이미 작성된 정형명세를 효율적으로 재사용할 수 있는 방법을 제공하지 못함으로써 새로운 정형 명세 개발에 있어서 중복된 투자

본 논문은 한국과학재단 특정기초연구의 지원을 받음
(과제번호 : 1999-2-303-008-3)

와 시간이 요구될 수 있다.

2.2 컴포넌트

컴포넌트는 소프트웨어 재사용 통해 소프트웨어 생산성을 향상시키는 분야로써 많은 연구가 진행중인 분야이다. 컴포넌트는 시스템의 부분으로서 분리되어 수행될 수 있는 소프트웨어 단위를 의미한다. 컴포넌트를 이용하면 소프트웨어를 부품처럼 조립할 수 있기 때문에 빠른 시간 내에 소프트웨어를 개발할 수 있는 장점을 지닌다. 컴포넌트의 내부 내용은 블랙 박스로 처리되며 외부에서 컴포넌트 인터페이스 명세를 통해 컴포넌트를 사용할 수 있다. 컴포넌트에서 인터페이스 명세 방법은 매우 중요하며, 이에 대한 기법 중 하나는 contract를 사용하는 것이다. 컴포넌트의 contract는 다음 <표 1>과 같은 사항이 포함되어야한다[6][7].

<표1> 컴포넌트의 contract

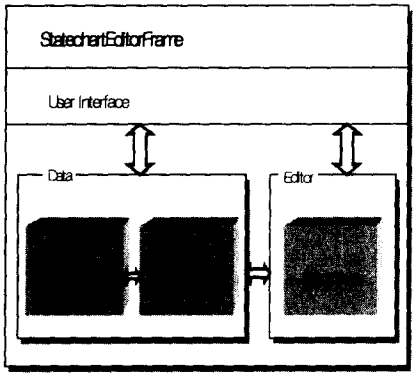
Functionality	컴포넌트의 기능을 명세
Environment	컴포넌트가 실행되는 환경을 명세
Interfaces	가장 중요한 부분 컴포넌트가 제공되는 서비스를 명세
Service Level	서비스 수준을 명세 (예 : 신뢰성, 확장성,...)

3. 컴포넌트 지원 Statechart 도구 설계

본 논문은 Statechart를 컴포넌트시켜 정형 명세의 재사용을 중점을 두었으므로 Statechart 도구 구현에서 컴포넌트와 관련된 부분을 중심으로 기술한다.

3.1 전체 시스템의 구성

본 논문에서 개발한 도구는 컴포넌트를 지원하는 Statechart 개발 도구로써 네부분으로 구성되며 <그림 1>과 같다.



<그림 1> 전체 시스템의 구성

① StatechartEditorFrame

StatechartEditorFrame는 도구의 메인 프레임으로써 도구에 대한 사용자 인터페이스를 제공한다. 또한 Statechart를 작성, 수정할 수 있는 EditorCanvas를 가지며 Statechart의 객체 표현 외에 도구에서 필요 한 정보입력은 모두 이 클래스에서 처리한다.

② EditorCanvas

EditorCanvas는 State, Transition등 Statechart의 객체를 표현하고 수정하는 기능을 제공한다.

③ StatechartObjectManager

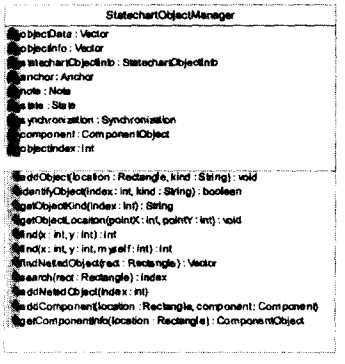
StatechartObjectManager는 EditorCanvas에서 표현된 Statechart 객체를 관리하며, 개발자가 컴포넌트를 생성시키려고 할 때 개발자에 의해 선택된 Statechart 객체를 ComponentObjectManager에 제공하여 컴포넌트가 구성될 수 있도록 한다.

④ ComponentObjectManager

ComponentObjectManager는 도구에서 가장 중요한 부분으로 컴포넌트를 생성하며, 생성된 컴포넌트를 관리하고, 컴포넌트의 인터페이스 명세인 Contract에 대한 정보를 관리한다.

3.2 도구 설계

개발하는 도구는 Statechart에 표현된 객체들 중 일부 또는 전체를 컴포넌트로 하여 다른 Statechart에 재사용에 있다. 이를 위해 컨포넌트가 구성될 수 있도록 지원하는 StatechartObjectManager와 Statechart 객체를 컴포넌트로 생성하고 관리하는 ComponentObjectManager가 가장 핵심되는 부분이며 이에 대한 설명은 <그림 2>와 <그림 3>에 있다.



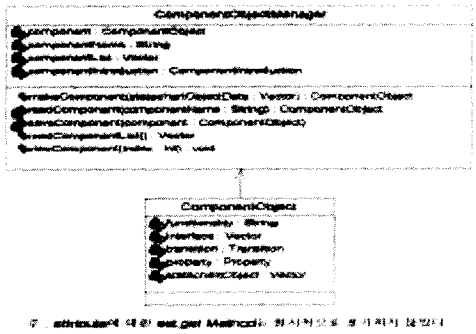
주 : attribute에 대한 set,get Method는 명시적으로표기하지 않았다.

<그림 2> StatechartObjectManager의 클래스 다이어그램

<그림 2>에서 StatechartObjectManager 클래스는 EditorCanvas에서 표현되는 Statechart의 요소들(예를 들어 State, Note)에 대한 클래스의 인스턴스를 생성하고 관리하는 클래스이다. 관리되는 Statechart 객체는 사용자가 컴포넌트를 생성할 때 필요한 정보가 된다. StatechartObjectManager는 Statechart를 표현하기 위해 다른 부수적인 클래스들을 가진다.

<그림 3>에서 ComponentObjectManager 클래스는 개발자가 컴포넌트 생성을 원할 경우 StatechartObjectManager가 가진 Statechart의 객체 자료와 개발자가 입력한 contract

정보를 결합하여 새로운 컴포넌트 객체를 생성시킨다. 또한, 이미 개발된 컴포넌트의 리스트(list)를 개발자에게 보여주고 사용할 수 있게 하는 기능과 선택된 컴포넌트의 이렇게 만들어진 Change Component를 시계의 contract를 보여주는 기능을 제공한다. 컴포넌트 객체의 자료구조는 ComponentObject 클래스 형태이다.



<그림 3> ComponentObjectManager의 클래스 다이어그램

4. 구현

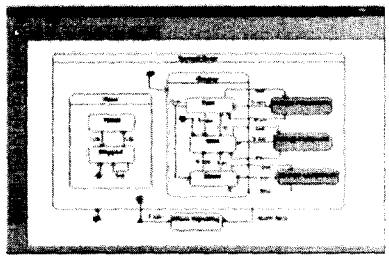
4.1 구현 환경

본 논문에서 구현한 컴포넌트 지원 Statechart 도구의 구현 환경은 다음과 같다.

CPU : Pentium III-500
RAM : 128M RAM
OS : MS-Windows 98
Language : Java 2
Programming Tool : JBuilder 3

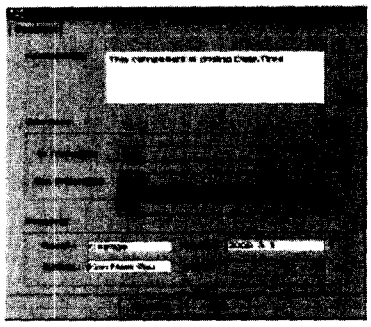
4.2 구현 예제

구현된 도구를 설명하기 위해 간단한 시계를 예제로 기술한다. 시계는 각각의 기능이 명확한 부품들로 이루워져서 작동하므로 각 부품들을 컴포넌트 시키기 적합하다. 본 논문에서는 시계의 날짜, 시간, 알람등을 설정(setting)하는 부분을 컴포넌트 시켰다. 시계에서 설정부분은 들어오는 입력값을 사용자가 원하는 대로 맞추는 부분과 그것을 돌려주는 부분으로 구성되어 있으며 서로가 공통의 성질을 가지고 있어서 재사용성에서도 컴포넌트 장점을 살릴 수 있다.



<그림 4> 계발된 도구를 이용한 시계예제

설정부분을 먼저 Statchart을 작성후에 컴포넌트 시키기 위해 선택후 Change Componet라는 컴포넌트를 만든다. Interface는 In Transition에서 Set, Out Transition에서 5 sec, Esc라는 것을 만들었다. Statechart에 추가하여 기존의 Statechart와 연결시키고 난 후 완성된 시계 Statechart의 모습은 <그림 4>와 같다. <그림 5>는 Change Component의 contract를 나타내고 있다.



<그림 5> Change Component의 contract

5. 결론

최근의 내장형 시스템은 점점 대형화되고 복잡하며 생명주기가 짧아지고 있는 추세이다. 이를 위해 시스템의 신뢰성과 안정성을 주기 위해 시스템 설계부터 명확한 기능을 명세할 수 있는 정형 명세 기법이 필요하다.

본 논문은 신속하면서 견고한 정형 명세 개발을 위해 컴포넌트를 지원하는 Statechart 지원하는 도구를 구현하였다. 새로운 시스템의 정형 명세 개발에 있어서 본 도구를 이용하면 이미 작성된 정형 명세를 최대한 재사용 함으로 최소의 시간으로 새로운 정형명세 개발할 수 있다. 개발된 도구를 이용함으로써 얻는 장점은 다음과 같다. 첫째, 개발된 Statechart를 컴포넌트 시킴으로써 새로운 제품에 대한 정형 명세 시간을 획기적으로 단축시킬 수 있다. 둘째로 새로운 제품의 정형명세 개발비용을 감축시키는 효과가 있다. 셋째로, 새로운 제품에 대한 명세를 컴포넌트 시킴으로써 유연성과 유지 보수성을 향상시킨다.

[참고 문헌]

[1] Hennessy and Patterson, Computer Architecture 2nd edition, 1997

[2] http://www.emultek.com/

[3] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesly, 1999

[4] D. Harel, " STATECHARTS : A VISUAL FORMALISM FOR COMPLEX SYSTEMS" , Science of Computer Programming 8, 1997

[5] D. Harel and A. Naamad. " The STATEMATE Semantics of Statecharts" , Published in ACM Trans. Soft. Eng. Method, October 1996

[6] N.H.Lassing, D.B.B. Risenbrij, J.C. van Vliet, " A View on Components" , IEEE, 1998

[7] C. Pfister, " Component Software : A Case Study using BlackBox Components" , http://www.oberon.ch/docu/case_study/index.html, 1997