임베디드 시스템을 위한 컴포넌트 기반 Statechart 도구 구현

김분희[0] 김삼택 김영찬
중앙대학교 컴퓨터공학과
{bhkim[0], stkim, yckim}@sslab.cse.cau.ac.kr

The Implementation of Statechart Tool Based on Component for Embedded System

Boon-Hee Kim[0] Sam-Tak Kim Young-Chan Kim
Dept. of Computer Science and Engineering, University of Chung-Ang

요 약

내장형 시스템의 경쟁력 향상을 위해 제품의 신속한 설계는 매우 중요하다. 내장형 시스템 설계에 사용하는 기존 Statechart 도구는 Statechart 재사용 측면에서 복사해서 붙이는 방법을 이용하고 있다. 이러한 방법은 새로운 제품 설계 시 시스템 개발자의 불필요한 투자가 요구되는 문제점이 있다.

본 논문은 이미 작성된 Statechart를 효율적으로 재사용하기 위해 컴포넌트 기능을 기존 도구 기능에 새롭게 추가함으로써 신제품의 신속한 설계를 지원하는 도구를 구현한다. 또한, 본 논문에서는 이러한 컴포넌트 기능을 추가하기 위해 컴포넌트 사용상 규약인 계약(contact)을 Statechart 구현에 적합하도록 재구성하였다. 구현된 컴포넌트 지원 Statechart 도구는 컴포넌트를 이용하여 새로운 제품의 신속한 설계를 지원함으로써 제품의 생명주기를 단축함은 물론 신제품의 시장적기진입을 제공할 수 있고, 이로써 제품의 경쟁력 향상에 도움 줄 수 있다.

1. 서 론

복잡성이 존재하는 내장형 시스템의 안정성을 보장받기 위해서는 시스템 설계 단계에서부터 정확한 기능적 명세가 필요하다. 순차 시스템의 명세를 위해 Z, VDM, Larch등이 있으며, 동시적 시스템의 명세는 Statechart, CSP, CCS등이 있다[3]. 이중 David Harel이 제안한 Statechart는 이벤트에 대해 동작을 명세하는 언어로써 계층적 상태도를 제공할 수 있다[4][5]. 그리고, 텍스트 형태의 정형 명세인 CSP, CCS과 비해 시각적인 정형 명세 기법을 제공하는 Statechart는 복잡한 제품의 기능적 명세를 쉽게 이해될 수 있는 장점이 있다. 또한, Statechart는 국제 표준화 기구인 OMG(Object Management Group)에서 객체지향 모델링 언어(UML)의 일부분으로 포함되어 사용하고 있다[6]. 본 논문은 내장형 시스템의 설계를 위해 Statechart를 이용한다.

이미 작성된 Statechart를 재사용 측면에서 기존 Statechart를 지원하는 도구인 Rapid PLUS, MagicDraw, xjChart등에서는 복사해서 붙이는 방법(Copy & Paste)을 사용하고 있다. 이 방법은 개발자가 기존 복잡한 시스템 설계에 대해 세밀한 분석이 필요할 뿐만아니라, 새롭게 작성될 시스템 설계에 적용하기도 어렵기 때문에 새로운 제품의 설계시 불필요한 투자와 시간이 소요될 수 있는 문제점이 존재한다.

본 논문은 기존 Statechart로 작성된 명세를 효율적으로 재사용하기 위해 재사용성에 널리 이용되고 있는 컴포넌트 기능을 기존 Statechart 도구에 추가하여 구현하였다. 이를 위해 기존 컴포넌트 인터페이스 명세 방법인 계약(contract) 정보를 Statechart 명세에 맞게 수정하였다. 이로써 본 구현 도구는 기존 Statechart 도구에 컴포넌트 생성, 유지 및 관리하는 기능을 추가하였다. 본 논문에서 추가된 컴포넌트 기능은 내장형 시스템의 설계 시 각 기능을 블록화 시킴으로써 새로운 시스템 제품의 신속한 설계를 지원할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 개발된 도구의 전체적인 구성과 예제를 통해 도구를 설명한다. 3장에서는 도구를 평가하며 마지막으로 4장에서 결론을 맺는다.

2. 컴포넌트 지원 Statechart 도구

2.1 도구 설계

본 도구는 컴포넌트를 지원하는 Statechart 도구로써 크게 다섯 부분으로 구성되어 있으며 그림 1과 같다.

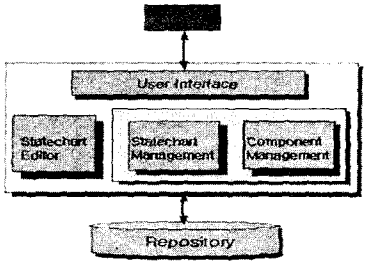

그림 1. 전체 시스템의 구성도

* 본 연구는 한국과학재단 목적기초연구(R01-1999-00246) 지원으로 수행 되었음.

개발된 도구의 구성 요소 중 재사용을 위한 컴포넌트를 생성하고 유지 및 관리하는 컴포넌트 관리기, 생성된 Statechart 및 컴포넌트를 저장하는 저장소가 기존 도구와 구별되는 요소이다. 그외 기존 도구에서 제공하고 있는 사용자 인터페이스와 Statechart를 작성, 수정 및 삭제할 수 있는 Statechart 편집기, Statechart 편집기에서 기술한 Statechart의 명세를 저장하고 관리하는 Statechart 관리기가 있다. 구현된 주요 클래스는 다음 표 1과 같다.

표 1. 주요 클래스 기능

클래스 이름	기 능
StatechartEditorFrame 클래스	사용자 인터페이스 제공
EditorCanvas 클래스	Statechart를 기술
ObjectData 클래스	Statechart의 명세를 저장
ComponentManagement 클래스	컴포넌트를 생성, 관리
Component 클래스	컴포넌트의 명세

표 1에서 본 논문에서 기존 도구와 달리 새롭게 제안하고 있는 컴포넌트 기능을 추가하기 위해 구현된 ComponentManagement 클래스과 Component 클래스에 대한 설명은 다음과 같다.

○ *ComponentManagement 클래스*

구현된 도구는 Statechart에 표현된 객체들 중 일부 또는 전체를 컴포넌트로 하여 다른 Statechart로 재사용 할 수 있다. 이를 위해 ComponentManagement 클래스는 컴포넌트를 구성하게 될 Statechart 객체를 생성 및 관리하는 StatechartObjectManager 클래스와 생성된 Statechart 객체를 컴포넌트 객체로 생성 및 관리하는 ComponentObjectManager로 구성되어있다.

○ *StatechartObjectManager 클래스*

StatechartObjectManager 클래스는 EditorCanvas 클래스에서 표현되는 Statechart의 객체들(예를 들어 State, Note)에 대한 클래스의 인스턴스를 생성하고 관리하는 클래스이다. Statechart 객체는 사용자가 컴포넌트를 생성시키는 구성요소이다. StatechartObjectManager 클래스의 대한 클래스 다이어그램은 그림 2과 같다.

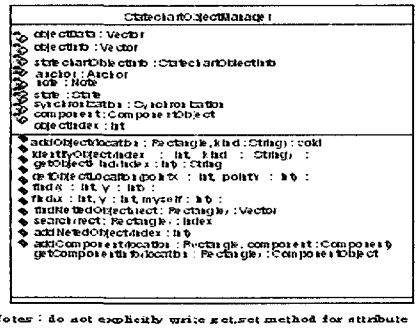

그림 2. StatechartObjectManager 클래스의 클래스 다이어그램

○ *ComponentObjectManager 클래스*

ComponentObjectManager 클래스는 개발자가 컴포넌트 생성시킨 경우 StatechartObjectManager가 가진 Statechart 객체와 개발자가 입력한 계약 정보를 결합하여 새로운 컴포넌트 객체를 생성시킨다. 또한, 이미 개발된 컴포넌트 리스트(list)를 개발자에게 보여주고 사용할 수 있게 하는 기능과 선택된 컴포넌트의 계약을 보여주는 기능을 제공한다. ComponentObjectManager 클래스의 클래스 다이어그램은 그림 3과 같다.

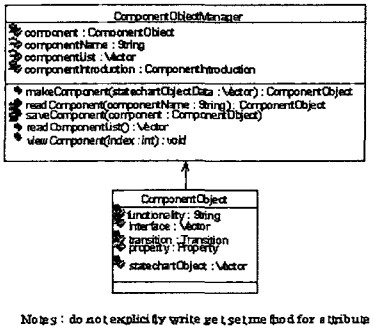

그림 3. ComponentObjectManager 클래스의 클래스 다이어그램

○ *Component 클래스*

Component 클래스에 지니고 있는 정보는 컴포넌트에 포함된 Statechart 객체들 의 정보와 컴포넌트 인터페이스 명세 방법인 계약(contract) 정보가 있다. 일반적으로 계약 정보는 컴포넌트 기능, 컴포넌트 환경, 인터페이스, 서비스 수준으로 분류된다. 이중 서비스 수준은 현재 연구가 진행 중에 있다[13][14][15]. 기존 계약 정보에서 본 논문은 현재 연구중인 서비스 부분을 제외하고, State 객체의 내부적인 행위를 정의하기 위해 행위(action) 부분을 추가하여 재구성하였으며 표 2와 같다.

표 2. 컴포넌트의 계약

컴포넌트 정보	내 용
컴포넌트 이름	컴포넌트의 고유명칭
컴포넌트 기능	컴포넌트의 기능을 기술한 부분으로 자연어로 서술
인터페이스	컴포넌트가 도구에서 명세한 기존 Statechart 객체나 컴포넌트와 전이(transition)로 연결되어 기술될 때 정의되어진 전이규칙을 제공
행위(action)	컴포넌트 내부에서 State 객체와 같은 내부적 행위를 정의를 기술

Component 클래스의 클래스 다이어그램은 그림 4와 같다. Component 클래스는 Statechart의 객체를 인스턴스로 가지며, 인터페이스 명세를 위해서 TransitionSpecification 클래스를, 행위 명세를 위해서 ActionSpecification 클래스를 사용한다. 그림 4에서 Statechart 명세시 필요한 위치정보에 대한 변수와 함수를 기술하지 않았으며, 자바에서 멤버변수를 사용하는 일반적인 함수표기법인 get, set형식의 함수와 생성자는 기술하지 않았다.

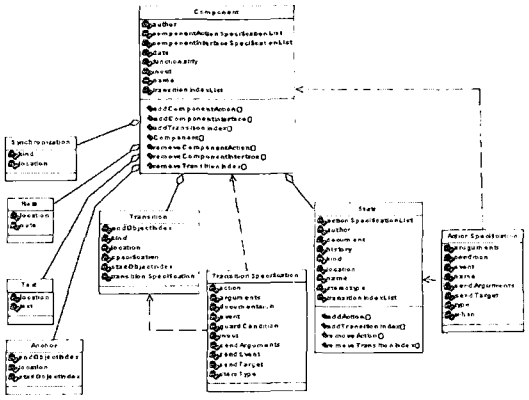

그림 4. Component 클래스의 클래스 다이어그램

2.2 도구 구현

개발한 도구에서 시계는 각각의 기능이 명확한 부품들로 구성되어 작동하므로 각 부품들을 컴포넌트 시킬 수 있다. 시계의 설정부분은 들어오는 입력값을 사용자가 원하는 대로 맞추는 부분과 입력값을 주었던 상태로 구성되있는 공통의 성질을 가지고 있어서 재사용성에서 컴포넌트 장점을 살릴 수 있기 때문이다.

개발된 도구로 시계의 Statechart를 작성한 화면은 그림 5와 같다. 그림 5는 기존 Statechart를 개발하는 기본적인 기능을 제공하고 있으며, 컴포넌트 개념을 제공하기 위해 추가된 부분(○)이 기존 도구와의 차이점이다. 추가된 기능은 컴포넌트를 생성, 유지, 관리하는데 사용하는 MakeComponent, ListComponent, ViewComponent 부분이다.

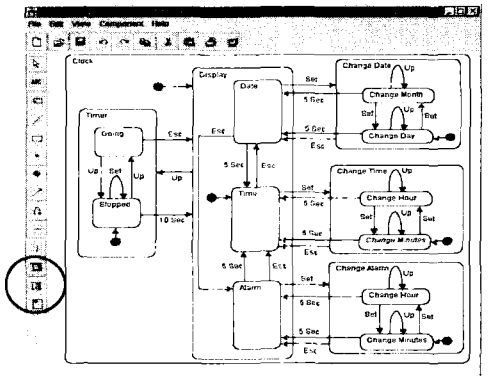

그림 5. 컴포넌트 지원 Statechart 도구

3. 비교평가

이미 작성된 Statechart를 재사용 하기 위해 기존 도구인 Rapid Plus 경우 Statechart 자체를 복사나 변경할 수 없고 모든 트리를 복사하여 사용하기 때문에 기존 설계에 대한 세심한 분석이 요구된다. MagicDraw나 xjChart의 경우 자바 기반으로 구현되어 있어서 플랫폼 독립성을 지원할 수 있다. 특히, MagicDraw 같은 경우 XML을 지원 함으로써 인터넷을 통해 상호 정보 교환을 할 수 있는 장점을 지니고 있다. 본 논문에서 제안하고 있는 도구는 자바로 구현되어 있으며, 컴포넌트를 지원함으로써 이미 설계된 Statechart를 효율적으로 재사용할 수 있는 장점을 지니고 있다. 기존 도구와 본 논문에서 제안하고 있는 도구의 비교는 표 3과 같다.

표 3. 기존 도구와 CSST 비교

	플랫폼 독립성지원	XML 지원	Copy & Paste 지원	Component 지원	Component Repository 지원
Rapid Plus	×	×	△	×	×
MagicDraw	○	○	○	×	×
xjChart 2.0	○	×	○	×	×
CSST	○	×	○	○	○

4. 결론

내장형 시스템이 대형화되고 복잡하지는 추세에서 그 생명주기가 짧아지면서 보다 신뢰할 수 있고 신속한 설계 방법의 요구가 증가되고 있다. 기존 Statechart 도구는 Statechart 재사용 측면에서 Statechart를 복사한 후 붙이는 방법으로써 재사용하고 있기 때문에 설계자의 세밀한 분석이 요구되며, 이는 제품의 생산성 향상을 저하시키는 원인이 되고 있다.

본 논문에서 제시한 컴포넌트 개념을 지원하는 Statechart 재사용 방법은 새로운 시스템의 설계 시 기존의 컴포넌트 기반 정형 명세를 재사용 함으로써 시스템 설계 시간을 보다 효율적으로 단축시킬 수 있다. 본 개발 도구의 장점을 다음과 같이 요약할 수 있다. 첫째, 본 논문에서 추가한 컴포넌트 기능을 이용하여 신제품의 설계 시간 및 노력을 단축시킬 수 있다. 둘째, 신제품의 빠른 설계를 지원함으로써 제품의 생명주기를 줄일 수 있다. 마지막으로, 신제품의 신속한 설계를 지원함으로써 시장적기진입이 가능하고, 이로 인해 제품의 경쟁력을 향상시킬 수 있는 도구로써 이용될 수 있다.

참고 문헌

[1] Hennessy and Patterson, Computer Architecture A Quantitative Approach 2nd edition, 1997.

[2] 조경순, "정형기법을 이용한 VLSI 검증", 2000 정형기법 워크숍, 2000.

[3] Edmund M. Clarke and Jeannette M. Wing, "Formal Methods : State of the Art and Future Directions", ACM Computer Survey, Dec. 1996.

[4] D. Harel, "STATECHARTS : A VISUAL FORMALISM FOR COMPLEX SYSTEMS", Science of Computer Programming, Aug. 1987.

[5] D. Harel and A. Naamad. "The STATEMATE Semantics of Statecharts", ACM Transaction Software Engineering Method, Oct. 1996.

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