내장형 시스템의 신속한 설계를 위한 컴포넌트 지원 Statechart 도구 구현

박 홍 진† · 김 영 찬††

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 설계시 시스템 개발자의 불필요한 투자와 시간이 소요되는 문제점이 있다.

본 논문은 이미 작성된 Statechart를 효율적으로 재사용하기 위한 컴포넌트 기능을 기존 도구 기능에 새롭게 추가하여 신제품의 신속한 설계를 지원하는 도구를 구현한다. 또한, 본 컴포넌트 기능을 추가하기 위해 컴포넌트 사용상의 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 있음으로써 제품의 경쟁력 향상에 도움 줄 수 있다.

Implementation of Component Support Statechart Tool for a Rapid Design of the Embedded System

Hong-Jin Park† · Young-Chan Kim††

ABSTRACT

The rapid design of the embedded system is crucial for improving the product's competitiveness. Existing statechart tools used for designing the embedded system rely on the copy and paste of the statechart for reuse. However, such method has a major drawback of wasting both time and cost of the system developer.

This paper implements the component supporting statechart tool. A tool that makes efficient reuse of a statechart by adding a component function to the existing functions is implemented in this paper. Also, to add a component function, this paper restructured the contract as protocol of the inter-component The implemented tool helps not only reduce the life cycle of a product, but also enhance a product's competitiveness by supporting the product's time-to-market

키워드 : 내장형 시스템(Embedded System), Statechart, 컴포넌트(Component)

1. 서 론

가전 제품을 비롯하여 통신 및 멀티미디어 시스템 내부는 마이크로 프로세서, 주문형 반도체 등으로 이루어진 하드웨어 및 응용 프로그램과 사용자 인터페이스 코드로 형성된 소프트웨어가 어우러져 복합적으로 구성된 내장형 시스템(embedded system)으로 이루어진다. 내장형 시스템을 이용한 제품의 생명주기(life cycle)는 매우 짧아 신제품 출시 속도는 30%이상 빨라지고, 복잡도도 매3년마다 2배정도 증가하고 있다[1]. 복잡성이 증가함에 따라 그에 따른 에러도 점점 증가하며 제품의 단순한 에러를 해결하기 위해서는 엄청난 비용과 시간이 소비될 수 있다(예를 들어 Intel Pentium FDIV bug[2]).

복잡성이 존재하는 내장형 시스템을 신뢰할 수 있고 안정성을 보장받기 위해서는 시스템 설계부터 보다 정확한 기능적 명세가 필요하다. 순차적 시스템(sequential system)의 명세를 위해 Z, VDM, Larch등이 있으며, 동시적 시스템(concurrent system)의 명세는 Statechart, CSP, CCS등이 있다[3]. 이중 David Harel이 제안한 Statechart는 이벤트에 대해 동작을 명세하는 언어로써 계층적 상태도를 제공할 수 있다[4, 5]. 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

※ 본 연구는 한국과학재단 목적기초연구(1999-2-303-008-3) 지원으로 수행되었음.
† 정 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 정 회 원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 2001년 3월 20일, 심사완료 : 2001년 6월 30일

설계를 위해 Statechart를 이용한다.

이미 작성된 Statechart를 재사용 측면에서 기존 Statechart를 지원하는 도구인 Rapid PLUS, MagicDraw, xjChart 등에서는 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. 구현된 도구는 컴포넌트 기능을 지원하기 위해 기존 Statechart 도구에 컴포넌트 생성, 유지 및 관리하는 기능을 추가하였다. 본 논문에서 추가된 컴포넌트 기능은 내장형 시스템의 설계시 각 기능을 블록화 시킴으로써 새로운 시스템 제품의 신속한 설계를 지원할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 Statechart를 위한 기존 개발된 도구를 기술하며, 기존 도구의 문제점과 제안한 도구의 장점을 기술한다. 3장에서 개발된 도구의 전체적인 구성과 예제를 통해 도구를 설명한다. 4장에서는 도구를 평가하며 마지막으로 5장에서 결론을 맺는다.

2. 기존 도구

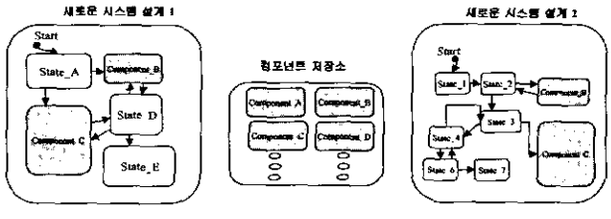
Statechart 기반의 내장형 시스템 구현 도구인 Rapid PLUS[7]는 휴대폰, 전자 시계, 항공기의 제어부분등 다양한 내장형 시스템을 실제 제품이 나오기 전에 미리 테스트와 디버깅을 할 수 있는 가상 프로토타입핑 도구이다. Rapid PLUS에서는 시스템의 기능을 설계하기 위해 모드 트리(mode tree)를 통해 Statechart를 생성시킨다. 따라서, 이 도구에서 Statechart를 변경하기 위해서는 Statechart 자체를 변경할수 없고, 모드 트리 변경을 통해 이루어 진다. 또한, 시스템의 명세언어인 Statechart를 재사용하기 위해서는 모드 트리를 복사하여 사용한다.

MagicDraw[8]는 Statechart가 포함된 UML 모델링을 지원하는 도구이다. MagicDraw는 C++, Java, CORBA IDL프로그램을 생성시킬 수가 있으며, UML제품군중 Rational Rose와 호환성을 지니고 있어서 확장성이 뛰어나다. 실시간 시스템, 클라이언트/서버 및 분산 어플리케이션을 설계시 유용한 도구이다.

xjChart[9]는 이벤트 기반에 응용 컴포넌트 개발을 위한 도구이다. xjChart는 Statechart를 구현하는 라이브러로써 Statechart를 작성하면 C++이나 Java로 소스 프로그램이 자동적으로 생성되는 특징이 있다. 또한, 생성된 소스 프로그램을 추가, 삭제가 가능하다. 그외 Statechart를 이용한 내장형 시스템도 설계도구는 ObjectDomain[10], Cover[11], Better State[12]등이 있다.

그러나, 기존 도구는 각각의 특징이 있으나, 신속한 설계를 위해 Statechart 재사용 측면에서는 기존 Statechart의 부분을 복사하여 붙이는 방법을 사용한다. 이 경우는 시스템 설계자가 복잡한 기존 시스템 설계도에서 재사용할 수 있는 Statechart를 찾기 위해 세밀한 분석이 요구된다. 또한, 기존 시스템 설계분석은 시스템 설계가 복잡하면 복잡할수록 어렵게 되며 찾아낸 Statechart를 새로운 시스템 설계에 추가하는 것도 쉽지 않을 수 있다.

본 논문에서 구현한 도구는 (그림 1)처럼 재사용 가능한 부분을 컴포넌트 저장소에 저장한 후 새로운 시스템 설계시 이미 저장된 컴포넌트를 재사용할 수 있으므로써 설계 시간 및 노력을 단축시킬 수 있는 장점을 지닌다

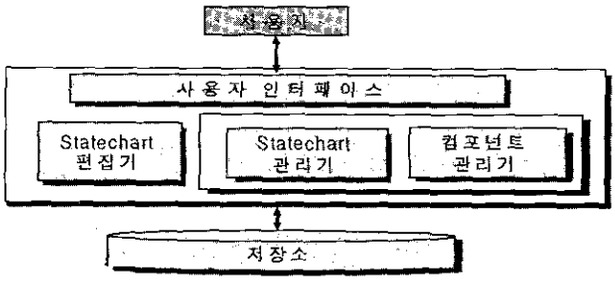


(그림 1) 컴포넌트를 이용한 Statechart 재사용

3. 컴포넌트 지원 Statechart 도구

3.1 도구 설계

구현한 도구는 컴포넌트를 지원하는 Statechart 도구로써 크게 다섯 부분으로 구성되어 있으며 (그림 2)와 같다.



(그림 2) 전체 시스템의 구성도

개발된 도구의 구성 요소중 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 1>과 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

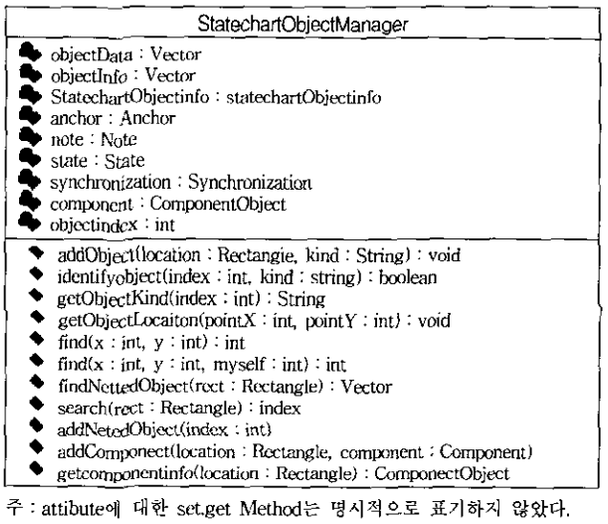
<표 1>에서 본 논문에서 기존 도구와 달리 새롭게 제안하고 있는 컴포넌트 기능을 추가하기 위해 구현된 ComponentManagement 클래스과 Component 클래스에 대한 설명은 다음과 같다.

• *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 3)과 같다.

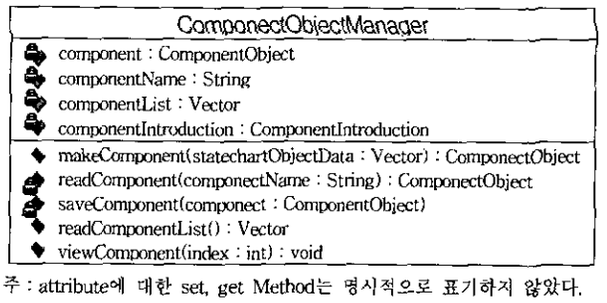


주 : attibute에 대한 set,get Method는 명시적으로 표기하지 않았다.

(그림 3) StatechartObjectManager 클래스의 클래스 다이어그램

• *ComponentObjectManager 클래스*

ComponentObjectManager 클래스는 개발자가 컴포넌트 생성시킬 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 4)와 같다.



주 : attribute에 대한 set, get Method는 명시적으로 표기하지 않았다.

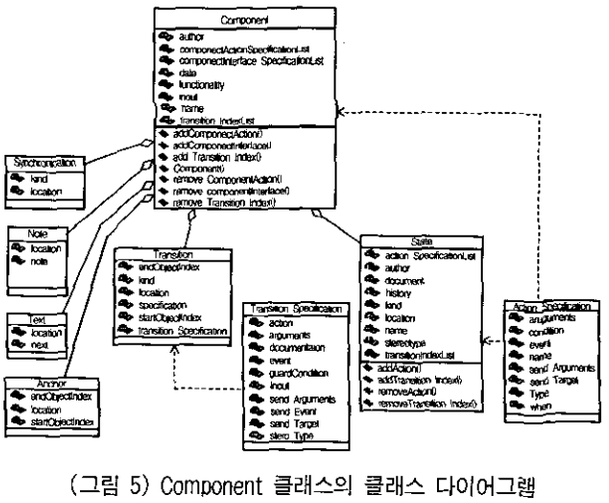
(그림 4) ComponentObjectManager 클래스의 클래스 다이어그램

• *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 [13-15]. 기존 계약 정보에서 본 논문은 현재 연구중인 서비스 부분을 제외하고, State 객체의 내부적인 행위를 정의하기 위해 행위(action) 부분을 추가하여 재구성하였으며 <표 2>와 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



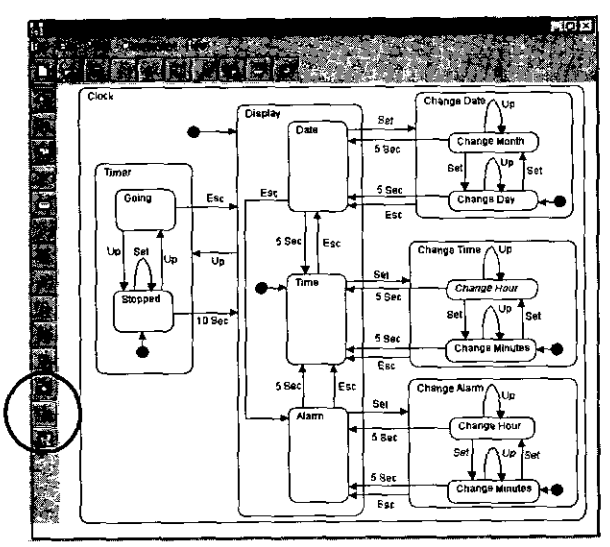
(그림 5) Component 클래스의 클래스 다이어그램

Component 클래스의 클래스 다이어그램은 (그림 5)와 같다. Component 클래스는 Statechart의 객체를 인스턴스로 가지며, 인터페이스 명세를 위해서 TransitionSpecification 클래스를, 행위 명세를 위해서 ActionSpecification 클래스를 사용한다. (그림 5)에서 Statechart 명세시 필요한 위치 정보에 대한 변수와 함수를 기술하지 않았으며, 자바에서 멤버변수를 사용하는 일반적인 함수표기법인 get, set형식의 함수와 생성자는 기술하지 않았다.

3.2 도구 구현

개발한 도구는 JDK 1.2.2과 Win98을 기반으로 하였으며, 자바로 구현하였다. 본 논문에서 추가하고 있는 기능인 컴포넌트를 설명하기 위해 시계에 대한 Statechart 설계를 통해 기술한다. 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.

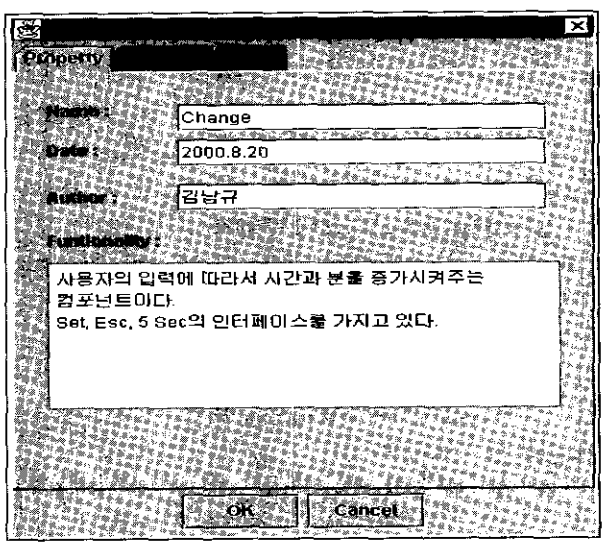
개발된 도구로 시계의 Statechart를 작성한 화면은 (그림 6)과 같다. (그림 6)은 기존 Statechart를 개발하는 기본적인 기능을 제공하고 있으며, 컴포넌트 개념을 제공하기 위해 추가된 부분(○)이 기존 도구와의 차이점이다. 추가된 기능은 컴포넌트를 생성, 유지, 관리하는데 사용하는 MakeComponent, ListComponent, ViewComponent 부분이다.



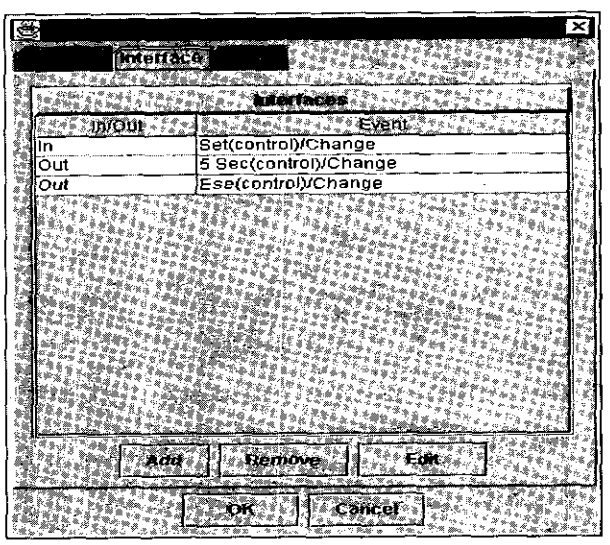
(그림 6) 컴포넌트 지원 Statechart 도구

본 논문에서 추가된 컴포넌트 기능을 설명하기 위해 먼저 이미 구현된 Statechart를 재사용 하기 위해 컴포넌트화 시키는 과정을 먼저 설명하며, 다음으로 저장소에 저장된 컴포넌트를 재사용 하는 과정을 (그림 6)의 예시된 시계를 통해 기술한다.

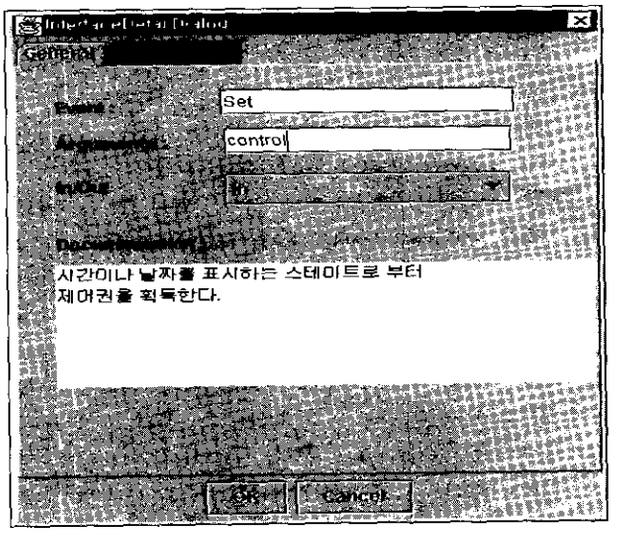
개발된 도구에서 컴포넌트를 생성시키기 위해서 Make-Component버튼을 누른 후 화면에 사용자가 컴포넌트로 생성시킬 부분을 드래그(drag)하여 사각형을 만들면 Make-Component 대화상자가 활성화된다. 이때 활성화된 대화상자는 컴포넌트의 명세를 나타내는 계약 부분이다. 예를 들어 (그림 6)에서 오른쪽 Change Time 상태를 컴포넌트로 생성시키기 위해 컴포넌트로 원하는 부분을 드래그하면 (그림 7)과 같이 계약 부분의 대화상자가 활성화된다. (그림 7)에서는 계약의 이름, 기능등을 기술하며, 인터페이스를 선택하면 (그림 8)과 같이 컴포넌트의 인터페이스를 추가 및 삭제시킬 수 있는 대화상자가 활성화된다. 컴포넌트 인터페이스 추가 버튼을 선택하면 (그림 9)와 같이 컴포넌트 인터페이스를 명세하기 위한 대화상자가 활성화되며 이를 기술함으로써 원하는 부분에 컴포넌트를 생성시킬 수 있다.



(그림 7) 컴포넌트의 계약 명세

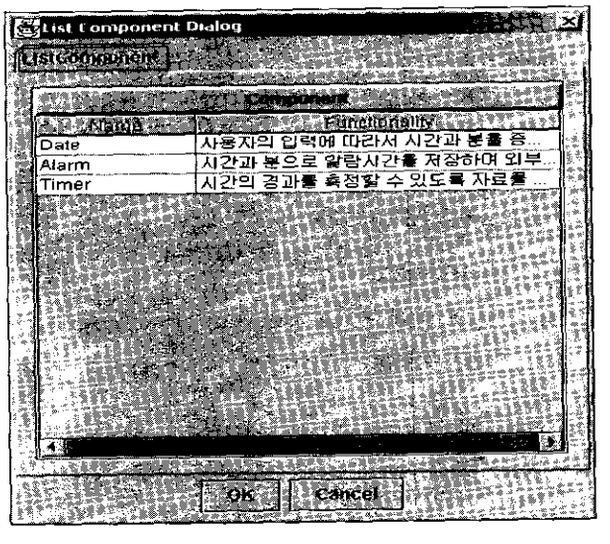


(그림 8) 컴포넌트의 인터페이스 리스트



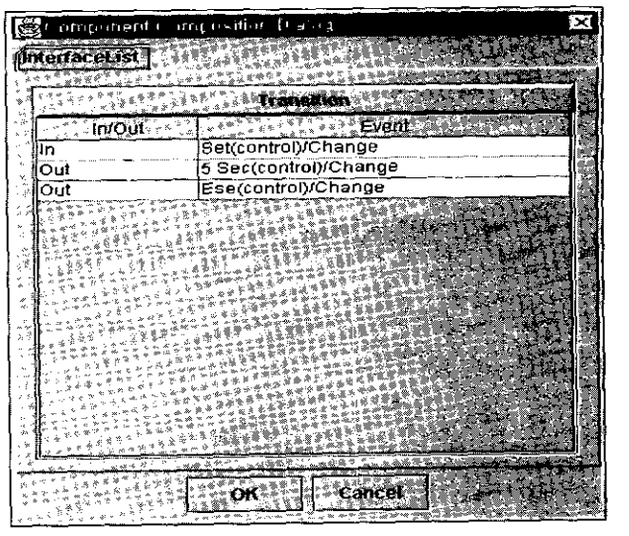
(그림 9) 컴포넌트의 인터페이스 명세

사용자가 컴포넌트를 생성시키면 컴포넌트는 저장소에 저장된다. 저장소에 저장된 컴포넌트는 개발된 도구의 List-Component 버튼을 선택하여 볼 수 있다(그림 10). 또한, 저장된 컴포넌트는 새로운 제품을 설계시 재사용하여 사용될 수 있다.



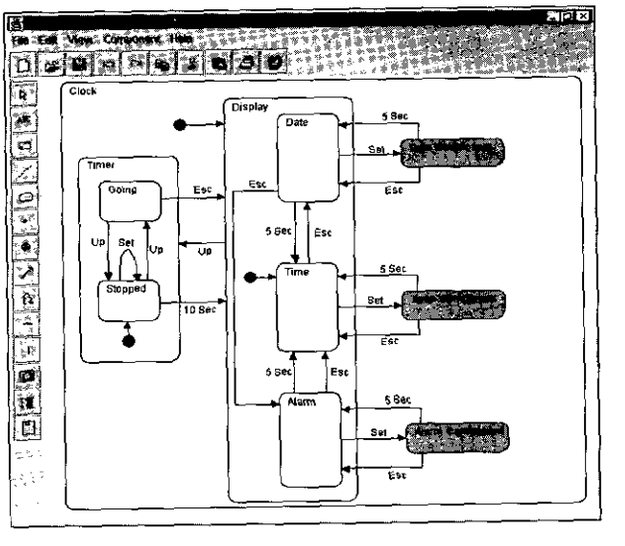
(그림 10) 저장소에 있는 컴포넌트 리스트

작성된 Statechart에서 컴포넌트를 재사용하기 위해서는 컴포넌트 리스트에서 추가할 컴포넌트를 선택한 후 원하는 위치를 선택하면 컴포넌트가 화면에 표시된다. 추가한 컴포넌트와 이미 구현된 Statechart간 연결을 위해 전이(transition)를 생성시키기 위해서는 추가한 컴포넌트와 구현된 Statechart를 연결하면 (그림 10)과 같이 Component Composition 대화상자가 활성화된다. 이때 사용자는 컴포넌트 생성시 정의한 인터페이스를 선택하여 전이를 생성시킬수 있다.



(그림11) Composition Component Dialog에 나타난 컴포넌트의 인터페이스 리스트

이와 같은 방법으로 개발된 도구에서는 저장된 컴포넌트를 재사용할 수 있으며, (그림 6)에서 Change Date, Change Time, Change Alarm 부분을 각각 컴포넌트화 시키면 (그림 12)와 같다.



(그림 12) 컴포넌트화 된 시계 예제

4. 비교 평가

이미 작성된 Statechart를 재사용 하기 위해 기존 도구인 Rapid Plus 경우 Statechart 자체를 복사나 변경할 수 없고 모드 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 3>과 같다.

〈 표 3 〉 기존 도구와 CSST 비교

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

5. 결 론

내장형 시스템은 점점 대형화되고 복잡하며 생명주기가 짧아질수록 보다 신뢰할 수 있고 신속한 설계 방법의 중요도가 증가될 것이다. 기존 Statechart 도구는 Statechart 재사용 측면에서 Statechart를 복사하여 붙여서 재사용하고 있기 때문에 설계자의 세밀한 분석이 요구되며 이는 제품의 생상성 향상을 저하시키는 원인이 될 수 있다.

본 논문에서는 제시하고 있는 컴포넌트의 개념을 지원하는 Statechart 재사용은 새로운 시스템의 설계시 이미 컴포넌트로 구현된 정형 명세를 재사용 함으로 시스템 설계 시간을 보다 효율적으로 단축시킬 수 있는 장점을 지닌다. 개발된 도구의 장점은 다음과 같다. 첫째, 본 논문에서 추가된 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 시장적기진입으로 인해 제품의 경쟁력을 향상시킬 수 있는 도구로 이용될 수 있는 장점이 있다.

참 고 문 헌

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

[2] 조경순, "정형기법을 이용한 VLSI 검증", 2000 정형 기법 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.

[7] http : //www.e-sim.com/embed/index.htm.

[8] http : //www.nomagic.com/magicdrawuml/.

[9] http : //xjtek.com/products/xjCharts/index.html.

[10] http : //www.objectdomain.com/domain30/index.html.

[11] http : //xjtek.com/products/covers/31/index.html - Covers 3.1a.

[12] http : //www.wrs.com/products/html/betterstate.html - Better State.

[13] D. F. D'Souza and A. C. Wills, Objects, Components, and Frameworks with UML : The Catalysis Approach, Addison-Wesley, 1998.

[14] N. H. Lassing, D. B. B. Risenbrij, J. C. van Vliet, "A View on Components," Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, Sep. 1998.

[15] C. Pfister, "Component Software : A Case Study using BlackBox Components," http : //www.oberon.ch/docu/case_study/ch3.html, 1997.

박 홍 진

e-mail : hjpark@sslab.cse.cau.ac.kr

1993년 원광대학교 컴퓨터공학과 졸업 (공학사)

1995년 중앙대학교 대학원 컴퓨터공학과 (공학석사)

1997년~현재 중앙대학교 대학원 컴퓨터공학과 박사과정

관심분야 : 가상 프로토타입핑, 실시간 시스템, 운영체제

김 영 찬

e-mail : yckim@chungang.edu

1965년 연세대학교 전기공학과(공학사)

1968년 연세대학교 대학원 전자 공학과 (공학석사)

1983년 연세대학교 대학원 전자공학과 (공학박사)

1982년~1983년 프랑스 Grenoble대학 연구교수

1996년~1996년 한국정보과학회장

1973년~현재 중앙대학교 컴퓨터공학과 교수

관심분야 : 운영체제, 분산시스템