제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

PBO 모델에 기반한 컴포넌트 기반 실시간 내장형 시스템 개발 툴셋의 설계에 관한 연구

서영준, 이승룡, 송영재
경희대학교 컴퓨터공학과
e-mail: yjseo@khu.ac.kr

A Study on Design of Component-based Real-Time Embedded System Development Toolset based on PBO model

Young-Jun Seo, Sungyoung Lee, Young-Jae Song
Dept of Computer Engineering, Kyung-Hee University

요 약

최근 디지털 가전과 모바일 컴퓨팅이 화두로 떠오르면서 내장형(embedded) 기술이 전성기를 맞이하고 있다. 실시간 내장형 시스템을 컴포넌트 기반 소프트웨어 기술을 사용하여 구축한다면, 재사용과 재구성이 용이해 저렴하고 신뢰성 있는 시스템 개발이 가능해진다. 이러한 이유로 인하여, 자원이 제한된 실시간 내장형 시스템 개발 사용에 적합한 컴포넌트 모델이 소개되고 있으며, 그 중 PBO 모델은 다른 모델에 비해 실시간 시스템에 적합한 인터페이스를 정의하고 있으나, 실시간 내장형 시스템 개발에 적용되는 범위를 넓히기 위해서는 구성과 분석 기능을 제공하는 툴셋을 필요로 하고 있다. 따라서, 본 논문에서는 구성과 분석 기능을 포함하는 PBO 모델에 기반한 개발 툴셋의 아키텍처를 제안하였으며, 이를 통해 설계 시간과 에러 감소, 생산성을 증가하는 방안을 제시하였다.

1. 서론

[1]최근 디지털 가전과 모바일 컴퓨팅이 화두로 떠오르면서 내장형(embedded) 기술이 전성기를 맞이하고 있다. 내장형 기술은 모바일 시스템에서부터 가전제품, 산업용 기기까지 그 적용분야가 다양하며, 그에 따른 운영체제 또한 셀 수 없이 많다. Gartner Dataquest에 따르면 2000년 전세계 내장형 소프트웨어 시장규모가 8억 6,600만 달러에 이르러 1999년의 6억 2,300만 달러 대비 37%의 성장률을 기록한 것으로 나타났다(출처: Dataquest 2001.7). 프로세서의 측면에서 보더라도 1998년에 이미 개발되는 모든 프로세서의 2%가 일반적 목적의 워크스테이션이었으며, 나머지 98%는 내장형 시스템이었다. 2000년 현재에는 0%에 접근하고 있다. 즉, 내장형 시스템과 프로세서의 수가 워크스테이션보다 훨씬 빠르게 증가하고 있는 추세이다.

내장형 시스템(embedded system)은 실세계와 중단 없는 상호작용을 하는 고정된 프로그램을 갖는 프로세서 시스템이다. 내장형 시스템은 도처에 산재한(ubiquitous) 컴퓨팅 플랫폼에 적용되며 자원의 제약이 심한 환경에서 안전성, 신뢰성의 요구사항을 만족해야 하고 이동성, 실시간성, 융통성, 상호 운용성, 이식성 등을 지원해야 한다[1].

이러한 시스템을 위한 소프트웨어는 빠르고 신뢰성 있게 생산해야 하나, 현재 내장형 시스템을 개발하기 위한 효율적인 방법이 부족한 실정이다. 그 대안이 될 수 있는 방법으로 내장형 시스템 개발 툴셋이 주목 받고 있으며, 이러한 변화와 아울러 툴셋도 재사용, 적응성을 효율적으로 지원할 수 있는 컴포넌트기반 개발 방법론을 적용하고 있다.

컴포넌트기반 개발 방법론(Component-based Software Engineering: CBSE)은 빠르게 시스템을 구축할 수 있고, 제품의 품질을 개선할 수 있으며, 컴포넌트 재사용이 가능하여 개발 비용을 절감시킬 수 있고, 소프트웨어의 증가하는 복잡도를 감소시킬 수 있다는 이점이 많다. 이 때문에, 이를 실시간 내장형 시스템 개발에 적용하려는 연구가 진행중이다[2,3].

현재 PBO, VEST와 같은 여러 컴포넌트 기반 실시간 내장형 모델들이 소개되고 있으며, 실시간 속성, 인터페이스/통신, 구성 툴, 분석 툴, 재사용성, 개조 능력과 같은 특징들에 대해서 장단점을 가지고 있다. PBO는 제한된 실시간 속성과 설계 가이드라인만을 제공하는 구성 툴 지원, 그리고 분석 툴을 지원하지 않는 단점이 있으나, 인터페이스를 정의하지 않은 VEST에 비해 실시간 시스템에 적합한 인터페이스를 정의하고 있다. 즉, PBO가 실시간 내장형 시스템 개발에 적용되는 범위를 넓히기 위해서는 구성과 분석 기능을 제공하는 툴셋을 필요로 하고 있다.

따라서, 본 논문에서는 실시간 내장형 환경에서 수행되는 애플리케이션 개발에 적합하도록 실시간 내장형 컴포넌트 모델인 PBO(Port-Based Object)에 기반한 개발 툴셋의 아키텍처를 제안한다. 제안한 개발 툴셋은 구성과 분석 기능을 포함하는 6개의 소프트웨어 컴포넌트들로 모듈화 되어 지원되며, 이를 통해 본 논문에서는 설계 시간과 에러 감소, 생산성 증가라는 최적의 환경에서 실시간 내장형 시스템을 개발하는 방안을 제시하였다.

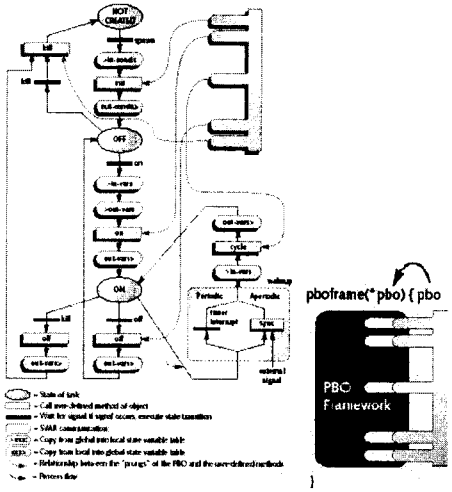
1) 본 연구는 한국과학재단 목적기초연구(과제번호 : R01-2001-000-00357-0) 지원으로 수행되었음.

2. 관련 연구

본 장에서는 컴포넌트기반 실시간 내장형 모델들과 그 특징들을 비교하여 살펴보도록 하겠다.

2.1 컴포넌트기반 실시간 내장형 모델

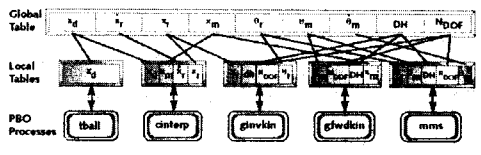
PBO(Port-based object)[4]는 Carnegie Mellon Univ에서 개발하였으며, 내장형 실시간 제어 시스템의 개발을 위한 컴포넌트 모델이다. PBO의 데이터 흐름은 입출력 포트를 통하여 규정하며, 컴포넌트 기반 소프트웨어 지원은 프레임워크 프로세스(framework process)라 불리는 단일의 표준 프로세스를 생성함으로서 구체화 된다. 프레임워크는 인자로서 PBO를 취하며, 세 가지 상태(NOT_CREATED, ON, OFF)를 갖는 유한 상태기계로 구성된다. 또한 PBO가 필요로 하는 모든 선언을 포함하는 매크로 확장을 통해 구현된다.



(그림 1) PBO 프레임워크의 프로세스 플로우 다이어그램

(그림 1)은 프레임워크에 플러그인(plugs in) 되는 PBO와 프레임워크 프로세스의 내부를 기술하는 프로세스 플로우 다이어그램을 나타낸다. PBO는 PBO가 구현해야 하는 함수 집합(Init, On, Cycle, Off, Kill)을 통해 PBO 프레임워크와 인터페이스한다. 인터페이스 함수는 PBO의 생명주기의 특정 단계에서 PBO 프레임워크에 의해 호출 된다.

PBO간의 통신은 (그림 2)와 같이 전역, 지역 테이블에 저장된 상태 변수(state variable)을 통해 수행된다. PBO는 오직 지역 테이블만을 접근할 수 있으며, 모든 PBO가 지역 테이블을 가지기 때문에 읽고 쓰기 위해 동기화 할 필요가 없다. 따라서, PBO 프로세스는 다른 프로세스에 독립적으로 수행될 수 있다. 그러나, 전역 테이블의 동일 상태 변수에 대한 접근은 락킹 메카니즘(locking mechanism)에 의해 상호 배제(mutually exclusive) 된다.



(그림 2) PBO의 조립을 위한 SVAR 메카니즘의 구조

2.2 컴포넌트기반 내장형 실시간 모델의 비교

다음 <표 1>은 컴포넌트 기반 내장형 실시간 모델들의 특징들을 비교 정리하였다[6].

<표 1> 컴포넌트기반 내장형 실시간 모델의 특징들

x : 전체 지원 x/p : 부분 지원

모델 / 특징		VEST[5]	PBO
실시간 속성	not preserved		
	preserved	x	x/p
인터페이스/통신	standardized		
	system specific		
	ports/unbuffered	-	x
구성 툴	not available		
	available	x	x/p
분석 툴	not available		x
	available	x/p	
재사용성	component	x	x
	architecture	x	
개조 능력	none		
	low		
	moderate		
	high	x	x

첫째, 실시간 속성. 내장형 실시간 시스템의 컴포넌트는 신뢰성 있는 시스템을 조립하기 위해 예측 가능하여야 하며, 따라서 WCET(Worst-Case Execution Time), release time, deadline, precedence constraints, mutual exclusion, period와 같은 잘 정의된 시간 속성을 가져야 한다. PBO 모델의 컴포넌트는 frequency, deadline의 오직 두 가지 시간 속성만을 가진다. VEST는 전체 시간 속성 리스트뿐만 아니라 전력 소비, 메모리 요구 같은 내장형 환경에 필수적인 속성을 갖는 리스트로 확장 가능하다.

둘째, 인터페이스/통신. 컴포넌트는 잘 정의된 인터페이스를 통해 다른 컴포넌트와 통신을 하며, 실시간 컴포넌트 통신은 메시지 패싱을 통해 행해지는 버퍼(buffered) 통신과 공유 메모리를 통한 비버퍼(unbuffered) 통신의 두 가지 가능한 방법이 있다. 그러나, 실시간 시스템 중 인터페이스가 정의되지 않은 VEST를 제외하고는 송수신 메시지가 오버헤드가 발생할 수 있으며, 버퍼 오버플로우의 결과로서 중요한 메시지 손실이 발생할 가능성이 있는 버퍼 통신의 단점 때문에 비버퍼 통신을 사용한다.

셋째, 구성 툴. 자동화된 구성(configuration)은 새로운 컴포넌트를 개발하고 컴포넌트 라이브러리부터 적합한 컴포넌트를 선택하고 조립하기 위한 규칙을 지원한다. VEST에서 조립 규칙은 네 가지 타입의 의존성 검사를 통해 정의된다. PBO 모델은 컴포넌트들로부터 시스템을 조립할 때 설계자를 돕기 위한 가이드라인을 제공한다.

넷째, 분석 툴. 조립된 시스템의 신뢰성은 컴포넌트의 정확성에 의존하기 때문에, 분석 툴은 컴포넌트와 조립된 시스템의 행위를 검증하기 위해 필요하다. 특히, 실시간 시스템은 시간 속성을 만족해야 하고, 시스템이 시간 속성을 만족하도록 하기 위해 적합한 분석이 요구된다. VEST는 시스템의 신뢰성과 실시간 분석 표기를 소개하나 상세한 내용은 제공하지 않고 있다.

다섯째, 재사용성. 시스템은 라이브러리의 재사용 컴포넌트로 조립되며, 시스템의 아키텍처는 시스템의 개발에 재사용 될 수 있다. 따라서, 내장형 실시간 시스템에서 재사용할 수 있는 부분은 컴포넌트와 아키텍처이다. 이중 모든 시스템에서 재사용될 수 있는 부분은 컴포넌트이며, VEST는 아키텍처도 재사용될 수 있

으므로 높은 수준의 재사용성을 가진다.

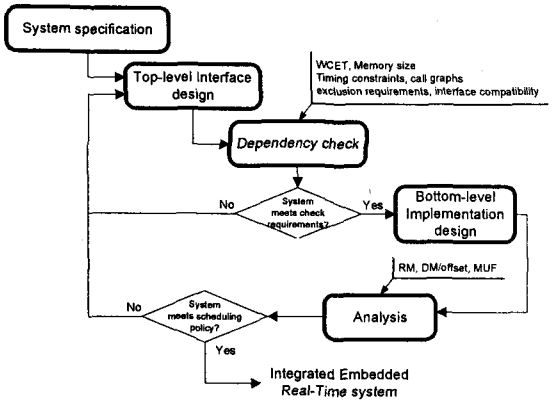
여섯째, 개조(tailoring) 능력. 구성 시스템은 높은 개조 능력을 가진다.

3. 컴포넌트기반 실시간 내장형 개발 툴셋

본 논문에서 제안한 컴포넌트 기반 실시간 내장형 개발 툴셋은 PBO 모델에 기반하여 설계된 컴포넌트의 선택과 컴포넌트간의 비주얼한 계층적 조립, 그리고 의존성 검사와 분석, 코드 생성을 수행한다.

3.1 개발 프로세스

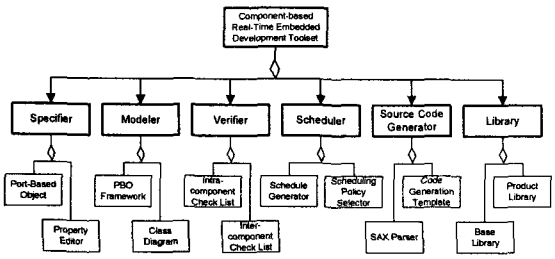
개발 프로세스는 (그림 3)과 같이 시스템 명세, 상위레벨 인터페이스 설계, 의존성 검사, 하위레벨 구현 설계, 분석의 여러 단계로 나뉘어 진다. 시스템 명세에서는 모든 태스크를 PBO 모델을 사용하여 기술하여 라이브러리에 저장한다. 상위레벨 인터페이스 설계에서는 라이브러리의 컴포넌트 후보들로 비기능적 제약과 함께 시스템을 계층적으로 설계한다. 의존성 검사 단계는 시스템 조립 과정 중에 컴포넌트 사이의 의존성을 검사하기 위해 호출된다. 하위레벨 구현 설계에서는 구체적인 인스턴스들을 클래스 다이어그램을 사용해 기술한다. 분석 단계는 조립된 시스템이 deadline을 만족하는지 여부를 확인하기 위해 적합한 스케쥴링 알고리즘을 사용하여 스케쥴 한다. 상위 레벨 인터페이스 설계, 의존성 검사, 하위레벨 구현 설계, 분석 단계는 시스템 통합을 위한 모든 적합한 조건이 만족할 때까지 반복될 수 있다.



(그림 3) 개발 프로세스

3.2 아키텍처

본 논문에서 제안한 개발 툴셋은 (그림 4)와 같이 6개의 기본 소프트웨어 컴포넌트들로 구성되며, 사용 순서에 따라 명세자, 모델러, 검증기, 스케쥴러, 소스 코드 생성기, 라이브러리가 포함된다.



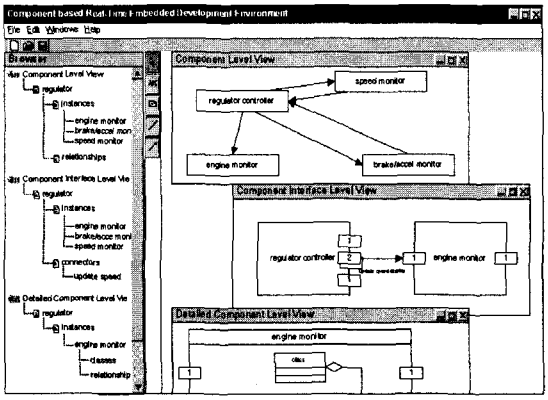
(그림 4) 전체 아키텍처

3.2.1 명세자(Specifier)

명세자에서는 PBO 모델이 제공되며 모든 내장형 실시간 태스크가 PBO 컴포넌트 모델을 사용하여 기술 된다. 기술된 컴포넌트는 재사용 개체이고, 타입 기술과 구현을 포함하는 정적 구조이다. 아이디, 버전, 실시간, 내장형 제약이 모델내에 기술 되며 저장될 수 있다. 표준 인터페이스인 4가지 타입의 포트(in, out, resource, configuration 포트)가 기술된다. resource 포트는 I/O 디바이스 드라이버를 통해 센서(sensor)와 구동기(actuator)에 연결되며, configuration 포트는 특정 하드웨어 또는 애플리케이션 사용을 위해 generic 컴포넌트의 재구성을 위해 사용된다. in, out 포트는 얼마나 많은 배수의 포트가 초기화 되어야 하는지를 가리키는 MNOI(Maximum Number of Instance) 속성을 가지며, 향후 PBO 컴포넌트 모델들 사이의 통신을 위해 local table에 저장되는 상태 변수를 선언한다.

3.2.2 모델러(Modeler)

모델러는 PBO 모델을 사용하여 동적 행위를 정의하며, 컴포넌트의 선택과 계층적 조립을 수행한다. 모델러는 (그림 5)와 같이 컴포넌트 레벨 뷰, 컴포넌트 인터페이스 뷰, 상세 컴포넌트 레벨 뷰의 세 개의 설계 모델 뷰를 제공하며, 뷰 간에는 변경이 이루어질 때 동기화를 지원한다. 컴포넌트 레벨 뷰는 애플리케이션에 사용되는 컴포넌트 인스턴스들의 집합이며, 애플리케이션은 여러 개의 컴포넌트 레벨 뷰로 구성된다. 기능적으로 관련된 컴포넌트 인스턴스는 같은 뷰에 넣어진다. 라이브러리로부터 선택된 컴포넌트들은 인스턴스화 되며, 개체화 될 수 있는 포트의 최대 수는 MNOI 속성에 의해 주어진다. I/O 포트에 연결된 local table의 상태 변수도 인스턴스화 되며, 컴포넌트는 오직 직접 연결된 local table만을 액세스 할 수 있다. 컴포넌트 인터페이스 뷰는 상호 연결된 컴포넌트 인스턴스들의 집합체이다. 애플리케이션은 다중의 컴포넌트 인터페이스 뷰로 구성되며, 컴포넌트 인스턴스는 하나 이상의 컴포넌트 인터페이스 뷰에 나타날 수 있다. 컴포넌트 간의 통신을 위해 공유 메모리에 저장되는 global table을 정의하며, 컴포넌트 레벨 뷰에서 인스턴스화 된 각각의 local table과 연결된다. 상세 컴포넌트 레벨 뷰는 컴포넌트의 내부와 포트 사이의 관계를 기술한다. 컴포넌트의 내부는 UML 클래스 다이어그램을 사용해 설계 되며, 특정 애플리케이션의 특징들이 추가된다. 또한, 각 컴포넌트는 생성, 설정된 PBO 프레임워크와 구현해야 하는 함수 집합을 통해 연결된다.



(그림 5) 모델러의 조립 예시

3.2.3 검증기(Verifier)

개발 초기에 timing error와 같은 설계 결함을 탐지하는 것은 시스템 개발의 마지막 단계에서 재설계를 피하는데 중요하다. 검증기는 VEST의 의존성 검사 기능을 사용하여 태스크 집합이

모든 주어진 실시간, 내장형 제약을 만족하는지를 보여준다. 의존성의 복잡성 때문에 컴포넌트 내부와 컴포넌트간 검사의 두 가지 형태로 나뉘며, 각 형태마다 세부적인 검사 리스트를 포함한다.

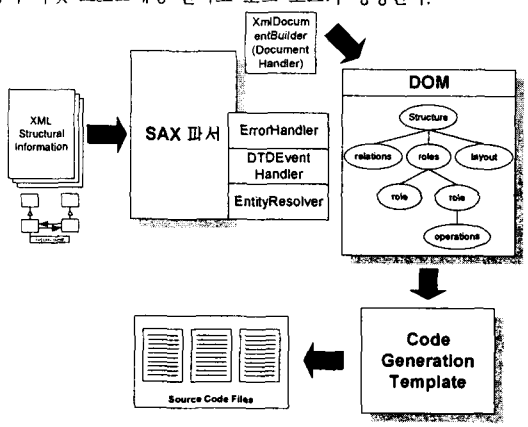
컴포넌트내(Intra-component) 검사는 가장 간단하며, 리스트는 확장 가능하다. 검사되는 의존성 요구사항 수가 많을수록 간단한 실수를 피하기 쉽다. 이러한 검사는 단순한 방식으로 컴포넌트 속성을 검사함으로서 수행될 수 있으며 컴포넌트들의 attribute들 사이에서 기능적 의존성이 없다. 예를 들면, memory size 검사는 각 컴포넌트의 메모리 요구 사항과 하드웨어 컴포넌트에 의해 제공되는 메모리를 비교하는데, 각 컴포넌트의 메모리 요구를 합산하고 일치하는지 여부를 검사한다. 한 컴포넌트의 메모리 요구는 다른 컴포넌트의 메모리 요구에 의존적이지 않다. 컴포넌트간(Inter-component) 검사는 컴포넌트 사이의 의존성을 검사한다. 예를 들어, Call graphs는 호출하는 컴포넌트가 현 시스템으로부터 빠져 있는지 여부를 검사하며, interface compatibility는 반환형(return type)이 호출 컴포넌트의 반환형과 호환성이 있는 지, 또는 입력 매개변수 리스트(input parameter list)가 호출 컴포넌트의 입력 매개변수 리스트에 호환성이 있는 지를 검사한다. exclusion requirements는 서로를 배제해야 하는 두 컴포넌트가 포함되지 않았는지를 검사한다.

3.2.4 스케줄러(Scheduler)

스케줄러는 모든 태스크가 그들의 deadline을 만족하는지 여부를 확인하며, Rate Monotonic, DM/offset과 같은 스케줄링 알고리즘을 사용하여 스케줄 한다. 그러나, 기존 스케줄링 분석 기법 중 어느 것도 모든 실시간 내장형 시스템에 적용이 불가능하다. 서로 다른 타입의 내장형 시스템을 다루기 위해, 기존 스케줄링 분석의 장점을 갖는 유연하고 확장할 수 있는 스케줄러를 제공한다. 스케줄러는 컴포넌트들을 조사하여 태스크의 특징들을 수집한 뒤, 일치하는 가정들이 있는 분석 기법을 선택한다. 이를 통해 시스템에 적절한 스케줄링 정책이 적용되는 것이 가능하다.

3.2.5 소스 코드 생성기(Source Code Generator)

이전에 기술된 모델러의 상세 컴포넌트 레벨 뷰는 최종 애플리케이션 코드를 생성하기 위해 사용된다. 소스 코드 생성기는 XML에 기반한 소스 코드 생성 시스템을 제공하며, 사용자에 의해 쓰여지는 코드를 줄이는 방법이다. 제공하는 정보는 클래스, 메소드, 클래스 사이의 관계와 같은 구조 정보와 기능 수행을 위한 클래스의 행위를 나타내는 행위 정보(pseudo code)로 나뉜다. 행위 정보는 개발자에 의해 추가 또는 변경된다. 구체적 타입과 구현 코드를 추가, 변경하여 XML로 기술된 구조 정보는 파싱되어 타겟 프로그래밍 언어로 소스 코드가 생성된다.



(그림 6) 소스 코드 생성 프로세스

XML 파서는 (그림 6)과 같이 입력으로 들어온 XML 구조 정보가 XML 문서 스펙에 맞게 쓰여있는 지를 검사하고, SAX를 이용해 특정 태그명을 지정하면 그 태그를 하위 노드를 찾는다. 그 결과물은 DOM 트리로 구축되고 이를 트리 형태로 보여준다. 각 프로그래밍 언어는 템플릿을 사용해 생성 정보를 기술하므로, 다른 언어로 변경 하기 위해서는 그 언어의 템플릿이 준비되어야 한다. 템플릿은 매크로 확장을 사용해 기술되며, 매크로명이 # 심볼 사이에 나타날 때마다 트리의 값으로 확장된다.

3.2.6 라이브러리(Library)

라이브러리는 계층적 조립을 위한 컴포넌트들을 저장하는 기반(base) 라이브러리와 기반 라이브러리의 컴포넌트들로 재구성된 내장형 실시간 제품을 저장하는 제품(product) 라이브러리로 구성된다. 기반 라이브러리는 컴포넌트 객체를 평면 구성 대신에 계층적인 분류로 구성하였다. 계층의 상단으로부터 하단으로 움직일 때, 컴포넌트 객체는 더욱 구체화 된다. 상단에서 객체는 추상적이며 컴포넌트의 집합을 표현한다. 하단에서 계층의 말단 노드는 완전하게 구체화된 컴포넌트를 표현한다.

라이브러리의 각 객체는 AND-OR 트리[7]를 사용하여 재구성 모델의 형태로 다중 행위 관점을 포함한다. 시스템에 컴포넌트를 포함할 때, 설계자는 컴포넌트 선택(component selection)과 모델 선택(model selection)이라는 두 단계를 완료해야 한다. 첫 번째 단계에서, 설계자는 특정 기능의 구현을 위해 어떤 컴포넌트를 사용할 것인지를 결정한다. 초기에, 이것은 컴포넌트의 집합을 표현하는 추상 객체가 될 수 있으며 후에 구체적인 인스턴스에 의해 대체될 것이다. 두 번째 단계에서, 설계자는 특정 애플리케이션환경에 적합한 컴포넌트 구현을 선택한다.

4. 결론

본 논문에서는 실시간 내장형 시스템 개발을 위해 PBO 모델에 기반한 개발 툴셋을 소개하였다. 본 논문에서 제안한 개발 툴셋은 PBO 모델에서 지원하지 못하는 구성과 분석 기능을 포함하는 6개의 소프트웨어 컴포넌트들로 구성되어, 증가된 신뢰성과 감소된 복잡도를 갖는 시스템 개발을 가능하게 한다. 현재 본 논문에서 제안한 개발 툴셋의 아키텍처를 근거로 성능, 경량화, 실시간성 요소들간의 이해득실을 고려한 컴포넌트기반 실시간 내장형 개발 툴셋이 구체적으로 설계, 구현되고 있다.

참고문헌

[1] Edward A. Lee, "What's Ahead for Embedded Software?", *IEEE Computer*, pp. 18-26, September 2000.

[2] Ivica Crnkovic, Magnus Larsson, Building Reliable Component-Based Software Systems, Artech House publisher, 2002.

[3] Damir Isovic, Markus Lindgren, "Real-Time Components", *Technical Report*, Malardalen Real-Time Research Centre, Malardalen University, March 2000.

[4] David B. Stewart, "Software Components for Real Time", *Embedded Systems Programming*, Vol. 13, No. 13, pp. 100-138, 2000.

[5] J. Stankovic, "VEST: A Toolset For Constructing and Analyzing Component-Based Operating Systems for Embedded and Real-Time Systems," *University of Virginia TR CS-2000-19*, July 2000.

[6] Aleksandra Tesanovic, et. al, "Embedded Databases for Embedded Real-Time Systems: A Component-Based Approach", *MRTC Technical Report*, 2002.

[7] Diaz-Calderon, A., Paredis, C. J. J. and Khosla, P. K., "Reconfigurable Models: A Modeling Paradigm to Support Simulation-Based Design.", *2000 Summer Computer Simulation Conference*. Vancouver, Canada. Society for Computer Simulation.