제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

내장형 실시간 기반 자바가상머신 개발을 위한 컴포넌트 모델에 관한 연구

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

A Study on Component Model for Java Virtual Machine Development based on Embedded Real-Time

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

요 약

최근 이동내장형 시스템 기술이 차세대 정보통신 산업의 주력으로 급부상함에 따라 다양한 이기종 제품간의 호환성과 이식성 제공의 핵심 역할을 수행하는 자바가상머신(Java Virtual Machine)이 주목 받고 있다. 가상머신을 컴포넌트 기반 소프트웨어 기술을 사용하여 구축한다면, 재사용과 재구성이 용이해 저렴하고 신뢰성 있는 가상머신 개발이 가능해진다. 이러한 이유로 인하여, 자원이 제한된 내장형 실시간 시스템 개발 사용에 적합한 컴포넌트 모델의 필요성이 동시에 증대되고 있다. 따라서, 본 논문에서는 기존에 제안된 내장형 실시간 컴포넌트 모델인 PBO(Port-Based Object) 모델을 내장형 실시간 환경에서 수행되는 자바가상머신에 적합하도록 개선하여 증가된 신뢰성과 감소된 시스템 복잡도를 갖는 자바가상머신 시스템을 제안하였다.

1. 서론

[1]최근 컴퓨터, 유무선 통신 기술이 빠르게 발전함에 따라, 언제 어디서나 다양한 서비스를 제공받을 수 있는 이동통신 기술이 차세대 정보통신 산업의 주력으로 급부상하게 되었고, 이에 따라 이동성을 제공하는 내장형 시스템 소프트웨어 기술의 확보가 필수적인 요건이 되었다. 이동성을 가지는 내장형 시스템은 자원 제약이 심한 환경에서 이동성, 실시간성, 융통성, 이식성을 지원해야 되고 멀티미디어 서비스 제공을 요구받는다. 그러나, 현재의 내장형 운영체제들이 인터넷 연결이나 플랫폼 독립적인 프로그램 다운로드 등과 같은 기능을 충분히 지원하기 어렵기 때문에 플랫폼 독립성과 이식성을 충실히 지원하는 환경을 제공하는 자바가상머신(Java Virtual Machine: JVM)이 이동 내장형 시스템에서 여러 가지 응용 소프트웨어를 지원하는 핵심 역할을 수행하게 되었다.

이러한 변화와 아울러 소프트웨어도 운영체제, 미들웨어, 응용 수준에 이르는 모든 영역에서 재구성, 재사용, 적응성을 효과적으로 지원할 수 있는 컴포넌트기반 소프트웨어 개발 방법론(Component-based Software Engineering: CBSE)이 주목받고 있다. 컴포넌트기반 개발 방법론은 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. 따라서, 가상머신이 실시간 컴포넌트 기반 미들웨어 기술을 사용하여 구성되어 진다면, 소프트웨어 재사용과 재구성을 통해 빠르고 신뢰성 있는 가상머신 개발이 가능해진다. 이 때문에, 현재 내장형 실시간 시스템 개발에 적용하려는 연구가 진행중이다[1,2]. 그러나, 내장형 실시간 환경의 자바가상머신 개발에 적합한 내장형 실시간 시스템 모델은 없는 실정이며, 그 대안이 되는 컴포넌트 모델을 필요로 하고 있다.

따라서, 본 논문에서는 내장형 실시간 환경에서 수행되는 자바가상머신에 적합하도록 기존의 내장형 실시간 컴포넌트 모델인 PBO(Port-Based Object)를 개선한 모델을 제안한다. 개선된 PBO 모델은 기존의 PBO 모델에서 제공하지 못하는 계층적 조립과 검증 방안을 지원하며, 이를 통해 본 논문에서는 자바가상머신을 계층적으로 조립할 수 있도록 구성하며, 전체 시스템의 시간 제약을 검증하는 방안을 제시하였다.

2. 관련 연구

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

2.1 컴포넌트기반 내장형 실시간 모델

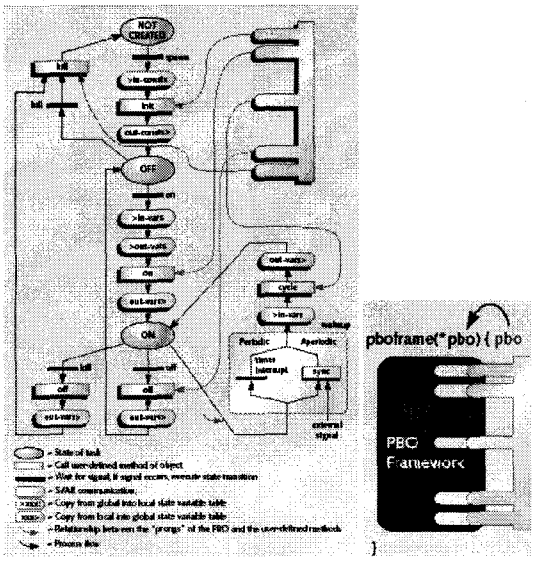
실시간 컴포넌트를 갖는 시스템 개발[3]은 Isovic등이 복잡도에 상관없이 실시간 시스템을 대상으로 하는 개발 방법을 정의하였다. 이 개발 방법에서 컴포넌트 라이브러리는 COTS 컴포넌트와 메모리 소비, id, 환경, 기능과 같은 명세뿐만 아니라 다른 컴포넌트에 대한 의존성을 포함한다. 컴포넌트는 태스크에 매

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

핑되며, 컴포넌트간의 통신은 공유 메모리를 통해 이루어진다. 개발 프로세스는 시스템 명세, 상위 레벨 설계, 상세 설계, 스케쥴링/인터페이스 검사의 여러 단계로 나뉘어 진다. 상위 레벨 설계에서는 시스템 명세의 입력으로 시작되며 라이브러리의 컴포넌트 후보들로 시스템을 설계한다. 상세 설계에서는 period, release time, precedence constraints, deadline, mutual exclusion과 같은 시간 속성이 컴포넌트에 할당된다. 검사 단계는 선택된 컴포넌트의 시스템 적합 여부, 새로운 컴포넌트 개발 필요성, 인터페이스의 타입 매치 여부를 알기 위해 수행된다. 위의 개발 프로세스는 시스템 조립을 위한 적합한 컴포넌트가 발견될 때까지 반복될 수 있다.

VEST(Virginia Embedded Systems Toolset)[4]는 미국 버지니아 대학의 Stankovic 교수팀이 컴포넌트기반 내장형 실시간 시스템의 개발, 구현, 평가를 개선할 목적으로 개발되고 있으며, 조립과 분석 아키텍쳐를 포함한다. 또한, VEST는 컴포넌트 기반 실시간 내장형 시스템을 개발, 분석하기 위한 통합 환경이며, 소프트웨어, 하드웨어 컴포넌트를 포함하는 다양한 라이브러리와 컴포넌트 조립이나 의존성 검사를 수행하거나 컴포넌트를 프로세서나 하드웨어에 매핑하는 구성 도구(configuration tool)와 실시간, 신뢰성, 스케쥴링 분석을 수행하는 분석 도구를 포함하는 대화식 개발 도구로 구성된다. VEST의 단점으로는 인터페이스 문제를 전혀 기술하지 않았으며, 시스템의 분석을 제안하였으나 구체적인 방법은 제시하지 않았다.

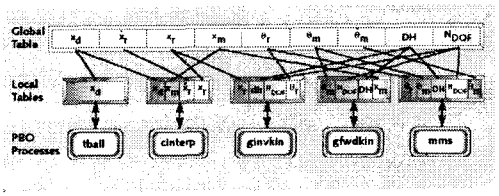
PBO(Port-based object)[5]는 Carnegie Mellon Univ에서 개발하였으며, 내장형 실시간 제어 시스템의 개발을 위한 컴포넌트 모델이다[4]. PBO의 데이터 흐름은 입출력 포트를 통하여 규정하며, 컴포넌트 기반 소프트웨어 지원은 프레임워크 프로세스(framework process)라 불리는 단일의 표준 프로세스를 생성함으로서 구체화 된다. 프레임워크는 인자로서 PBO를 취하며, 세 가지 상태(NOT_CREATED, ON, OFF)를 갖는 유한 상태기계로 구성된다. 또한 PBO가 필요로 하는 모든 선언을 포함하는 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[1].

<표 1> 컴포넌트기반 내장형 실시간 모델의 특징들

x : 전체 지원 x/p : 부분 지원

특징 \ 모델		[3]	VEST[4]	PBO[5]
실시간 속성	not preserved			
	preserved	x	x	x/p
인터페이스/통신	standardized			
	system specific			
	ports/unbuffered	x	-	x
구성 툴	not available			
	available	x/p	x	x/p
분석 툴	not available			x
	available	x	x/p	
재사용성	component	x	x	x
	architecture		x	
개조 능력	none			
	low			
	moderate			
	high	x	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. Isovic등에 의해 소개된 개발 방법에서 컴포넌트는 열거된 시간 속성을 모두 가지며, 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 가능하며, 이는 VEST를 가장 유연한 내장형 실시간 접근법으로 만든다.

둘째, 인터페이스/통신. 컴포넌트는 잘 정의된 인터페이스를 통해 다른 컴포넌트와 통신을 하며, 실시간 컴포넌트 통신은 메시지 패싱을 통해 행해지는 버퍼(buffered) 통신과 공유 메모리

를 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. Isovic에 의해 소개된 컴포넌트 기반 실시간 시스템을 위한 개발 방법은 시스템 조립을 위해 컴포넌트의 시간 속성과 인터페이스 검사와 같은 적합한 컴포넌트를 선택하기 위한 구성 지원을 제공하며, 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. Isovic등에 의해 소개된 개발 방법에서는 WCET 검사만 수행되며, VEST는 시스템의 신뢰성과 실시간 분석 표기를 소개하나 상세한 내용은 제공하지 않고 있다.

다섯째, 재사용성. 시스템은 라이브러리의 재사용 컴포넌트로 조립되며, 시스템의 아키텍쳐는 시스템의 개발에 재사용 될 수 있다. 따라서, 내장형 실시간 시스템에서 재사용할 수 있는 부분은 컴포넌트와 아키텍쳐이다. 이중 모든 시스템에서 재사용될 수 있는 부분은 컴포넌트이며, VEST는 아키텍쳐도 재사용될 수 있으므로 높은 수준의 재사용성을 가진다.

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

3. 자바가상머신 아키텍치

자바가상머신은 일차적으로 클래스 로더(Class Loader)에 의해 수행중인 머신에 클래스 파일들을 로드하고 링크하며, 수행엔진(Execution Engine)과 NMI(Native Method Interface)는 다수의 런타임 데이터 구조에 의해 내부적으로 클래스들을 표현하고 수행을 돕는다[6]. 자바가상머신의 메인 컴포넌트는 <표 2>와 같이 Class Loader, Security Manager, Memory Manager, Execution Engine, Input/Output, Communication, Scheduler, Native Interface로 분류할 수 있다. Class Loader는 지역 파일 시스템과 네트워크와 같은 다양한 소스들로부터 애플리케이션과 라이브러리 클래스들을 동적으로 로드하기 위해 사용되며, Memory Manager는 다른 내부 가상머신 데이터 구조들을 저장하기 위해 사용되는 메모리를 관리하고 객체 인스턴스들을 위한 가비지 컬렉트 된 힙을 제공한다. Execution Engine은 가상머신의 핵심부이며 클래스 로더를 통해 로드된 바이트코드 명령들을 실행하며, 마지막으로 Native Interface는 가상머신이 애플리케이션과 클래스 라이브러리에서 비 자바 루틴을 호출할 수 있도록 허용한다.

<표 2> 자바가상머신 컴포넌트들

컴포넌트	서브컴포넌트	기 능
Class Loader	Loader	클래스파일을 표현하는 바이트 배열 획득
	Verifier	클래스파일의 구조적 적형성과 허용되지 않은 연산들을 수행하지 않도록 내용 검사
	Preparer	정적 클래스 필드를 위한 저장 공간의 할당과 디폴트 초기화
	Resolver	클래스파일내의 심볼 참조는 실행시간에 직접 참조
	Initializer	클래스의 초기화 메소드 실행
Security Manager	Encryption	암호화 관련 기능
	Authentication	인증 관련 기능
	Digital Signature	전자서명 관련 기능
Memory Manager	Initializer	메모리 초기화
	Garbage Collector	더 이상 참조되지 않는 메모리 공간 회수
Execution Engine	Interpreter – Bytecode Verifier	클래스 로더로부터 전달받은 바이트코드의 적합성 검증
	Interpreter – Code Analysis	검증된 바이트코드가 요구하는 명령 분석
	Interpreter – Bytecode Execute	바이트코드에 해당하는 명령 수행
	JIT Compiler	한번 번역된 기계어를 기억하여 재실행할 경우 번역 없이 실행
Input	Keyboard	입력 장치 지원
	Mouse	
	TouchPen	
	TouchScreen	
	Mic	
Output	Speaker	출력 장치 지원
	TouchScreen	
Communication	WAP	WAP 프로토콜 지원
	HTTP	HTTP 프로토콜 지원
Scheduler	-	실시간 스케쥴링 알고리즘을 적용한 스케쥴링 수행
Native Interface	-	가상머신이 애플리케이션과 클래스 라이브러리에서 비 자바 루틴을 호출할 수 있도록 허용

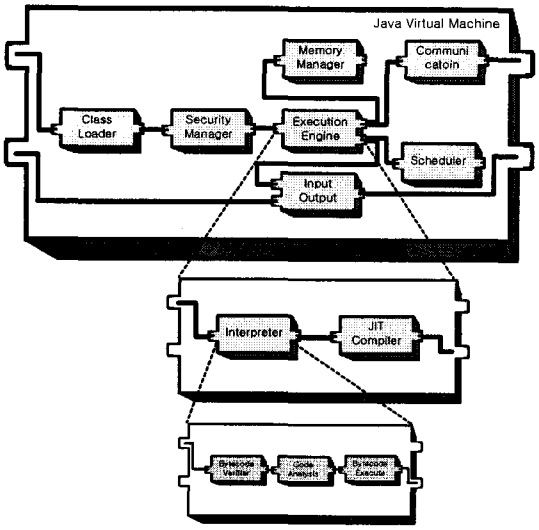
4. 개선된 PBO(Port-Based Object) 모델

기존의 PBO 모델[7]은 재사용 컴포넌트 개발의 필수인 계층적 조립 개념이 없으며, 시간 속성들은 있으나 시간 제약을 표현하고 검증하는 능력이 약한 문제점이 있다. 따라서, 본 논문에서는 자원이 제한된 내장형 실시간 환경에서 수행되는 자바가상머신 개발에 적합하도록 PBO 모델을 개선한 컴포넌트 모델을 제안하였으며, 계층적 조립과 시간 제약 검증에 기반하여 비교하였다.

4.1 계층적 조립(Hierarchical composition)

컴포넌트의 계층적 조립은 재사용 컴포넌트 구축을 위해 필수적이며, 통합되고 단일의 인터페이스를 갖는 몇 개의 작은 컴포넌트들을 조립함으로서 지원된다. 기존의 PBO는 계층적 조립 개념이 없으나, 제안한 모델은 포함하였다.

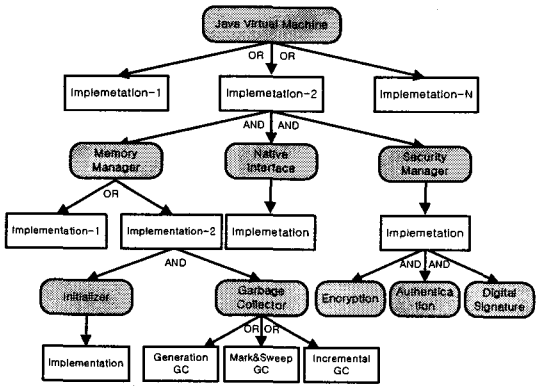
본 논문에서 제안한 모델은 하위 레벨의 컴포넌트들에 의해 제공되는 서비스를 사용하며, 더 작은 서브컴포넌트로 나누어 질 수 없는 단계에서 종료된다. 시스템은 상위 레벨 컴포넌트와 그들 사이의 상호작용에 의해 기술되며 상위 레벨 컴포넌트는 반복적으로 재정의 되므로, 시스템 모델은 점차적으로 더욱 상세한 설계 특성이 알려지도록 적용된다. 또한, 기존의 PBO 모델에 재구성 모델의 특징인 인터페이스와 구현의 분리를 고려하였다. 두 개념을 독립적으로 고려함으로서, 하나의 단일 인터페이스에 서로 다른 구현을 연관시키는 것이 가능하고, 결과적으로 재구성 모델(reconfigurable model) 생성이 가능하다.



(그림 3) 자바가상머신 컴포넌트의 계층적 조립

(그림 3)은 자바가상머신 시스템을 Class Loader, Native Interface, Execution Engine, Memory Manager, Security Manager등의 서브 컴포넌트들의 조립으로 계층적으로 정의하였으며, 그중 Execution Engine은 Interpreter와 JIT Compiler 컴포넌트의 조립으로, 그리고 Interpreter는 Bytecode Verifier, Code Analysis, Bytecode Execute로 구성한 그림이다.

구현은 하나의 일치하는 인터페이스를 가지나, 단일 인터페이스가 연관된 다중 구현을 가질 수도 있으며, AND-OR 트리[8]로서 표현될 수 있다. AND 아크는 하나의 구현으로부터 조립되는 인터페이스들을 가리키며, OR 아크는 하나의 인터페이스로부터 실체화될 수 있는 구현들을 가리킨다. 먼저 상위 레벨의 인터페이스에 하나의 구현을 할당해야 하며 선택된 구현을 구성하는 각각의 인터페이스에도 순환적으로 할당하는 방식이다.



(그림 4) 자바가상머신 시스템의 AND-OR 트리 표현

예를 들면, (그림 4)의 AND-OR 트리는 자바가상머신의 일부 모델을 기술한다. 상위 레벨의 인터페이스는 세 개의 OR 아크로 표현되며 연관된 서로 다른 세 개의 구현을 갖는다. implementation-2 모델 구현은 세 개의 AND 아크를 갖으며, 세 개의 인터페이스들(Memory Manager, Native Interface, Security Manager)로 구성된 합성 모델이다.

4.2 시간 제약의 검증(Verification)

검증은 실시간 시스템에서 시간 제약 만족 여부를 확인하는 필수 부분이다. 기존 PBO 모델은 개별 컴포넌트의 시간 제약만을 정의하였으나, 개선된 PBO 모델에서는 컴포넌트 사이에서 예측 가능한 요구 뿐만 아니라 다른 컴포넌트에 의한 비주기적인 호출, 즉 이벤트 인터럽트를 처리하는 부분까지 고려하였다.

자바가상머신을 구성하는 각 PBO 컴포넌트들은 동적 행위를 정의하기 위해 PBO 프레임워크를 가지며, 프레임워크내에서는 기존 PBO 모델의 프레임워크와 달리 주기적 또는 비주기적인 이벤트 인터럽트를 처리하는 부분을 추가하였다. PBO 프레임워크사이에 통과되는 이벤트는 다음과 같이 Call-Graph G = (V, E)로서 표현될 수 있다. 여기서 V는 PBO 프레임워크이며, E는 PBO 프레임워크사이의 이벤트 호출 관계이다. Call-Graph로 표현된 이벤트 호출 관계를 통해 부여된 시간 제약을 만족하는지 여부를 검사하면 자바가상머신의 예측성과 검증 능력을 향상시킬 수 있다.

5. 결론

본 논문에서는 내장형 실시간 환경에서 수행되는 자바가상머신 개발에 적합한 개선된 PBO 모델에 대하여 소개하였다. 본 논문에서 제안한 컴포넌트 모델은 기존의 PBO 모델이 지원하지 않는 계층적 조립과 검증 방안을 지원함으로서 증가된 신뢰성과 감소된 복잡도를 갖는 자바가상머신 개발을 가능하게 하며, 이에 기반하여 자바가상머신의 컴포넌트들을 기능별로 구성하고 전체 시스템의 시간 제약을 검증하는 방안을 제시하였다.

현재 본 논문에서 제안한 컴포넌트 모델을 기반으로 하는 성능, 경량화, 실시간성 요소들간의 이해득실을 고려한 컴포넌트기반 자바가상머신 개발 툴셋이 설계, 구현되고 있다.

참고문헌

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

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

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

[4] 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.

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

[6] Bill Vernners, "Inside the Java Virtual Machine", McGraw-Hill, 1999.

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

[8] 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.