제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

컴포넌트기반 자바가상머신 개발 툴셋 설계

서영준, 이승룡, 송영재
경희대학교 전자계산공학과
e-mail:yjseo@cvs2.khu.ac.kr

A Design of Toolset for Component-based Java Virtual Machine Development

Young-Jun Seo, 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, 컴포넌트기반 내장형 실시간 시스템을 효과적이고 체계적으로 개발하고 검증하기 위한 개발 툴셋의 중요성도 동시에 증대되고 있다. 따라서, 본 논문에서는 PBO(Port-Based Object) 모델을 확장하여 자바가상머신 컴포넌트들을 툴셋에서 제공하는 구성 툴에 의해 조립과 검사를 수행하며, 빠르게 재사용성과 신뢰성을 지원하는 자바가상머신을 개발할 수 있는 환경을 제안한다.

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, 응용수준에 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[1,2,3]. 그러나, 컴포넌트기반 내장형 실시간 시스템을 개발하기 위한 효율적인 방법이 부족한 실정이며, 그 대안으로 컴포넌트기반 개발 툴셋을 필요로 하고 있다.

따라서, 본 논문에서는 컴포넌트기반 개발 방법론을 적용한 자바가상머신 개발 툴셋을 제안한다. JVM은 내장형 실시간 시스템을 위한 컴포넌트 모델인 PBO에 기반하여 개별 컴포넌트들로 분리, 저장되며, 개발 툴셋을 구성하는 구성 툴에 의해 선택, 조립된다. 조립 과정마다 구성 툴의 의존성 검사를 통해 컴포넌트 사이의 의존성을 검사하며, 이를 통해 조립과정의 신뢰성을 증가시킬 수 있다. 최종적으로 재구성된 JVM은 분석 툴을 통해 실시간성, 스케쥴링 분석을 수행, 검증된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 내장형 실시간 컴포넌트 모델인 PBO 모델과 개발 툴셋에 대해 소개하고, 3장에서는 본 논문에서 제안하는 컴포넌트기반 자바가상머신 개발 툴셋의 설계 방안을 기술한다. 마지막으로 4장에서는 결론 및 향후 연구방향에 대해 언급하였다.

2. 관련 연구

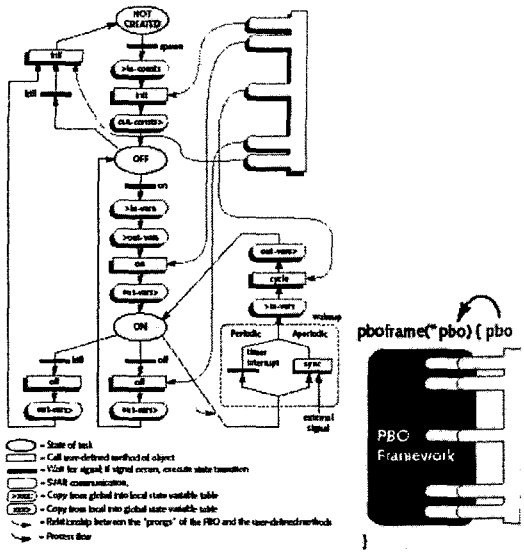
본 장에서는 컴포넌트기반 내장형 실시간 시스템을 위한 컴포넌트 모델인 PBO와 관련 개발 툴셋에 대하여 살펴보도록 하겠다.

2.1 PBO(Port-based object) 모델

PBO는 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

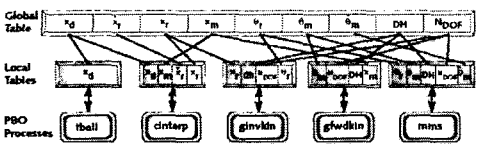
1 본 연구는 한국과학재단 목적기초연구(과제번호 : R01-2001-000-00357-0) 지원으로 수행되었음.

(NOT_CREATED, ON, OFF)를 갖는 유한 상태기계로 구성된다. 또한 PBO가 필요로 하는 모든 선언을 포함하는 매크로 확장을 통해 구현된다. (그림 1)은 프레임워크에 플러그인(plugs in) 되는 PBO와 프레임워크 프로세스의 내부를 기술하는 프로세스 플로우 다이어그램을 나타낸다.



(그림 1) PBO 프레임워크의 프로세스 플로우 다이어그램

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

PBO 모델은 재사용 컴포넌트 개발의 필수인 계층적 조립 개념이 없으며, 시간 속성들은 있으나 시간 제약을 표현하는 능력이 약한 문제점이 있다. 따라서, JVM 컴포넌트 개발 툴셋의 개발과 동시에 PBO 모델의 문제점들을 보완해 주는 연구가 필요하다.

2.2 컴포넌트기반 내장형 실시간 시스템 개발 툴셋

컴포넌트기반 내장형 실시간 시스템 개발 툴셋으로는 VEST, MetaH, CIP등이 있으며, 각각의 특징은 다음과 같다.

VEST(Virginia Embedded Systems Toolset)[5]는 미국 버지니아 대학의 Stankovic 교수팀이 컴포넌트기반 내장형 실시간 시스템의 개발, 구현, 평가를 개선할 목적으로 개발되고 있으며, 조립과 분석 아키텍쳐를 포함한다. 또한, VEST는 컴포넌트기반 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.

MetaH[6]는 미국의 Honeywell사에서 개발한 신뢰성 있는 실시간의 다중 프로세서 항공공학 시스템을 개발하기 위한 언어이자 툴셋이다. MetaH 툴셋은 그래픽, 텍스트 포맷으로 MetaH 명세를 변환하며, 자동으로 실행 이미지를 생성한다. 또한 실시간 스케쥴가능성, 신뢰성, 안전/보안 모델링 분석 도구와 같은 분석 도구를 사용한 다양한 분석을 수행한다. MetaH의 특징으로는 첫째, 진통적인 언어로 쓰여진 코드 모듈을 애플리케이션을 형성하기 위해 조립하는 방법을 규정한다. 둘째, 특정 HW 시스템의 구조를 규정한다. 셋째, 소프트웨어가 하드웨어에 할당되는 방법을 규정한다. 넷째, 특정한 종류의 객체를 제공하며 다양한 분석을 수행한다.

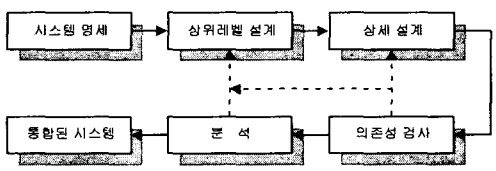
CIP(Communicating Interacting Processes)[7]는 스위스의 CIP System AG사에서 개발한 방법론(method)이며, 툴셋이다. CIP 방법론은 내장형 시스템을 위한 모델 기반 소프트웨어 개발 방법이며, CIP 툴셋에 의해 개발된다. CIP 툴셋은 자동 코드 생성기와 검증 함수를 갖는 그래픽 모델링 프레임워크이며, CIP 모델을 실행할 수 있는 CIP 컴포넌트로 변환한다. CIP 방법론을 가지고, 내장형 시스템은 확장된 유한 상태 머신으로서 설계된 프로세스들의 구조와 행위 모델들에 의하여 규정된다. CIP 모델은 채널, 프로세스, 메시지 또는 오퍼레이션과 같은 모델링 요소들로 구성되며, 이러한 모델링 요소들은 CIP 객체라고 불리운다. CIP 툴셋은 유연한 방법으로 이들 요소들을 생성, 조립, 연결하도록 허용한다. 툴의 편집기 프레임워크는 CIP 모델의 조립적인 구조를 반영하며, 한 편집기에서의 이루어진 변화는 즉시 모든 의존적인 모델의 부분과 뷰에 갱신되므로, 일관성 있는 CIP 모델을 구축하는 것을 허용한다.

3. 컴포넌트기반 자바가상머신 개발 툴셋

본 논문에서 제안한 JVM CDT(JVM Component Development Toolset)는 컴포넌트기반 내장형 실시간 시스템인 JVM을 개발, 분석하기 위한 통합 환경이며, PBO에 기반하여 설계된 JVM 컴포넌트의 선택과 컴포넌트간의 비주얼한 조립, 그리고 의존성 검사(dependency check)와 분석(analysis)을 수행한다.

3.1 개발 프로세스

실시간 컴포넌트 개발 프로세스[8]를 기반으로 재구성된 컴포넌트기반 자바가상머신을 위한 개발 프로세스는 (그림 3)에서 보여주는데로 몇 단계로 나뉘어진다.



(그림 3) 컴포넌트기반 JVM을 위한 개발 프로세스

상위 레벨 설계(top-level design)에서는 시스템 명세(system specification)를 입력받아 라이브러리의 컴포넌트들로 시스템 조립이 수행된다. 설계자는 컴포넌트 라이브러리를 통해 브라우징

하고 가능한 컴포넌트 후보들로 시스템을 설계할 수 있다. 상세 설계(detailed design) 단계에서는 period, release time, deadline 과 같은 시간적 속성이 컴포넌트에 할당된다. 의존성 검사(dependency check) 단계는 조립 과정동안의 컴포넌트 사이의 의존성을 검사하며, 잘 정의된 검사는 발생 가능한 에러를 줄이기 때문에 조립된 시스템의 신뢰성을 증가시킨다. 분석(analysis) 단계에서는 개발중인 전체 시스템이 실시간, 신뢰성, 스케쥴링과 같은 기능외의 요구사항을 만족하는 지를 검사한다. 상위 레벨 설계, 상세 설계, 의존성 검사, 분석 단계는 최종적으로 검증된 JVM이 나올 때까지 반복될 수 있다.

3.2 자바가상머신의 컴포넌트화

본 논문에서는 PBO 모델을 적용하여 JVM을 세부 컴포넌트로 설계하였다. 기능별로 분리 설계된 JVM 컴포넌트는 각각 입출력 포트와 시간 파라미터등을 정의한 후, 프레임워크와 플러그인되는 함수 집합(init, on, cycle. off)의 파라미터와 내부 수행 코드를 정의한다. <표 1>과 <표 2>는 JVM을 구성하는 클래스 로더, 실행 엔진, 메모리 관리자, 스케쥴러/쓰레드 관리자등의 컴포넌트들과 대체 가능한 서브컴포넌트들의 구성과 기능을 나타낸다.

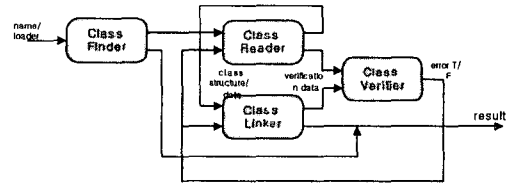
<표 1> JVM 컴포넌트의 구성

컴포넌트	기 능
Class Loader	JVM내에서 원하는 클래스를 요청된 상태로 만들어 줌
Execution Engine	사용자의 프로그램에서 요구하는 작업에 대한 바이트코드를 클래스 로더를 통해 전달받아 수행
Memory Manager	최초 JVM이 구동할 때, 미리 정의된 힙 메모리 크기만큼 운영체제로부터 할당받으며, 매번 메모리 할당 요청이 있을 때마다 가비지 컬렉션 작업 수행
Scheduler /Thread Manager	스케쥴링과 쓰레드의 생성 및 관리를 담당

<표 2> JVM 서브컴포넌트의 구성

컴포넌트	서브 컴포넌트	기 능
Class Loader	Class Finder	로드된 클래스 리스트에서 클래스 찾음
	Class Reader	클래스 파일에서 클래스 데이터를 읽음
	Class Linker	클래스 객체 생성, 데이터구조 초기화 후 기존 심볼릭 레퍼런스를 직접 참조로 매핑
	Class Verifier	클래스 바이트코드 검증 수행
Execution Engine	Bytecode Verify	Class Loader로부터 전달받은 바이트코드 검증
	Code Analysis	검증 받은 바이트코드가 요구하는 명령 분석
	Bytecode Execute	바이트코드에 해당하는 명령 수행
Memory Manager	Memory Initializer	JVM이 최초 구동할 때, 미리 설정된 환경 변수 값에 따라 힙 메모리를 초기화
	Memory Allocator (Garbage Collector)	메모리 할당 요청을 처리 (더 이상 참조되지 않는 객체(garbage)들이 차지하는 메모리 공간 회수)
Scheduler /Thread Manager	Scheduler	실시간 스케쥴링 알고리즘을 적용한 스케쥴링
	Thread Manager	실시간 및 비실시간 쓰레드의 생성 및 관리, 제어 담당

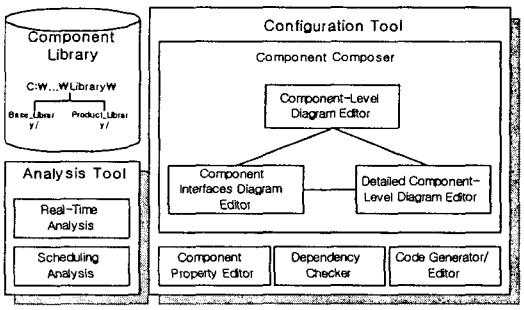
(그림 4)는 PBO 모델을 적용한 클래스 로더를 나타낸 것이다. 클래스 로더는 <표 2>와 같이 class finder, class reader, class linker, class verifier의 세부 컴포넌트들로 구성되며, 다음과 같은 수행 순서를 갖는다. class finder에서 현재 로드된 클래스 리스트에서 클래스를 찾으며, 이미 로딩되어 있는 경우 다른 과정 없이 바로 위치를 넘겨 받을 수 있다. 만약, 로딩되어 있지 않으면 소스가 되는 클래스 파일을 찾으며 class reader에서 클래스 자료를 읽는다. class reader의 결과는 class linker에서 클래스 객체 생성과 데이터 구조 초기화, 그리고 기존 심볼릭 레퍼런스를 직접 참조로 매핑하는 resolution 과정을 수행한다. 마지막으로 class verifier에서는 class reader와 linker에서 넘겨 받은 바이트 코드의 검증을 수행하며, 모든 클래스 로더 컴포넌트의 결과물로 클래스 구조 포인터를 얻을 수 있다.



(그림 4) PBO모델에 기반한 클래스 로더

3.3 시스템 구조

JVM CDT는 (그림 4)에서처럼 JVM 구성(configuration), 분석(analysis) 지원 툴과 라이브러리(library)로 구성된다.



(그림 5) JVM CDT 아키텍처

3.3.1 JVM 구성 툴

구성 툴에서는 크게 JVM 컴포넌트의 선택과 계층적 조립을 수행하는 컴포넌트 조립기와 의존성 검사기, 그리고 코드 생성/편집기로 나뉘어 지며, 조립기의 다이어그램 편집기(컴포넌트 레벨 다이어그램, 컴포넌트 인터페이스 다이어그램, 상세 컴포넌트

레벨 다이어그램 편집기)간에는 변경이 이루어질 때 동기화를 지원한다. 각 모듈들의 기능은 <표 3>과 같다.

<표 3> JVM 구성 툴의 모듈별 기능

모 듈	기 능
컴포넌트 레벨 다이어그램 편집기	라이브러리로부터 선택된 컴포넌트들의 조립으로 JVM을 표현하며, 컴포넌트간의 관계를 지시하는 의존관계 정의
컴포넌트 인터페이스 다이어그램 편집기	컴포넌트 레벨 다이어그램의 의존관계에 대해 생성되며, 설계자는 각 컴포넌트의 포트 정의 및 컴포넌트간의 관계를 구체적으로 명시
상세 컴포넌트 레벨 다이어그램 편집기	컴포넌트 레벨 다이어그램의 모든 컴포넌트에 대해 생성되며, 컴포넌트의 내부와 포트가 연결되는 방법 정의
컴포넌트 속성 편집기	컴포넌트 명세 정보의 생성 및 수정
의존성 검사기	컴포넌트간 조립 과정마다 VEST의 4가지 타입의 의존성 검사 리스트를 통해 의존성 검사
코드 생성/편집기	조립된 JVM의 코드 생성 및 편집 지원

3.3.2 JVM 분석 툴

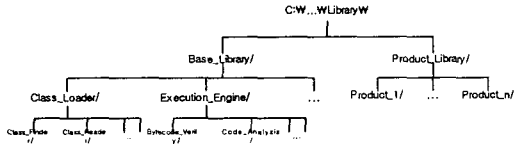
조립된 시스템의 행위를 검증하기 위해서 분석 툴이 필요하며, 특히 실시간 시스템은 시간 속성을 만족해야 하므로, 적합한 분석이 요구된다. 따라서, 실시간 시스템인 JVM CDT에서는 조립된 JVM의 실시간, 스케쥴링 분석을 수행하는 분석 툴이 지원된다. 분석 툴의 하부 모듈과 그 기능은 <표 4>와 같다.

<표 4> JVM 분석 툴의 모듈별 기능

모 듈	기 능
실시간 분석	WCET, deadline등의 시간 속성 만족 여부 분석
스케쥴링 분석	RM(Rate-Monotonic) 분석을 통한 JVM의 스케쥴링 가능 여부 분석

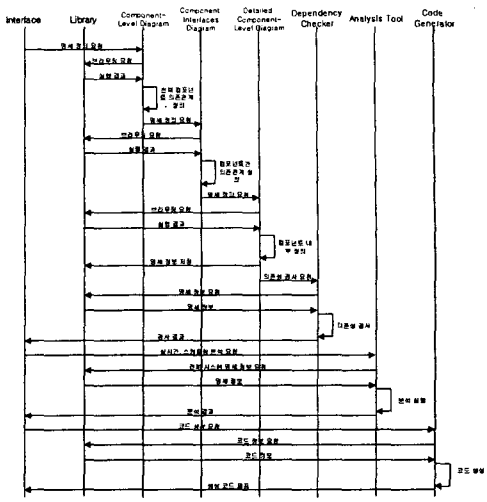
3.3.3 라이브러리

JVM CDT의 라이브러리는 계층적인 디렉토리 구조를 가지며, JVM의 계층적 조립을 위한 컴포넌트들을 저장하는 기반(base) 라이브러리와 기반 라이브러리의 컴포넌트들로 재구성된 JVM 제품을 저장하는 제품(product) 라이브러리로 구성된다. 예를 들어, execution engine은 bytecode verify, code analysis, bytecode execute를 포함하는 디렉토리로 구성되며, (그림 6)은 JVM CDT의 라이브러리 구조를 보여준다.



(그림 6) 컴포넌트 라이브러리 구조

(그림 7)은 JVM CDT의 구성 모듈간의 JVM 개발 과정을 나타낸다.



(그림 7) JVM CDT 구성 모듈간의 Sequence Diagram

4. 결론

본 논문에서는 컴포넌트기반 자바가상머신 개발 툴셋의 설계에 대하여 소개하였다. 본 논문에서 제안된 개발 툴셋은 PBO 모델에 기반하여 설계된 JVM 컴포넌트들을 사용하였으며, 저장된 컴포넌트들을 툴셋에서 제공하는 구성 툴에 의해 조립과 검사를 수행함으로서 빠르게 재사용성과 신뢰성을 만족하는 JVM을 개발할 수 있도록 하였다.

제안된 개발 툴셋은 성능, 경량화, 실시간성 요소들간의 이해득실을 고려한 최적화된 JVM 컴포넌트 모델링 기법에 관한 추가 연구가 더 필요하다. 또한, 각 구성 모듈에 대한 구체적인 명세 정의와 함께 시제품을 구현할 예정이다.

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] Damir Isovic, Markus Lindgren, Ivica Crnkovic, "System Development with Real-Time Components", *Proc. of ECOOP2000 Workshop 22 - Pervasive Component-based systems*, Sophia Antipolis and Cannes, June 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] Honeywell Technology Center, "MetaH User's Manual", technical report, 1998.

[7] Hugo Fierz, "The CIP Method: Component-and Model-Based Construction of Embedded Systems", *European Software Engineering Conference 1999 - ESEC 99*, LNCS Vol. 1687, pp. 374-391, 1999.

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