서비스로봇의 신뢰성 향상을 위한 OPRoS 기반 Fault-tolerance 기법

OPRoS based Fault Tolerance Support for Reliability of Service Robots

안 희 준*, 이 동 수, 안 상 철
(Heejune Ahn, Dongsu Lee, and Sang Chul Ahn)

Abstract: For commercial success of emerging service robots, the fault tolerant technology for system reliability and human safety is crucial. Traditionally fault tolerance methods have been implemented in application level. However, from our studies on the common design patterns in fault tolerance, we argue that a framework-based approach provides many benefits in providing reliability for system development. To demonstrate the benefits, we build a framework-based fault tolerant engine for OPRoS (Open Platform for Robotic Services) standards. The fault manager in framework provides a set of fault tolerant measures of detection, isolation, and recovery. The system integrators choose the appropriate fault handling tools by declaring XML configuration descriptors, considering the constraints of components and operating environment. By building a fault tolerant navigation application from the non-fault-tolerant components, we demonstrate the usability and benefits of the proposed framework-based approach.

Keywords: OPRoS (Open Platform for Robotic Services), fault tolerance, design pattern, robot software framework

I. 서론

최근 서비스로봇에 대한 개발과 상용화에 대한 관심이 증가하고 있다. '서비스로봇'이란 제조업을 제외한 분야에서 반자율 또는 완전 자율로 인간 및 시설에 대한 서비스를 제공하는 로봇을 의미한다[1]. 정의에서 추론할 수 있듯이 서비스로봇은 특정 서비스에 국한되지 않고 다양한 서비스가 지속적으로 개발되어 추가될 수 있어야 한다. 이러한 관점에서 컴포넌트 기반 방식은 서비스 로봇의 확장성 요구사항에 적합한 개발 방식으로 웹 응용, WIPI 등 다양한 통신서비스 프레임워크에서 그 유용성이 입증이 되었으며, 최근 로봇시스템에도 한국의 OPRoS[2,3], OMG의 RTC (Robot Technology Component)[4], 및 마이크로소프트 사의 MSRDS (Microsoft Robotics Developer Studio)[5] 등 주요한 로봇 프레임워크가 시스템 모델로 사용하고 있다.

한편, NASA의 화성탐사 로봇과 2009년 국내 로켓 발사 실패 등의 사례에서도 볼 수 있듯이, 아무리 사전점검을 충분히 하여도, 고장의 발생은 개발 및 운영과정에서 피할 수 없는 요인이다. 서비스로봇은 모터와 기계를 사용하고 인간의 생활공간에서 동작하므로, 일반 가전제품보다 훨씬 높은 신뢰성과 안전성을 요구한다[6]. 따라서 고장을 감지하여 극복하거나 최소한 사람과 환경 그리고 로봇자체의 안전을 보장하기 위한 고장감내(fault-tolerance) 기술은 서비스로봇의 개발에 있어서도 필수적인 요소이다.

일반적으로 fault는 특정한 기능이 동작을 하지 않는 경우를 의미하므로, fault-tolerance는 응용계층의 컴포넌트를 개발하는 사람이 고려하여야 한다는 생각이 통상적이다. 그러나, 저자들의 그 동안 실시간 임베디드 시스템에 대한 연구 개발의 경험과 OPRoS를 위한 fault-tolerance 개발을 위한 기반 조사작업을 통하여 수많은 fault-tolerance 관련 연구 결과들을 분석한 결과, 대부분의 고장감내기술은 일정한 패턴을 가지고 있다고 판단하였다(표 1 참고). 따라서 OPRoS 기반의 로봇시스템 구축에 있어서, 컴포넌트 개발자가 fault-tolerance 기능을 포함하는 것 보다, 프레임워크에서 일괄적으로 제공하고 응용 개발자나 통합 개발자가 선별할 수 있도록 하면 많은 장점이 있다고 판단된다.

표 1. 로봇과 관련된 전형적인 고장원인과 처리 방법.
Table 1. The robot and computing system's typical faults causes and handling.

고장원인	고장검출	고장 처리
SW/메모리 접근에러	OS 예외검출	Restart
SW/논리에러	시그널모델검출 다수결(Voting)	Restart/이중화 다수결
SW/데드락	Watchdog	Restart
SW/설계에러	프로세스모델검출	이중화 대치
HW/파손	시그널모델검출	이중화 대치
HW/오동작	프로세스모델검출 다수결	이중화 대치 다수결
HW/연결손상	신호검출	고장정지
외부/동작범위 외사용	센서에 의한 환경측정	고장경고&정지
외부/통신두절	Timeout	재시도/이중화대치
외부/걸림 등	프로세스 모델	(대응절차사전설계)

본 논문에서는 이러한 프레임워크기반의 fault-tolerance 지원방법을 OPRoS 시스템을 기반으로 개발함으로써, 이와 같은 주장을 입증해보고자 한다. 서론에 이어 제 II 장에서는 우선 OPRoS 컴포넌트 모델에 대하여 약술하며, 제 III 장에서는 제안하는 fault tolerant OPRoS 프레임워크 구조를 소개하

* 책임저자(Corresponding Author)
논문접수: 2010. 1. 4., 수정: 2010. 2. 18., 채택확정: 2010. 3. 18.
안희준, 이동수: 서울산업대학교 제어계측공학과
(heejune@snut.ac.kr/acemania83@hanmail.net)
안상철: 한국과학기술연구원 영상미디어센터(asc@imrc.kist.re.kr)
※ 본 연구는 지식경제부 전략기술개발사업 지능형 로봇 개발을 위한 공통기반 플랫폼 기술개발 (OPRoS) No-10030826의 지원을 받았음.

고 프레임웍 기반 지원의 장점을 분석한다. 제 IV 장에서는 지금까지 포함시킨 구체적인 fault tolerance 방식들에 대하여 상술한다. 제 V 장에서는 몇 가지 대표적인 시나리오를 통하여 설계된 방식을 검증한다. 마지막 VI 장에서는 본 연구의 의미와 현재 진행중인 확장내용을 설명한다.

II. OPRoS 컴포넌트 모델의 소개

본 연구의 정확한 이해를 위해서는 OPRoS 플레임워크의 클래스 API 수준의 상세한 이해가 필요하겠지만, 논문의 공간 관계상 본 절에서는 꼭 필요하다고 생각되는 요소위주로 간략히 설명을 하려고 한다. 자세한 내용은 규격[2]이나, 참고구현[3], 그리고 OPRoS의 초안의 설명[7]을 참고하는 것이 바람직하다. 그림 1은 OPRoS 시스템의 전체적인 구조를 간략히 도식화 하고 있다. OPRoS 시스템은 보통 컨테이너라고 불리는 런타임 프레임워크과 컴포넌트들로 구성되어 있다. 컴포넌트는 각각이 특정적인 센서, 구동기, 알고리듬 등의 응용서비스를 제공하며, 이 컴포넌트들간의 정보교환은 포트를 통해서만이 가능하다. OPRoS 규격은 호출의 동기화 또는 비동기화 여부 및 전달데이터의 특성에 따라 데이터, 이벤트, 서비스 포트의 3가지 포트를 제공하고 있다. 각 컴포넌트는 'Component'라고 하는 부모 클래스를 상속받아 구현하도록 정의하고 있으며, 이를 사용자 정의 컴포넌트라고 부른다. 컴포넌트 개발자는 이 사용자 정의 컴포넌트와 다수의 보조 클래스를 통하여 서비스를 제공하며, 해당 컴포넌트의 구성 정보는 '컴포넌트 프로파일'이라고 부르는 XML 파일에 정의하도록 하고 있다. 보통은 몇 개의 컴포넌트들이 합쳐져서 하나의 하나의 태스크, 또는 응용서비스를 구성하게 된다. 이러한 구성내용과 설정내용들은 '응용 프로파일'이라고 부르는 XML 파일을 통하여 정의되도록 하고 있다.

프레임워크와 컴포넌트간의 인터페이스는 기저 Component 클래스의 인터페이스, lifecycle, port, property 인터페이스를 통하여 이루어 진다. 프레임워크는 이 인터페이스들을 통하여 컴포넌트의 생성과 소멸, 컨피규레이션(포트 연결, 컴포넌트 변수 설정, 포트 연결), 스케줄링(스레드관리), 통신서비스 등을 지원하게 된다. 예를 들어 컴포넌트의 동적 로딩과 언로딩을 포함한 객체의 생성, 소멸을 관리한다. 또한 프레임워크 내의 실행기는 컴포넌트에 정의된 콜백함수(callback function)들 initialize(), start(), stop(), destroy(), recover(), update(), reset() 등을 호출하여 생명주기를 제어하며, 컴포넌트의 서비스를 수행하기 위하여 onExecute()와 onEvent() 콜백함수를 호출한다. 이 함수의 호출시에 컴포넌트는 입력데이터의 수신, 알고리듬 수행, 데이터 출력 등을 수행한다. 그 외에 주요사항으로 원격 머신에 존재하는 컴포넌트들 간에 통신을 위한 어댑터와 커넥터 등이 정의되어 있으나, 본 논문의 주제와는 직결되지 않으므로 자세한 설명은 생략한다.

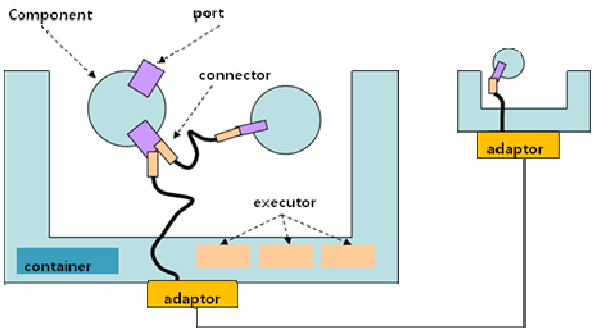


그림 1. OPROS 프레임워크의 구조 와 컴포넌트 모델.
Fig. 1. The OPRoS platform and component model[2].

III. Fault Tolerance 지원 구조

서론에서 주장하였듯이 본 연구에서는 그림 2에서 도시하는 바와 같이 fault tolerance 기능의 역할을 응용 컴포넌트에서 프레임워크의 fault manager로 이동시킨다.

컴포넌트 자체에는 특별한 fault-tolerance 기능을 갖지 않으며, 대신 fault manager가 여러 가지 로봇의 fault-tolerance 지원에 필요한 기능들을, 즉, fault detection, fault isolation, fault recovery를 지원한다. 컴포넌트들을 조합하여 복합컴포넌트(composite component)를 구성하거나, 컴포넌트를 사용하여 응용 프로파일의 구성 시에 컴포넌트 개발자나 통합개발자는 자신이 필요한 fault-tolerance 기능을 선언하여, 검출알고리즘에 필요한 변수, 고장의 위험 정도, 파급범위, 대응방안 등을 기술하도록 한다. 이러한 내용은 기존의 OPRoS 표준의 프로파일 XML을 확장하여 정의하였으며, 그 구체적인 사항은 다음 장에서 설명한다.

그 결과 제공되는 프레임워크를 사용한 개발 과정과 로봇 시스템은 다음과 같은 장점을 갖게 된다. 첫째, 디자인 공학적인 측면에서 다음과 같은 역할 분담이 가능해진다. 모든 컴포넌트 개발자는 실제적으로 fault-tolerance에 전문가가 아니며, 그럴 필요도 없다. 컴포넌트 개발자는 자신의 응용 서비스에만 집중하여 개발하면 된다. 기존에 컴포넌트 개발자가 fault-tolerance를 위하여 각자 중복적으로 개발하던 기능은 프레임워크에서 한번만 구현하면 된다. 두번째, 일관성 있는 신뢰성 보장이 가능해진다. 제각기 다른 숙련도와 습관을 가지고 있는 개발자들에 의하여 개발된 컴포넌트들은 신뢰성 정도에 크게 차이가 날 수 밖에 없다. 프레임워크에서 전문가에 의하여 잘 제공된 fault-tolerance기능은 이러한 문제를 해결하여 준다. 세번째, 컴포넌트 범위를 넘어선 문제를 해결해준다. 컴포넌트를 새로운 구성과 새로운 환경에서 응용되는 경우에도 적용이 가능하다. 반면, 컴포넌트 개발자가 모든 상황을 예측하여 고장을 정의하는 것은 불가능하다. 하나의 컴포넌트를 벗어나거나 하나의 응용 범위를 벗어나는 고장

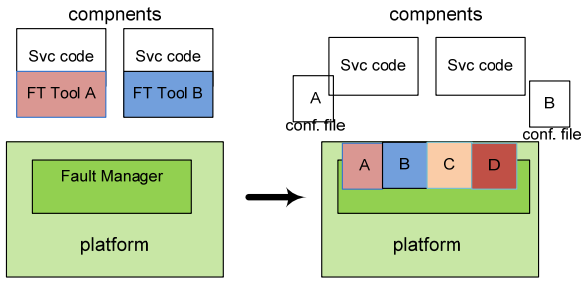


그림 2. 응용기반과 프레임워크기반 fault-tolerance지원 기법의 시스템 구조 비교.
Fig. 2. Comparison between application-based and framework-based fault tolerance supports.

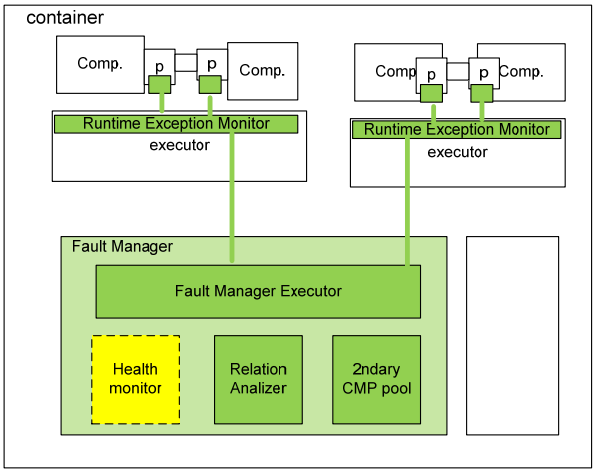


그림 3. 제안된 Fault tolerance지원 OPROS 프레임워크 구조.
Fig. 3. The proposed fault tolerance OPRoS framework architecture.

문제 (주로 고장의 파급에 관한)의 해결이 가능해진다.

이를 본 연구에서는 OPRoS 규격에 바탕을 두고 그림 3과 같이 fault-tolerance지원 구조를 설계하였다. 설계의 가장 큰 특징은 OPRoS 시스템의 계층적인 구조를 fault-tolerance 기능의 제공에 있어서도 그대로 반영한다는 점이다. 일반적으로 OPRoS와 같이 계층화된 시스템에서 오류가 여러 단계를 지나서 감지되게 되면, 문제의 원인 파악이 어려워지게 된다 [8,9]. 따라서 고장이 발생되는 컴포넌트의 가장 가까운 위치인 포트와 실행기에 고장의 감지에 필요한 기능을 배치하였다. 반면 고장분석과 처리기능을 수행하는 fault manager는 프레임워크의 중심에 배치하여 여러 컴포넌트 및 응용을 제어할 수 있도록 하였다. 여기에는 컴포넌트의 고장원인과 파급정도를 분석을 하는 분석모듈과, 상태를 모니터링 하기 위한 모니터, 고장 컴포넌트를 대치할 컴포넌트를 보관하는 컴포넌트 풀 관리 모듈이 포함되어 있다.

IV. 프레임웍에 적용된 fault tolerance 기법들

우선, 본 연구의 주 목적은 새로운 fault-tolerance기법자체의 개발에 있지 않고, OPRoS와 같이 컴포넌트 기반의 프레임워크를 사용하는 로봇시스템에서 신뢰성을 보장하기 위한 것에 목적을 두고 있다. 따라서 여기서 다루는 고장감내 방법은 기존에 다양한 연구에서 활용된 방법들에 기반하고 있으며, OPRoS 프레임워크에 적용하면서 고려된 사항들 위주로 기술한다.

고장처리는 일반적으로 고장 감지, 고장분석, 고장처리의 3단계로 이루어지며, 다음은 이 순서대로 기술한다. 사실 정확한 번역은 fault는 오류, error는 에러, failure는 고장으로 번역이 된다[1,8]. 각 처리 방식을 지원하기 위한 프로파일 XML 요소들은 표 2에 일괄적으로 정리하였다. 원칙적으로 OPRoS는 XML 스키마를 사용하지만, 현재 참고 구현에서는 유효성검증(validation)은 하지 않고 있으며, 본 논문에서는 편의상 DTD (Document Type Definition)를 사용하여 정의하였다.

1. 고장 감지

고장감지는 응용에 따라 모두 다르기 때문에 가장 어렵고 응용에 의존적인 기술이다. 그러나 사실상 모든 고장은 다음의 방법으로 감지가 가능하다.

표 2. 프로파일 파일의 확장 XML 요소.
Table 2. The extended fault tolerance elements for profile XML files.

기능	XML 확장요소	관련 프로파일
Fault diagnosis	dependency:= (name, reference, severity) name = (#PCDATA) reference := (URL) severity:= (ignore| stop | destroy)	System
Fault recovery	Fault:= (retry_max, secondary+, severity) retry_max = (#PCDATA) reference := (URL) severity:= (ignore| local | global)	Application
	Resurrect := (Reset, Backup) Backup:= <backup>checkTime </backup> CheckTime = (#PCDATA)	Component
Fault detection	validation:= (rang_min, range_max) rang_min = (#PCDATA) range_max:= (#PCDATA)	Component

1.1 실시간 예외 검출

실시간 오류검출은 메모리 접근오류, 연산오류(divided by zero)와 같이 계산환경상의 오류들을 의미한다. 서비스로봇의 경우 소프트웨어 컴포넌트가 주를 이를 것으로 예상이 되기 때문에 그 중요도는 일반적인 실시간 제어시스템에 비하여 증대되고 있다. MMU (Memory Management Unit)를 사용하는 시스템 환경에서는 잘못된 메모리 사용을 어느 범위 안에서 검출할 수 있으며, 프레임워크가 컴포넌트의 콜백함수들을 호출할 때 발생하므로, 함수 수출 이전에 예방조치를 함으로써 예외를 검출할 수 있다. 시스템에 따라 UNIX계열에서는 시그널 처리 ('sigsetjmp'와 longjmp')[10]를 사용하여 처리할 수 있으며, win32 환경에서는 structured exception ('__try __except')[11]을 활용하면 처리가 가능하다.

현재 OPRoS 프레임워크는 쓰레드 모델을 활용하여 컴포넌트간 메모리 공간을 공유하므로, 한 컴포넌트에서의 오류가 다른 컴포넌트로 파급되는 것이 가능하다. 사실상 메모리 접근 에러는 C언어나 C++과 같이 배열이나 포인터를 사용하는 경우에 가장 고질적인 오류로 완벽하게 제거하거나 예방한다는 것은 불가능하다. 그러나 [12]등의 연구결과를 보면 대부분의 메모리 에러의 시작은 포인터 초기화 오류이거나 배열 인덱스 오버플로우에서 시작되는 것이 대부분이다. 이를 위하여 본 연구에서 컴포넌트간의 메모리간에 접근 방어벽을 구현하였다. 이 방식은 Electric Fence[13,14]와 같은 디버깅 프로그램이나 바이러스 예방 프로그램에서 사용하는 방식이나, 할당단위로 방어벽을 설정하게 되면 부하가 심하여 실시간 동작에 영향을 주게 되므로 컴포넌트 단위의 방어벽을 설정하는 방법을 사용한다.

1.2 시그널 모델 기반 오류 검출

로봇시스템은 기계적인 부분을 포함하며 움직이는 전자장치이므로 생산 당시에는 온전하지만, 시간이 지나면 하드웨어가 고장이 나던가 회로의 연결이 끊어지는 등의 오류가 발생하게 된다. 컴포넌트 모델기법은 일반적으로 컴포넌트를

블랙박스로 취급한다. 따라서 컴포넌트 자체의 논리적인 오류를 발견하는 것이 쉽지 않다. 하지만 해당 컴포넌트의 출력 값은 일반적으로 포트가 갖는 문법적인 범위보다 훨씬 제한적인 것이 일반적이다. 따라서 이러한 입출력 값의 의미 있는 범위를 검사하면 고장 컴포넌트의 발견이 용이해진다. 본 시스템에서는 이러한 입출력 검증을 개발자가 선언한 XML 요소에 바탕으로 포트에서 직접 수행하게 된다. 대표적인 예는 거리 센서에서 들어오는 측정 값인 경우(Dmin, Dmax)를 사용하는 것이다. 이때 Dmin을 0이 아닌 0보다 약간 큰 값으로 사용하게 되면, 센서컴포넌트의 논리적인 고장뿐 아니라, 하드웨어적인 작동 중지 및 연결회로의 고장으로 0값이 나와도 확인할 수 있게 된다.

이를 구현하기 위하여 프레임워크는 컴포넌트의 포트를 연결시 해당 응용프로파일의 설정을 기초로 입출력 값의 허용범위를 세팅하도록 하였으며, 컴포넌트가 입출력시 범위를 검사하고 위반시 사용자 정의 예외를 전달하도록 하였다.

1.3 프로세스 모델 기반 오류 검출

프로세스 기반 모델은 다수의 컴포넌트들이 연결되어 있거나 플랜트가 끼어있는 상황에서 일종의 시스템이 내부적으로 가지고 있는 모델과 실제동작 결과의 측정한 값과의 차이 비교하여 고장 여부를 판단하는 방법이다. 보통 이 방법은 사용되는 컴포넌트가 복잡하던가 변수를 정확히 정의할 수 없을 경우 또는 불확정적인 외부환경의 요인이 개입되는 경우에 사용되는 고급 오류 검출 기법이다[15]. 그러나 서비스로봇의 환경과 구조는 현재 까지는 매우 간단하게 모델링이 가능하므로, 상당히 손쉽게 이를 적용할 수 있다고 판단된다. 예를 들어 네비게이션 컴포넌트에서 로봇의 위치구동 모터에 90도 회전을 명령하고, 이를 자이로센서 등을 통하여 감지한 결과와 비교하는 것이다. 이때 모터의 이상이나 외부의 인위적인 힘이나 바퀴의 고장이 있는 경우 90도와 큰 차이가 나게 될 것이며 이를 비교하여 고장을 발견할 수 있다. 이때 시스템의 모델을 제공하는 것과 시뮬레이션 한 결과와 실측 결과의 차를 비교하여 결정을 내리는 과정을 정의해야 한다.

본 OPRoS 환경에서는 모델의 제공은 시스템 통합자가 라이브러리 형태로 하게 정의하였으며, 결정은 XML에 일정한 오차 범위로 설정하도록 하였다. 좀더 간편한 프로세스 모델을 정형화하기 위한 노력은 현재 계속 진행중인 부분이다.

2. 고장 분석과 고장 고립

오류가 감지되면, 오류 분석은 fault manager에 의하여 수행이 된다. 오류 분석은 오류 원인과 오류 영향분석 그리고 오류 처리를 위한 결정으로 구성된다. 본 연구에서 적용된 방법은 컴포넌트 시스템의 고장 파급에 대한 분석 및 대책의 연구인[16]에 기초하여 OPRoS의 에 맞도록 수정 확장하였다.

우선, 오류 원인 분석은 앞서 기술된 오류검출에서 상당히 결정이 되며. 프로세스 모델인 경우를 제외하면, 오류 감시를 컴포넌트 단위로 하고 있기 때문에, 컴포넌트 레벨의 위치는 직접적으로 파악된다. 또한 모든 오류코드는 저자들에 의하여 표준에 세분화되도록 정의되어 있다. 즉, 오류의 제공자가 포트의 호출 컴포넌트인지 호출받는 컴포넌트인지, 원인이 메모리 에러인지, 데이터 불일치인지, 입출력 범위에러인지 등을 구별하고 있다.

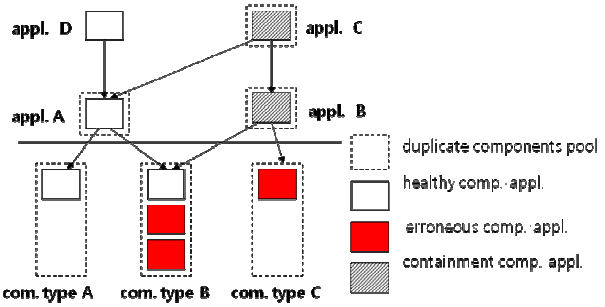


그림 4. 컴포넌트 모델에서의 오류 파급 분석 및 처리.

Fig. 4. Fault propagation and handling in component-based application models.

발생한 오류의 심각성과 응용서비스에 대한 파급정도는 앞서 설명한 XML 문서에 개발자가 기술한 정보를 바탕으로 분석한다.

그 결과는 무시 (‘ignore’), 컴포넌트 갱생 (‘reset’, ‘resurrect’), 컴포넌트 중지 (‘stop’)로 구별된다. 이 과정에서 컴포넌트와 응용관계뿐 아니라 응용간의 상관성도 고려된다. 또한 그림 4는 각 컴포넌트를 사용하는 응용과 이 응용에 의존하는 응용들간의 상관도를 표시한 것이다. 응용 A는 몇 개의 컴포넌트 B타입의 두 개의 구현이 고장에도 불구하고, 이중화로 인하여 동작이 가능한 반면, 응용 B는 컴포넌트 C 타입이 오류로 인하여 동작이 중단되었다. 또한 응용 B에 의존적인 응용 C도 동작을 정리하도록 관리하고 있다. 이에 다한 적용 예는 5절의 검증 시나리오에 소개한다.

3. 고장 처리

고장 처리는 고장난 컴포넌트를 다시 살리거나, 다른 컴포넌트로 대치하던가 하게 된다. 만약 두 가지 조치가 모두 불가능한 경우는 해당 컴포넌트의 오류가 가지는 리스크 (위험성)을 고려하여 안전하게 대피하던가 동작을 중지시키도록 정의하고 있다. 그림 5는 이 각 단계의 시나리오를 각기 설명하고 있다.

- 고장복원: 고장복원은 신속해야 한다.
- 고장동작: 고장의 영향이 인간이나 시스템의 안전과 관련이 없는 경우 해당 응용만을 정지하고 오류와 관련 없는 기능은 동작하도록 한다.
- 고장안전: 오류안전이 인간이나 시스템 자체에 영향을 주는 경우, 시스템의 상태를 안전하게 위치시킨 후 시스템을 정지하여야 한다.

내부에 유지해야 하는 상태정보가 없는 컴포넌트인 경우는 단순히 Reset을 함으로서 갱생이 가능하다. 즉, 프레임워크는 컴포넌트의 onError(), onReset(), onRecover() 를 차례로 호출하게 된다. 반면 자체적인 상태를 갖는 경우는 이 상태 정보를 주기적으로 저장하는 것이 필요하다. 본 연구에서는 이를 위하여 표준의 콜백함수를 확장하였다. 즉, 주기적인 상태 저장을 위하여 onBackup() 함수를 추가하였으며, onRecover( ) 함수는 void 인자에서 복원용 데이터 버퍼를 인자로 추가하였다.

고장 복원이 안 되는 경우 해당 컴포넌트는 대치를 하거나 대치 컴포넌트가 없는 경우는 기능을 정지해야 한다. OPRoS 시스템에서 고장 컴포넌트 대치과정은 우선 고장 컴포넌트

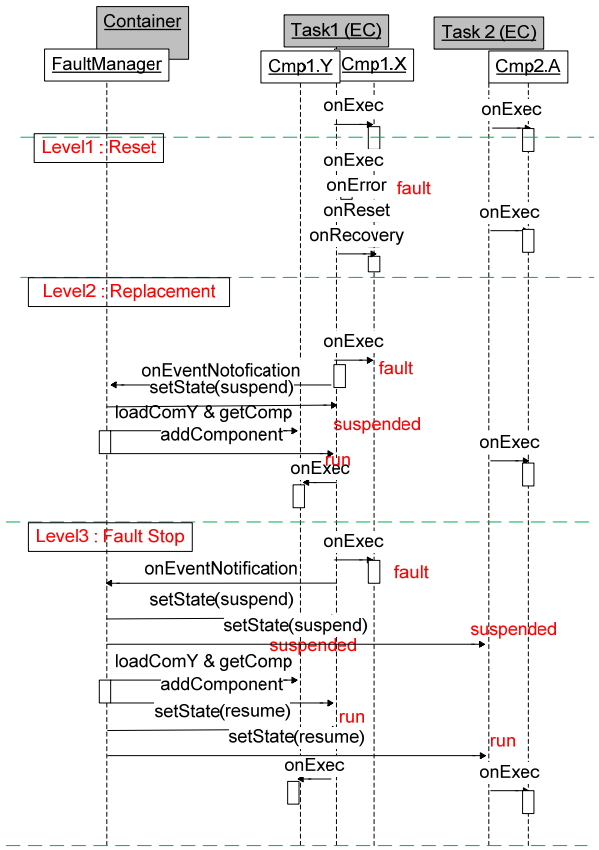


그림 5. 3가지 고장상황의 고장 처리. 메소드 호출 절차.
Fig. 5. Method call flows for 3 different fault handling cases.

에 관련된 응용 태스크를 중지시키고, 해당 컴포넌트와 연결된 포트들을 모두 제거한다. 그리고 새로운 컴포넌트를 로딩 및 생성한 후 기존의 컴포넌트들과 연결을 대치한다. 연결대치가 완료면, 해당 응용 및 관련 응용을 다시 실행시킨다.

V. 검증 시험

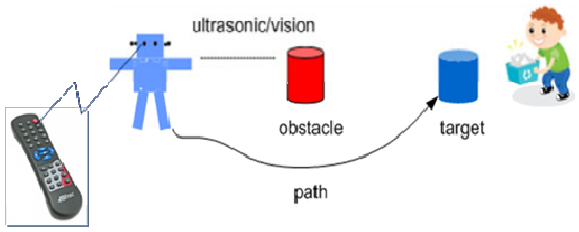
1. 기능 검증 시험

저자들은 OPRoS 규격을 참고로 OPRoS 프레임워크와 몇 가지 응용 시나리오를 개발하였다[9,10]. 실험은 마이크로 소프트 윈도우즈 XP와 데스크탑 리눅스, 그리고 임베디드 리눅스 환경에서 시험하였다. 특히 임베디드 리눅스 환경은 Robonova 바디와 임베디드 리눅스를 사용하는 HBE-Robonova-AI[17]를 기반으로 실제 로봇환경에서 구축하였다. Robonova-AI의 기본 사양은 표 3에 제시하였다. 참고로, 향후 OPRoS는 실시간 OS 환경에서도 구현하는 것을 목적으로 하고 있으나, 이를 위하여서는 컴포넌트 동적로딩등 해결해야 할 기술적인 문제가 있다.

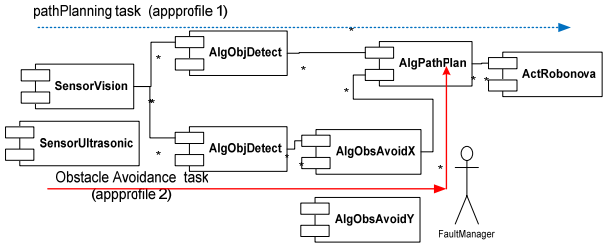
앞서 정의된 기술들을 하나씩 모두 평가하였다. 우선, 검출기능 중 예외처리 기능과 컴포넌트 장벽기능은 초기화 안된 포인터와 배열의 인덱스 범위를 넘는 오류 컴포넌트를 만들어서 검증하였으며, 신호 모델 기능은 초음파거리 측정센서의 출력 범위를 (2cm, 3m)로 제안한 후, 센서를 동작 중에 로봇에서 분리하여 신호범위를 벗어나도록 하고 실험하였다. 프로세스 모델은 경로계획 컴포넌트가 로봇이 전후 좌우를 이동하는 명령에 대하여, 실제 초음파센서의 거리측정값을

표 3. 실험에 사용한 Robonova-AI [17]의 기본 사양.
Table 3. System Specification of Robonova-AI [17].

	구성	사양
메인 보드	CPU	PXA272-520MHz,
	메모리	16MB flash, 64MB dram
	OS	linux 2.6.12 (WINCE 가능)
	카메라 입력	CCD, SAA7113H, FPGA 160x120 (upto NTSC)
로봇 바디	외형	1.3kg, 310x180x90mm
	서버모터	17개, 0.2sec/60°, 6 Volt, PWM, HMI
	CPU	8051, robo-basic



(a) The scenario illustration.



(b) Component and application task structure for robot navigation with obstacle avoidance.

(c) The snapshot of test campaign.

그림 6. 대표적인 테스트 시나리오.
Fig. 6. Test application scenario.

평균하여 원하는 정상보폭의 50% 이상이 변화하는지 기준으로 하도록 하고, 로봇의 다리에 있는 나사를 풀어지도록 조작 후 실험하여 고장을 검출하는지를 확인하였다.

오류분석과 처리기능에 대하여는 그림 6과 같이 구체적인 응용시나리오를 작성하여 실험하였다. 시험 응용 시나리오는 충돌방지(obstacle avoidance) 응용과 경로계획응용(path-planning)의 상관된 응용으로 구성하였다. 경로계획응용은 자체적으로는 파란색의 물체를 추적하여 이동하며, 추적물체 이외의 다

른 물체와의 충돌을 방지하기 위하여 충돌회피 응용에서 충돌 물체와의 위치 정보를 전달받게 된다.

충돌방지 응용은 비전센서(SensorVision) 컴포넌트와 오브젝트를 검출하는 영상처리 컴포넌트(AlgObjDetect), 그리고 이를 받아 방해물체들의 위치를 계산하고 이 결과를 경로계획 컴포넌트(AlgPathPlan)로 전달하는 컴포넌트(AlgObsAvoidX/Y)로 구성하였다. 경로구획 컴포넌트는 비전센서(SensorVision), 물체인식(AlgObjDetect), 경로계획(AlgPathPlan), 그리고 모터 구동 보드를 제어하는 구동 컴포넌트(ActRovonova)로 구성하였다.

우선, 시나리오의 정상적인 동작을 확인하였고, 두 가지 오류상황을 설정하였다. 첫 번째는 컴포넌트 교체를 실험하였다. 오류는 실험을 통제하기 위하여 IR 센서를 통하여 인위적으로 충돌물체 회피용 알고리듬(AlgObjAvoidX)에 포인터값을 변화시켜 메모리 접근에러가 발생하도록 하였다. 이 경우 로봇은 AlgObjAvoidX에서의 에러발생을 예외핸들러를 통하여 감지하였고, 해당 컴포넌트를 다른 알고리듬을 사용한 AlgObjAvoidY로 대치하고 계속 수행하여 목적을 완수하였다.

두번째 시험을 위해서는 충돌회피에 사용하는 센서컴포넌트를 비전에서 초음파센서(SensorUltrasonic)로 교체하였다. 오류는 초음파센서를 동작 중에 분리하였으며, 이때 시그널 범위에러를 검출하였다. 응용 컴포넌트 프로파일에 기술된 처리에 따라 센서가 오류가 있고 대치할 센서기능이 없는 경우는 물체탐지기능은 정지하게 된다. 또한 경로계획 응용은 물체충돌 응용이 오류상태에 있는데 동작하면 문제가 발생하므로 자체 컴포넌트들은 정상상태이지만 동작을 정지하고 사운드 출력을 하도록 하였다.

2. 성능 검증 시험

Fault-tolerance 기능의 성능상의 검증 요소는 오류검출 성능과 오류처리 지연시간을 고려하는 것이 필요하다. 그러나, 오류검출 성능을 보기 위해서는 다양한 실험이 가능해야 하는데, 현재 OPRoS의 컴포넌트들이 많이 존재하지 않기 때문에 데이터로 삼을 수 있는 수준의 다양성을 갖추고 있지 못하다. 따라서, 본 논문에서는 오류발견에서 처리까지에 걸리는 소요되는 시간지연만을 살펴보기로 한다. 표 4는 다양한 시스템 부하상태에서 오류를 주입하였을 때 오류를 발견에서 처리하는 시점까지의 시간지연을 측정한 결과를 표시하였다. 측정결과는 시스템이 60%이하의 CPU 활용률에서는 오류발견에서 처리까지 걸리는 시간은 수 msec 미만인 반면, 60%를 넘어가는 시스템 부하상태에서는 대체 컴포넌트를 사용하여 복구하는 경우에는 시간지연이 급속히 커지기 시작하여 80% 이상에서는 수백 밀리초에서 수초까지 걸리는 것을 확인할 수 있다.

표 4. 오류 처리 시간 (괄호안은 사전 로딩 사용시).
Table 4. Fault Recovery Time (numbers in parenthesis are latencies after our component pre-loading techniques are applied).

Load level	WinXP/P4	Linux/P4	Linux/arm (flash ROM)
~10%	8 ~ 20 ms (<1m)	1 ~ 3 ms (<1ms)	10 ~ 50 ms (<1m)
~40%	10 ~ 40ms (<1m)	2~ 12ms (<1ms)	10 ~ 120 ms (<1m)
~60%	20 ~ 100ms (<1m)	10~30 ms (<1ms)	100 ~ 350 ms (<3m)
~80%	> 200ms (<1m)	> 100 ms (<1ms)	> 500 ms (<5m)

이런 현상의 원인은 예상할 수 있듯이 과부하에서 급격히 응답시간이 저하되는 원인은 컴포넌트 로딩시간인 것으로 확인되었다. 또한, 리눅스 시스템에 비하여 윈도우즈에서 로딩 시간이 비교적 큰 것은 현재 CDL (Component Development Library)과 동적 라이브리 사용방식이 윈도우즈와 리눅스에서 차이가 있기 때문이다. 즉, OPRoS가 윈도우환경에서는 역참조기능(back-reference) 기능을 사용하고 있지 않아, 사용자 컴포넌트 라이브러리에 CDL를 정적으로 링크하고 있어서 전체적인 DLL 라이브러리 크기가 커진 반면, 리눅스에서는 역참조 기능인 '-Wl,export-dynamic' 링크옵션을 사용하여 사용자컴포넌트에 CDL이 포함되지 않아 컴포넌트의 크기가 작기 때문이다. 임베디드 시스템에서는 flash 롬을 저장장치로 사용하므로, 로딩시간이 일반적으로 크기 때문에 (과부하에서 초 단위까지 지연 됨) 특별한 주의가 필요하다.

이러한 컴포넌트 로딩지연의 문제를 완화하기 위하여, 부하를 제한하는 방법과 대체컴포넌트를 사전에 로딩해놓는 방법을 고려하였다. 현재 상당수 서비스로봇이 컴퓨터 비전 기술을 사용하고, 비전 처리는 상당한 시스템 부하를 요구하기 때문에 80%부하의 경우가 비현실적인 것은 아니다. 따라서 본 연구에서는 대체기능을 사전에 로딩하는 방법을 사용하였다. 그 결과 오류 검출에서 오류 처리까지 걸리는 지연의 시간은 부하 정도에 상관없이 5 msec 이내로 제한하는 것에 성공하였다.

VI. 결론

본 논문은 컴포넌트 기반 프레임워크를 사용하는 서비스로봇이 상용화를 위하여 필수적으로 요구되는 신뢰성을 확보하기 방안을 제안하였다. 제안된 방식은 기존에 개발에서 사용하던 응용 컴포넌트 레벨의 지원을 방식을 지양하고, 프레임워크에서 고장감내기능을 제공하며, 이를 컴포넌트 개발자나 통합개발자가 선언적으로 선택하여 사용하는 방식이다.

제 3절에서 프레임워크 지원 fault-tolerance 방식의 개발상의 기능상의 장점에 대하여 상세히 논거 하였으며, 이 구조하에서 대표적인 고장감내기법의 구현하고 시나리오 별 검증을 통하여, 제안하는 방식 들을 실제적으로 구현 가능함을 있음을 보였다. 또한, 실시간 성능분석을 통하여 문제점을 파악하고, 예비 컴포넌트 사용 방안도 제안하였다.

구체적인 fault-tolerance 기법에 있어서, 컴포넌트간의 접근 오류를 해결하는 방법은 좀더 개선이 필요한 부분이다. 또한 프로세서 모델 기반 방식은 그 범위가 매우 다양하여 매우 기초적인 수준으로 적용된 한계점이 존재한다. 더욱이 현재 많은 노력에도 불구하고, 사용자 입장에서 쓸만한 응용 컴포넌트가 많지 않을 뿐 아니라, 쉽게 확보하기도 어려운 상황이다. 따라서, 본 연구의 결과의 응용 시험도 상당히 제한적이다. 앞으로 새로운 응용서비스에 적용된 결과들은 허용하는 한 공식 OPRoS 홈페이지 등에 공개를 할 계획이다.

끝으로, 현재 본 논문에서 제안된 프레임워크 기반 구조가

OPRoS의 QoS와 보안에도 사용할 수 있다고 믿고, 이에 대한 설계를 진행 중이며, 본 연구의 결과가 다양한 컴포넌트와 하드웨어 플랫폼에 적용될 수 있도록 노력 중이다.

참고문헌

[1] KS B 6939, "서비스로봇-용어-제1부: 분류 및 일반용어," 한국표준협회, 2006.

[2] Korean Intelligent Robot Standard Forum, "OPRoS Component Spec," 2009.

[3] OPRoS project official site, http://www.opros.or.kr/

[4] OMG, Robotic Technology Component Specification Version 1.0, April 2008

[5] Microsoft, Microsoft Robotics Developers Studio R2, http://msdn.microsoft.com/en-us/robotics

[6] ISO/TC184/SC2/WG8, Service Robot Group.

[7] B. Song, S. Jung, C. Jang, and S. Kim "An Introduction to robot Component Model for OPROS," *Proc. of SIMPAR 2008*, Italy, Nov. 2008

[8] I. Koren and C. M. Krishna, *Fault Tolerant System*, Morgen Kaufman Publisher, San Francisco, CA, 2007.

[9] C. Ferrell, "Failure Recognition and Fault Tolerance of an Autonomous Robot," *Adaptive Behavior,* vol. 2, pp. 375-398, 1994.

[10] K. A. Robbins, *UNIX System Programming*, Prentice Hall, 2004.

[11] J. M. Hart, *Win32 system Programming: Chapter 5 Structured Exception Handling*, Addison-Wesley Professional, 1997

[12] Inhwan Lee, and R. K. Iyer, "Software Dependability in the Tandem GUARDIAN System," *IEEE Trans. on Software Engineering*, vol. 21 , no. 5, pp. 455-467, May 1995.

[13] G. R. Lueke, J. Coyle, J. Joekstra, M. Kraeva, Y. Li, O. Taborslaia, and Y. Wang, "A Survey of Systems for Detecting Serial Run-Time Errors," *Concurrency and Computation: Practice and Experience,* vol. 18, no.15, pp. 1885-1907, 2006.

[14] B. Perens, Electrical Fence, http://perens.com/Freesoftware/ElectricFence

[15] R. Isermann, "Supervision, fault-detection and fault-diagnosis methods — An introduction," *Control Engineering Practice*, vol. 5, no. 5, pp. 639-652, May 1997.

[16] H. Ahn, H.-J. Oh, and J. Hong, "Towards Reliable OSGi Operating Framework and Applications," *Journal of Information Science and Engineering*, vol. 23, no. 5, pp.1379-1390, Sep. 2007.

[17] Hanback Electronics Inc, HBE-Robonova-AI (Embedded Robot with Robonova Body), http://www.hanback.co.kr/products/

안 희 준

1993년 KAIST 전기및전자공학과(공학사). 1995년 KAIST 전기및전자공학과(공학석사). 2000년 KAIST 전기및전자공학과(공학박사). 1999년~2000년 독일 뉴른버그대학 박사후연구원. 1997년~2002년 LG 전자-정보통신 선임연구원. 2002년~2003년 Tmax Soft 연구소 책임연구원. 2004년~현재 서울산업대학교 제어계측공학과(부교수). 관심분야는 소프트웨어 최적화, 실시간 시스템, 영상 통신, 인터넷 프로토콜.

이 동 수

2009년 서울산업대학교 제어계측공학과(공학사). 2009년~현재 서울산업대학교 산업대학원 제어계측공학과 재학중. 관심분야는 임베디드 소프트웨어, 실시간 시스템, 로봇공학.

안 상 철

1988년 서울대학교 제어계측공학과(공학사). 1990년 서울대학교 제어계측공학과(공학석사). 1996년 서울대학교 제어계측공학과(공학박사). 1996년~1997년 미국 USC 방문연구원. 1997년~현재 KIST 영상미디어센터(책임연구원). 관심분야는 로봇공학, 비젼 기반 HCI, 가상현실.