로봇 소프트웨어 컴포넌트의 실행 모니터링/ 효율적인 데이터 관리방안

Health Monitoring and Efficient Data Management Method for the Robot Software Components

김 종 영, 윤 희 병*

(JongYoung Kim¹ and HeeByung Yoon¹)

¹Korea National Defense University

Abstract: As robotics systems are becoming more complex there is the need to promote component based robot development, where systems can be constructed as the composition and integration of reusable building block. One of the most important challenges facing component based robot development is safeguarding against software component failures and malfunctions. The health monitoring of the robot software is most fundamental factors not only to manage system at runtime but also to analysis information of software component in design phase of the robot application. And also as a lot of monitoring events are occurred during the execution of the robot software components, a simple data treatment and efficient memory management method is required. In this paper, we propose an efficient events monitoring and data management method by modeling robot software component and monitoring factors based on robot software framework. The monitoring factors, such as component execution runtime exception, Input/Output data, execution time, checkpoint-rollback are deduced and the detail monitoring events are defined. Furthermore, we define event record and monitor record pool suitable for robot software components and propose a efficient data management method. To verify the effectiveness and usefulness of the proposed approach, a monitoring module and user interface has been implemented using OPRoS robot software framework. The proposed monitoring module can be used as monitoring tool to analysis the software components in robot design phase and plugged into self-healing system to monitor the system health status at runtime in robot systems.

Keywords: robot software component, health monitoring technique, component based robot development, self-healing

I. 서론

기존 정보 시스템분야 뿐만 아니라 항공, 자동차, 그리고 로봇 분야에서도 기 개발되고 검증된 소프트웨어의 재사용과 컴포넌트 기반 개발방법론의 적용을 통해 개발비용을 줄이고 시간을 단축하려는 추세에 있다[1,2]. 로봇 분야의 경우, 기능 단위(예, 물체 회피, 경로 계획 등)로 재사용 가능한 구성요소(reusable building block)를 식별하고 이를 독립적으로 개발, 조립 및 실행시키는 컴포넌트 기반 로봇 개발이 활발하게 진행되고 있다[3-6].

로봇 소프트웨어의 재사용성을 증진시키기 위해 모듈화와 상호운용성은 중요한 요소이다. 컴포넌트를 기반으로 로봇을 개발하고 있는 과제의 경우 자체적인 로봇 소프트웨어 컴포넌트 모델을 정의하고 있다[3,7]. 컴포넌트 개발자는 컴포넌트 모델을 기반으로 로봇 기능단위로 컴포넌트를 개발한다. 로봇 응용 개발자는 기 개발되고 검증된 컴포넌트(COTS: Commercial Off-The-Shelf)를 응용에 맞게 조합하여 로봇을 개발하고 배포된 신규 버전의 컴포넌트를 교체하여 로봇 소프트웨어의 유지보수를 손쉽게 수행할 수 있다. 기

능별 소프트웨어 컴포넌트를 조합 시 컴포넌트에 대한 실행정보가 필요하다. 이러한 경우, 컴포넌트 개발자의 문서에 의존해야 하거나 별도의 테스트를 거쳐 분석해야 한다.

시간 제약을 갖는 실시간 로봇 응용의 경우, 설계단계에서 컴포넌트의 실행시간에 대한 분석을 통해 컴포넌트 실행시간, 실행주기 등을 결정해야 한다. 또한 로봇의 실행 중에 소프트웨어 컴포넌트의 결점 발생 여부를 탐지하기 위해 컴포넌트 수준에서 정확한 모니터링이 필요하다. 하지만 컴포넌트 기반 로봇 개발과제들을 살펴보면 컴포넌트 모델, 컴포넌트를 실행시키기 위한 프레임워크, 조립 도구 등에 초점을 두고 있다. 그리고 컴포넌트 실행정보에 대한 기준을 정성적인 수준에서 제시하고 있다. 예를 들어, 지식경제부 주관으로 개발하고 있는 오프로스(OPRoS: Open Platform for Robotic Services) 과제의 경우 로봇 소프트웨어 컴포넌트의 실행주기를 결정할 때 주기적으로 실행되는 태스크(프로세서 또는 쓰레드)는 전체 실행주기의 67% 이하로 동작하는 것을 권장하고 있다[8]. 또한 사용자 코드의 실행 시간을 분석하여 실행주기를 결정하도록 하고 있다. 따라서 로봇 실행환경에 따른 컴포넌트의 실행정보를 정량적으로 분석하기 위한 모니터링 요소 도출과 모듈의 개발이 필요하다.

본 논문에서는 포트 구조를 갖는 로봇 소프트웨어 컴포넌트에 대해 미들웨어 차원에서 효율적으로 컴포넌트의 실행

* 책임저자(Corresponding Author)

논문접수: 2011. 8. 20., 수정: 2011. 9. 5., 채택확정: 2011. 9. 25.

김종영, 윤희병: 국방대학교 국방정보체계

(kgy10g@naver.com/hbyoon37@gmail.com)

행 이벤트를 모니터링하고 관리하는 기법을 제안하고 오프로스 로봇 컴포넌트 모델을 기반으로 구현한다. 로봇 컴포넌트의 실행예외(runtime exception) [9], 데이터 값(data value) [10], 체크포인트-롤백(checkpoint-rollback), 그리고 실행 시간(time) [11] 관련 모니터링 요소를 도출하고 로봇 소프트웨어 컴포넌트가 실행될 때 수집되는 이벤트를 기반으로 효율적으로 관리하기 위한 테이블들을 설계한다.

제안 컴포넌트 모니터링 기법은 로봇 개발 및 운영단계에서 다양하게 활용될 수 있다. 우선, 로봇 응용 개발자는 분석·설계단계에서 로봇 소프트웨어 컴포넌트에 대한 WCET 분석[12]을 통해 하드웨어 플랫폼에 적합한 컴포넌트 실행정보를 정량적으로 분석하여 최적의 값을 설정할 수 있다. 또한 로봇 응용의 실행 중에 컴포넌트 상태를 지속적으로 모니터링하여 관련 정보를 관리함으로써 로봇의 자가치유[13] 시스템에 활용될 수 있다.

본 논문은 총 VI 장으로 구성되어 있다. II 장에서는 컴포넌트 모니터링과 관련된 연구를 살펴보고 III 장에서는 제안하는 로봇 소프트웨어 컴포넌트 모니터링 방법에 대하여 설명한다. IV 장에서는 모니터링 모듈의 설계 및 구현에 대해 기술한다. V 장에서는 제안 방법에 대한 실험 환경 및 결과를 기술하며, VI 장에서는 결론을 맺고 향후 연구과제에 대해 검토한다.

II. 관련 연구

전통적으로 결점을 탐지하기 위한 모니터링 기법에는 핑/에코(ping/echo), 와치독(watchdog), 하트비트(heartbeat), 그리고 예외탐지 기법이 있다[14]. 핑/에코 기법은 노드들 간의 요청과 응답의 쌍으로 구성된 비동기적인 메시지로써 네트워크 경로 상 도달과 지연 등의 상태를 확인한다. 와치독은 소프트웨어에 의해 주기적으로 리셋(reset)되는 하드웨어 기반의 타이머로써 주로 메시지 교환으로 인한 부하를 줄이기 위해 사용된다. 하트비트는 시스템 모니터와 프로세스 사이의 주기적인 메시지 교환을 통해 시스템이 올바르게 작동하고 있는지를 확인한다. 예외탐지 기법은 시스템이 결점을 탐지 시 예외를 발생하는 기법으로 윈도우는 구조적 예외처리(SEH: Structured Exception Handling)를 통해 주소 위반, 산술 실패, 시스템 오류와 같은 대처할 수 있는 매커니즘을 제공하고, 리눅스의 경우 프로세스의 정해진 시간을 초과하는 SIGXCPU 함수 등을 제공한다. 클라이언트-서버 아키텍처에서 시스템의 작동 여부를 확인하기 위해 사용되고 메시지 트래픽 등 부하가 발생하는 단점이 있다.

Sundmark et al [15]은 소프트웨어 컴포넌트의 실행시간, 메모리 사용, 컴포넌트 이벤트, 무결성 검사(sanity check) 등의 요소를 모니터링하여 컴포넌트 검증, 시스템 수준의 디버깅/테스팅, 컴포넌트 실행, 컴포넌트 관측의 컴포넌트 기반 개발방법론의 생명주기 관점에서 활용하는 방안을 제안하고 있다. 하지만 개념적인 수준에서 모니터링 방법을 제안하고 있으며 구체적인 기법은 제시하고 있지 않다.

Lüders et al [16]은 타임 스탬프 로그 메시지(timestamped log message)를 기반으로 소프트웨어 컴포넌트의 최대, 최소, 그리고 평균 실행시간을 모니터링하는 기법을

제안하고 있다. 컴포넌트의 실행시간은 프락시(proxy) 컴포넌트를 통해 호출의 시작과 종료에 대한 로그 전송을 통해 컴포넌트의 실행시간 측정하고 모니터링 한다. 컴포넌트 호출에 대한 허용가능시간(deadline) 충족 여부를 계산하기 위해 프락시 컴포넌트를 사용하는 경우 모니터링을 위한 별도의 컴포넌트가 필요하고 응용의 구성이 복잡해질 수 있다.

Garlan et al [17]는 실행중인 시스템의 행위를 모니터링하기 위해 프로브(probe)와 게이지(gauge) 개념을 제안한다. 프로브는 물리적인 환경 또는 목표 시스템에 탑재되고 실제 시스템의 실행정보를 프로브 버스(probe bus)를 통해 게이지에 전달한다. 게이지는 프로브로부터 전달된 정보를 취합 및 해석하여 상위수준으로 전달하는 역할을 수행한다. 실시간, 임베디드 환경 보다는 분산 컴퓨팅 환경에 적합한 모니터링 기법이다.

Edwards et al [18]은 메타-레벨 컴포넌트에 의한 모니터링 방법을 제안한다. 로봇 소프트웨어 컴포넌트는 상위 계층의 메타-레벨 컴포넌트에 의해 모니터링 된다. 메타-레벨 컴포넌트는 수행하는 역할에 따라 수집, 분석, 관리 컴포넌트로 구분되고 수집 역할을 수행하는 메타-레벨 컴포넌트가 응용 계층의 로봇 소프트웨어 컴포넌트에서 발생하는 이벤트 정보를 모니터링하여 상위 계층으로 전달한다.

Mahadevan et al [19]는 항공기에 사용되는 항공데이터 관성기준계기(air data inertial reference unit)의 시뮬레이션 사례를 통해 컴포넌트와 시스템 레벨에서 소프트웨어의 작동상태를 관리하는 2단계 접근방법을 제안하고 있다. 컴포넌트 수준은 단일 컴포넌트의 프로세스 실행종료 시간, 데이터 유효성(data validity)에 대해서 모니터링 한다. 컴포넌트 프로세스 실행시간을 모니터링하는 측면에서는 개념적인 수준에서 본 연구와 유사한 측면이 있다. 하지만 실행예외, 체크포인트-롤백 등을 포함하여 세부 모니터링 요소와 모니터링 된 데이터의 효율적인 관리측면에 대한 내용은 제시하고 있지 않다.

III. 로봇 소프트웨어 컴포넌트 실행 모니터링 방안

1. 로봇 소프트웨어 컴포넌트 모델링

소프트웨어 컴포넌트는 외부로 노출되고 계약적으로 명세된 인터페이스를 가진 시스템의 구성단위로 정의되며[20] 독립적으로 배포가능하고 제3자에 의해 재사용될 수 있다. OROCOS, OMG RTC, ORCA [21], 오프로스 등 대부분의 로봇 소프트웨어 컴포넌트 모델 역시 소프트웨어 컴포넌트 정의와 유사하나 컴포넌트 타입, 실행방식 등에서는 일부 차이를 보인다. 본 논문에서는 지식경제부 주관으로 개발하고 있는 오프로스 로봇 소프트웨어 컴포넌트 모델[7]을 기반으로 컴포넌트를 모델링한다.

오프로스 컴포넌트는 실행기에 의해 실행된다. 실행기는 컴포넌트 생명주기에서 컴포넌트 실행에 대한 기능을 수행하고 실행주기가 같은 컴포넌트는 동일 실행기에 할당되며 식 (1)과 같이 표현된다.

$$Executor_n = \{Comp_1, Comp_2 \dots Comp_n\} \quad (1)$$

실행기는 컴포넌트의 정해진 주기대로 onExecute() 메소

드를 호출하여 실행한다. 하나의 쓰레드로 여러 개의 컴포넌트를 처리하는 것은 불필요한 쓰레드 스위칭 시간을 줄여주는 장점이 있다.

로봇 소프트웨어 컴포넌트는 onExecute() 메소드가 호출되면 실행된다. 컴포넌트 실행시간($Comp_Et$, Execution time)은 컴포넌트의 실행이 시작되어 종료되는 시간을 의미한다. 실행시간은 식 (2)와 같이 표현된다.

$$Comp_Et = Comp_End - Comp_Begin \quad (2)$$

실행기의 전체 실행시간($Executor_Et_{total}$)은 실행기에 할당된 전체 컴포넌트의 실행시간을 더한 것과 같으며 식 (3)과 같다.

$$Executor_Et_{total} = \sum_{a=1}^n Comp_Et_a \quad (3)$$

실행기의 전체 실행시간은 각 컴포넌트의 실행주기($Comp_Ct$, Cycle time)를 정하는데 필요한 자료이다. 각 컴포넌트의 실행시간을 분석하여 설계단계에서 개발하고자 하는 로봇응용의 컴포넌트 실행주기 결정시 활용한다.

개별 컴포넌트의 실행시간($Comp_Et$)은 onExecute() 메소드를 실행할 때 소요되는 알고리즘 실행시간(E_At)과 인터페이스로 연결된 다른 컴포넌트의 서비스 코드 실행시간의 합으로 계산되며 식 (4)와 같이 표현된다.

$$Comp_Et = E_At + \sum_{i=1}^n E_St_i \quad (4)$$

컴포넌트 간의 데이터 교환은 데이터 포트를 통해 이루어진다. 컴포넌트에 의해 계산된 출력 값은 허용 범위를 갖는다. 즉 입력 포트를 통해 입력되는 데이터 값은 출력 포트의 정상범위(최솟값 ~ 최댓값)의 값을 가져야 한다. 예를 들어, 초음파 센서 컴포넌트는 센서 장치의 설계 허용 값 범위 내에서 데이터 값을 출력하고 다른 컴포넌트의 입력 포트를 통해 데이터 값이 입력된다. 이는 식 (5)와 같이 표현된다.

$$Min_Val_{Output} \leq Val_{Input} \leq Max_Val_{Output} \quad (5)$$

2. 로봇 소프트웨어 컴포넌트 실행 모니터링 요소

로봇 소프트웨어 컴포넌트 모델을 바탕으로 4가지 모니터링 요소를 도출한다. 모니터링 요소는 로봇 개발 시 분석·설계 단계, 실행 중에 활용 가능한 요소에 초점을 둔다. 모니터링 요소는 컴포넌트 실행 예외처리 관점, 컴포넌트 데이터 관점, 컴포넌트 체크포인트-롤백 관점, 그리고 컴포넌트 실행시간 관점이다.

첫 번째, 소프트웨어 컴포넌트 실행 시 발생 가능한 실행 예외처리 관점 모니터링 요소이다. 이는 컴포넌트의 코

그림 1. 컴포넌트 실행예외 관점 모니터링 요소.

Fig. 1. The monitoring factor of components runtime exceptions.

그림 2. 컴포넌트 데이터 관점 모니터링 요소.

Fig. 2. The monitoring factor of components data.

드 실행 중에 발생하는 시스템 에러로 컴포넌트의 알고리즘 코드를 실행하는 과정에서 발생하거나 또는 다른 컴포넌트의 서비스 코드 실행 중에 발생 가능하다.

두 번째, 컴포넌트 사이의 입/출력 데이터 관점 모니터링 요소이다. 컴포넌트의 데이터 포트를 통해 입/출력되는 수치적 데이터와 서비스 포트에서 서비스 코드를 실행 후 반환되는 값의 범위를 모니터링 한다. 컴포넌트의 데이터 관련 모니터링은 컴포넌트 실행엔진의 데이터 큐(queue)의 입/출력 함수와 서비스 호출 래핑(wrapping) 함수에 모니터링 코드를 삽입하여 구현이 가능하다.

세 번째, 컴포넌트 체크포인트-롤백 관점 모니터링 요소이다. 컴포넌트가 실행 중에 결점이 발생하면 정상적인 시점으로 컴포넌트 내부 정보를 복구해야할 경우가 발생한다. 컴포넌트가 동적 메모리의 할당/해제를 이용하거나 또는 사용하는 하드웨어 상태(hardware state)와 연관되어 있는 경우 체크포인트-롤백을 수행하는 경우 또 다른 결점이 발생할 수 있다. 따라서 본 논문에서는 Object Marshalling, Object Serialization 등의 기술을 이용하여 체크포인트를 생성하지 않고 컴포넌트에서 자체적으로 원하는 시점을 통보하여 해당 컴포넌트의 메모리를 복사하여 저장하는 방식을 사용한다. 컴포넌트는 스스로 실행 중 데이터 백업이 가능한 시점에 오프로스 실행엔진으로 체크포인트 저장 요청을 하며 해당 시점의 컴포넌트 내부정보를 저장하며 항상 최신 데이터를 저장한다.

마지막으로, 컴포넌트의 실행시간 관점 모니터링 요소이다. 서비스 로봇은 동적인 외부환경 하에서 운용되므로 로봇 컴포넌트들은 대부분 주기적으로 실행된다. 로봇 컴포넌트의 시간관련 모니터링 요소는 실행시간(Execution Time, Et), 실행주기(Execution Cycle Time, Ct), 데이터포트 입력주기(Data Input Cycle Time, Di_Ct), 데이터포트 출력주기(Data Output Cycle Time, Do_Ct)의 4가지 형태로 정의할 수 있다. 실행시간은 식 (4)에서와 같이 컴포넌트 알고리즘 코드 실행시간(E_At)과 다른 컴포넌트의 서비스 코드 실행시간(E_St)을 더한 값으로 정의한다.

그림 3. 컴포넌트 실행시간 관점 모니터링 요소.

Fig. 3. The monitoring factor of components execution time.

표 1. 모니터 데이터(monitor data) 레코드 정의.

Table 1. The definition of monitor data recode.

구분	시작 시간	종료 시간	예외 발생	입/출력 데이터
컴포넌트 실행	○	○	○	-
서비스 포트 실행	○	○	○	○
데이터 포트 입/출력	○	-	-	○

3. 컴포넌트 실행 모니터링 데이터 관리방안

로봇 소프트웨어 컴포넌트는 외부환경에 대한 주기적인 센싱을 위해 반복적으로 빠르게 수행된다. 모니터링 데이터에 대한 저장·관리는 로봇 소프트웨어 컴포넌트의 실행에 부가적으로 CPU, 시스템 메모리 등의 시스템 리소스를 사용하는 작업이다. 또한 다수의 컴포넌트가 실행되는 로봇 응용의 경우 많은 양의 데이터가 발생하게 된다. 따라서 모니터링 데이터 저장을 위한 간단한 데이터 처리 방법과 메모리 사용량을 줄이는 방안이 요구된다. 본 논문에서는 데이터 저장을 위하여 주로 사용되는 관계형 데이터베이스(RDB) 등을 이용하지 않고 모니터링 데이터 저장에 특화된 데이터 구조를 정의하여 사용한다.

컴포넌트의 모니터링은 컴포넌트의 코드 실행 이벤트 및 서비스 포트의 코드 실행 이벤트, 데이터 포트의 데이터 입/출력 이벤트 정보를 저장한다. 컴포넌트 모니터링에서 저장되는 이벤트 정보는 코드 실행의 경우 시작 및 완료 시간, 실행 중 에러 유무, 서비스 리턴 데이터를 가지며 데이터 입/출력 이벤트의 경우 데이터의 입/출력 시간 및 메소드 데이터를 저장하게 된다. 컴포넌트 모니터링에 저장되는 정보는 표 1과 같이 요약할 수 있다.

컴포넌트의 모니터링 데이터 저장은 표 1에 명시된 4가지 필드를 가지는 간단한 데이터 레코드를 모니터 데이터 구조체로 정의하여 모든 모니터링 이벤트를 저장한다. 컴포넌트에 따라 3가지 이벤트의 발생 빈도가 모두 다르므로 모니터 데이터를 이용한 모든 이벤트에 대한 동일한 레코드 구조는 효율적인 메모리 사용을 가능하게 한다. 모니터 데이터는 빠른 메모리 할당 및 해제를 위하여 로봇 소프트웨어 컴포넌트가 시작하는 시점에 미리 정의된 개수의 레코드를 미리 할당하여 모니터 레코드 풀(monitor record pool)을 구성하여 이벤트 발생 시 즉시 사용하도록 준비된다.

특정 이벤트가 발생하는 경우 모니터 레코드 풀에서 레코드를 할당 받아서 필요한 데이터를 기록한 후 해당 컴포넌트 또는 포트의 모니터 데이터 목록의 마지막에 추가한다. 특정 컴포넌트 또는 포트의 모니터 데이터는 추가적인 작업 없이 항상 시간 순서로 정렬된 리스트 구조를 가지게 되며 주로 최근 발생한 이벤트를 검색하는 컴포넌트 결점 탐지 등의 작업에서 유리한 구조를 가지게 된다. 또한 주기적으로 시스템에서 미리 지정된 시간이 지난 이벤트에 대한 모니터 데이터는 모니터 레코드 풀로 다시 이동하여 재사용하도록 하여 메모리 사용량이 일정하게 유지되도록 구성된다.

IV. 모니터링 모듈 설계 및 구현

본 장에서는 오프로스 컴포넌트 모델 기반 모니터링 모듈의 설계 및 구현에 대해 기술한다. 컴포넌트 실행엔진은 기존 실행엔진에 이벤트 모니터링을 위한 코드를 삽입하여 확장한다. 모니터링 모듈은 모니터 관리자, 이력 관리자로 구성되며 전체적인 소프트웨어 아키텍처는 그림 4와 같다.

1. 컴포넌트 모니터 관리자(component monitor manager)

컴포넌트 모니터 관리자는 오프로스 실행기, 컴포넌트 데이터/서비스 포트로부터 모니터링 이벤트를 수신하고 컴포넌트 이력 관리자에게 전달하는 역할을 수행한다. 오프로스 컴포넌트는 컴포넌트 클래스를 상속받아 onReset(), onStart(), onStop(), onExecute() 등의 콜백(callback) 메소드를 구현하도록 규정하고 있으며 컴포넌트는 오프로스의 실행 엔진에서 onExecute() 콜백 메소드를 호출하여 실행된다.

컴포넌트의 실행을 모니터링하기 위해 기존 실행엔진의 onExecute() 콜백 메소드 코드를 확장한다. 실행엔진의 onExecute() 메소드를 호출 직전에 컴포넌트 시작 이벤트를 생성하고 호출이 완료된 시점에서 컴포넌트 실행종료 이벤트를 컴포넌트 모니터링 관리자로 전달하여 실행시간 관점 요소 중 실행시간, 실행주기를 계산한다. 또한 구조적 예외 처리 기법을 이용하여 onExecute() 실행 중 발생하는 실행 예외를 감지하여 컴포넌트 실행종료 이벤트 발생 시 실행 예외 이벤트를 함께 전달한다.

오프로스 컴포넌트의 데이터 포트는 PortInterface로 부터 상속받아 구현되며 오프로스 CDL(component development library) 형태로 개발자에게 배포된다. 오프로스 컴포넌트는

그림 4. 로봇 소프트웨어 컴포넌트 실행 모니터링 모듈 아키텍처.

Fig. 4. Execution monitoring module architecture of the robot software components.

표 2. 컴포넌트/포트 이벤트 목록.

Table 2. The list of component/port events.

이벤트 명	이벤트 파라미터	비고
COMP_START	실행기/컴포넌트 ID	컴포넌트
COMP_COMPLETE	실행기/컴포넌트 ID 실행예외	실행/종료 시점 발생
SERVICE_START	시작 컴포넌트/포트 ID 도착 컴포넌트/포트 ID	서비스코드
SERVICE_COMPLETE	시작 컴포넌트/포트 ID 도착 컴포넌트/포트 ID 실행예외, 반환 값	실행/종료 시점 발생
DATA_PUSH	시작 컴포넌트/포트 ID	데이터포트
DATA_POP	도착 컴포넌트/포트 ID 데이터 값	입/출력 시점 발생
COMP_CHECKPOINT	컴포넌트 ID 데이터 값 및 크기	

데이터가 입력되는 경우 InputDataPortInterface의 pop() 메소드를 이용하여 출력 시에는 OutputDataInterface의 push() 메소드를 사용한다. 따라서 오프로스 CDL의 해당 코드를 확장하여 데이터 입력, 데이터 출력 이벤트를 입/출력 값과 함께 컴포넌트 모니터 관리자로 전달한다.

컴포넌트의 서비스 포트는 오프로스 컴포넌트가 생성되는 시점에서 다른 컴포넌트로 서비스 호출을 위하여 RequireServicePort 클래스를 확장하여 서비스 호출 래핑(wrapping) 메소드가 자동 생성된다. 래핑 메소드는 컴포넌트 실행 중에 다른 컴포넌트의 연결을 제공을 위하여 사용된다. 본 논문에서는 래핑 메소드에서 포트를 통해 연결된 컴포넌트의 서비스를 호출하는 코드를 확장하여 서비스 시작, 서비스 실행종료 이벤트를 전달한다. 또한 서비스 실행종료 이벤트의 경우 서비스 호출에 대한 결과 값과 구조적 예외처리 기법을 이용하여 서비스 수행 중에 발생하는 실행

예외 이벤트를 함께 컴포넌트 모니터 관리자에게 전달한다.

컴포넌트 모니터 관리자는 오프로스 실행기, 컴포넌트 데이터/서비스 포트로부터 표 2에 해당하는 모니터링 이벤트를 수신한다. 수신된 이벤트를 컴포넌트 이력 관리자에게 저장하기 위해 모니터링 이벤트가 발생한 컴포넌트의 데이터베이스 색인 테이블을 조회(queryComponent())하고 모니터링 이벤트를 저장할 새로운 레코드를 할당(getMonitorData()) 받는다. 마지막으로 컴포넌트의 체크포인트 정보와 할당 받은 모니터링 데이터 레코드를 데이터베이스 테이블에 저장한다.

2. 컴포넌트 이력 관리자(component history manager)

컴포넌트 이력 관리자는 컴포넌트 모니터 관리자로부터 수신된 모든 실행관련 이벤트 정보를 데이터베이스 형태로 저장하고 관리하는 역할을 수행한다. 이를 위해 이력 관리자는 컴포넌트/포트 모니터 테이블(component/port monitor tables)과 모니터 데이터 레코드 풀(monitor data records pool)로 구성된다.

컴포넌트/포트 모니터 테이블은 실행기에 의하여 실행되는 모든 컴포넌트/포트에 대한 결점 모니터링 이벤트 결점 상태 정보를 저장한다. 오프로스 실행엔진에서 실행되는 모든 컴포넌트는 하나의 CompContext 테이블이 생성되며 컴포넌트에 포함된 포트에 대하여 각각의 PortContext 테이블이 생성되어 1:N의 연결관계를 갖도록 한다. CompContext는 컴포넌트의 주기적 실행에 대한 모니터링 이벤트를 저장되고 PortContext에는 특정 포트에 대한 모니터링 이벤트가 저장된다.

모니터 데이터 레코드 풀은 빠른 메모리 할당과 해제를 위해서 로봇 소프트웨어 컴포넌트가 실행되는 시점에 미리 할당된 정해진 개수의 레코드 집합이다. 모니터 데이터(monitor data) 레코드는 표 1에서 살펴본 바와 같이 컴포넌트/서비스 포트의 실행 또는 데이터 입·출력 등의 하나의 이벤트에 대하여 발생시간, 실행시간, 입·출력 값을 저장하

그림 5. 컴포넌트 모니터 사용자 인터페이스(UI) 구성.

Fig. 5. User interface for the components monitor.

표 3. 컴포넌트 실행정보 목록.

Table 3. The list of component execution information.

구분	세부 항목	컴포넌트 실행정보
컴포넌트	컴포넌트	최대 실행시간(MaxEt)
		평균 실행시간(AvgEt)
		최소 실행시간(MinEt)
		평균 실행주기(AvgCt)
포트	서비스 포트	최대 실행시간(MaxE_St)
		평균 실행시간(AvgE_St)
		최소 실행시간(MinE_St)
		반환 최대값(Return MaxVal)
		반환 평균값(Return AvgVal)
		반환 최소값(Return MinVal)
	데이터 포트	데이터 포트 입력주기(Di_Ct)
		데이터 포트 출력주기(Do_Ct)
		최댓값(MaxVal)
		평균값(AvgVal)
		최소값(MinVal)

는 데이터의 최소 단위이다. CompContext, PortContext는 해당 컴포넌트/포트의 모니터링 이벤트에 대한 모니터 데이터를 시간 순서의 리스트 형태로 저장한다. 그리고 메모리 사용 효율을 위하여 설정된 시간 이후의 모니터링 이벤트를 삭제하여 최신 데이터를 유지하고 새로운 이벤트가 발생하는 경우 컴포넌트 모니터 관리자는 모니터 데이터 레코드 풀에서 새로운 모니터 데이터를 할당받아서 CompContext 또는 PortContext에 추가한다.

3. 컴포넌트 모니터 사용자 인터페이스(component monitor UI)

컴포넌트 개발자 또는 응용 개발자가 설계단계에서 로봇 소프트웨어 컴포넌트 실행정보를 활용하기 위해서는 사용자 인터페이스(UI, User Interface)를 통해 관련 정보를 적절하게 제공할 필요가 있으며 그림 5와 같이 구성한다. 컴포넌트 실행정보는 모니터링 요소로부터 도출된 이벤트로부터 계산되며 표 3과 같이 사용자에게 제공된다. 컴포넌트 실행정보를 클릭하면 그림 5의 오른쪽 팝업창 형태로 최근 30초 동안의 컴포넌트 실행상태를 그래프 형태 제공한다.

V. 실험 및 평가

1. 실험 환경

제안 모니터링 기법을 테스트하기 위해 감시정찰 임무를 수행하는 로봇 프로토타입을 개발하였으며 기본 사양은 표 4와 같다. 로봇 소프트웨어 컴포넌트 실행엔진은 오픈로즈 과제 홈페이지[22]에 공개된 실행엔진(ver. 20100630)을 활용하여 제안 컴포넌트 모니터링 모듈을 추가하고 기존 실행엔진의 일부 소스코드를 확장한다. 로봇 프로토타입은 개발의 신속성과 컴포넌트 실행에 대한 UI 구성을 위해 메인 보드는 노트북을 활용하고 모션보드, 하드웨어 센서 및 응용 컴포넌트로 구성한다.

모니터링 모듈이 추가된 오픈로즈 실행엔진을 노트북에 탑재하고 로봇 실행엔진의 컴포넌트 리포지토리(component

표 4. 감시정찰 로봇 기본 사양.

Table 4. Basic specification of the reconnaissance robot prototype.

구분	구성	사양
메인보드 (노트북)	CPU	Intel Pentium Dual CPU E2160 1.8GHz
	메모리	2GB DDR2 SDRAM
	운영체제	Window XP
모션보드	1EA	HRB-MTC-B10 Motion Board
실행엔진	1EA	OPRoS 실행엔진(ver.20100603)
센서	3EA	카메라, 초음파, IMU 센서
응용 컴포넌트	9EA	카메라, 음성, 얼굴인식, 데모 컴포넌트 등 9개

그림 6. 로봇 소프트웨어 컴포넌트 다이어그램.

Fig. 6. The software component diagram of the robot prototype.

repository)에는 9개의 소프트웨어 컴포넌트로 구성된 로봇 응용을 배포한다. 감시정찰 로봇 응용은 설정된 경로를 이동하면서 장애물을 회피하고 카메라 입력영상을 분석하여 침입자를 탐지하고 경보를 전파하는 임무를 수행한다. 그림 6은 감시정찰 로봇의 전체 컴포넌트 다이어그램 중에 카메라 영상을 받아 침입자를 탐지하고 음성을 통해 경보를 전파하는 부분의 소프트웨어 컴포넌트 다이어그램이다. 얼굴 인식 컴포넌트는 오픈로즈 홈페이지에 공개된 한국생산기술연구원(KITECH) 로봇 소프트웨어 컴포넌트[22]를 활용한다.

그림 6의 소프트웨어 컴포넌트 프로파일에 설정한 실행 시퀀팅 정보는 표 5에서 보는 바와 같다. 오픈로즈 컴포넌트 모델은 컴포넌트 실행에 대한 정보를 XML 형태의 프로파일로 정의하고 있다. 실행주기는 실행엔진에 의해 소프트웨어 컴포넌트가 주기적으로 실행되는 주기를 의미하고 FaultDemo, 카메라 컴포넌트는 40ms로 설정한다. 최대 실행 시간, 평균 실행주기는 모니터링을 위해 설정된 값이다. 스피치와 얼굴인식 컴포넌트는 FaultDemo 컴포넌트의 서비스 요청이 있을 경우에만 실행되는 수동형(passive) 컴포넌트로 설정한다.

표 5. 실험을 위한 컴포넌트 프로파일 설정 값.

Table 5. Component profile setting value for the test.

컴포넌트	실행주기 (Ct)	(단위: ms)	
		최대 실행시간 (Max.Et)	평균 실행주기 (Avg.Ct)
FaultDemo	40	50	50
Camera	40	30	50
Speech	Passive	N/A	N/A
KITECH_Skin	Passive	N/A	N/A

로봇 소프트웨어 컴포넌트가 실행되는 동안 관련 이벤트가 제대로 모니터링 되고 있는지 여부를 측정하기 위해 FaultDemo 컴포넌트의 실행시간을 지연시켜 보았다. 다시 말해, FaultDemo 컴포넌트가 실행된 후 약 30초 동안 얼굴 인식 컴포넌트의 분석 영상에 대한 서비스를 요청하지 않고 단순히 카메라 영상만을 처리하도록 하였다.

2. 실험 결과 및 분석

감시정찰 로봇의 전체 컴포넌트 다이어그램 중 카메라 영상을 받아 침입자를 분석·탐지하는 FaultDemo 컴포넌트에 대한 모니터링 결과 값은 표 6과 같다. 얼굴인식 컴포넌트로부터 서비스를 받기 전에는 컴포넌트의 최대 실행시간이 6.25ms임을 알 수 있다. 또한 실제 컴포넌트가 실행되는 평균 실행주기 역시 프로파일에 설정된 값인 40.00ms 대로 실행되고 있음을 알 수 있다. 하지만 카메라 영상에서 사람 얼굴에 대한 분석·탐지를 위해 얼굴인식 컴포넌트가 작동되고 관련 서비스를 수행하면서 FaultDemo 컴포넌트의 최대 실행시간은 43.54ms로 크게 증가함 됨을 알 수 있다. 또한 실제 실행되는 평균 실행주기 역시 44.52ms로 설계상의 목표인 40.00ms를 초과함을 알 수 있다. 이러한 결과를 기반으로, 응용 개발자는 설계상의 값을 확인하고 실제 모니

표 6. 로봇 소프트웨어 컴포넌트 모니터링 결과 값.

Table 6. Monitoring result of the robot software components.

컴포넌트	(단위: ms)			평균실행 주기	반 환 값
	컴포넌트 실행시간(Et)	최대	평균	최소	
FaultDemo (얼굴인식 실행 전)	6.25	1.18	0.00	40.00	-
FaultDemo (얼굴인식 실행 후)	43.54	9.23	0.00	44.52	-
얼굴인식 서비스	37.08	30.99	28.15	-	0.00

터링된 값을 기반으로 재설정한다.

그림 7은 표 6의 모니터링 결과 값에 대한 실제 모니터링 그래프이다. 컴포넌트 모니터 UI에서 정의한 모니터링 이벤트를 30초 동안 실시간으로 그래프 형태로 전시하고 결과 값은 텍스트 형태로 출력해 준다. 로봇 소프트웨어 컴포넌트 모니터링 모듈이 본 논문에서 제안하고 있는 컴포넌트 실행시간, 실행주기 등 관련 모니터링 요소를 실시간으로 잘 모니터링하고 있음을 확인할 수 있다.

VI. 결론

본 논문에서는 소프트웨어 컴포넌트 모델링을 통해 모니터링 요소를 도출하고 로봇 프레임워크 차원에서 효율적인 이벤트 모니터링 및 데이터 관리방안을 제시하였다. 모니터링 요소는 컴포넌트 실행 예외처리, 입/출력 데이터, 실행시간, 체크포인트-롤백 등 4가지 관점에서 도출하고 세부 모니터링 이벤트를 정의하였다. 또한 로봇 소프트웨어 컴포넌트에 적합한 이벤트 저장 레코드와 모니터 레코드 풀을 정의하여 효율적인 데이터 관리방안을 제안하였다.

그림 7. FaultDemo 컴포넌트의 실행시간(좌측상단) 및 평균 실행주기(우측상단), 얼굴인식 서비스 코드 실행시간(좌측하단) 및 데이터 반환 값(우측하단) 모니터링 결과.

Fig. 7. Execution time(upper-left) and average cycle time(upper-right) of FaultDemo component, execution time(lower-left) and data return value(lower-right) monitoring result of face recognition service code.

제안 개념의 타당성과 활용방안을 검증하기 위해서 오프 로스 로봇 프레임워크를 활용하여 모니터링 모듈을 구현하고 감시정찰 로봇 응용과 사용자 인터페이스를 통해 활용성을 검증하였다. 얼굴인식 컴포넌트의 실행 전·후의 컴포넌트 실행시간, 실행주기 등을 비교하여 모니터링 데이터를 비교하였다. 얼굴인식 컴포넌트의 실행 전 FaultDemo 컴포넌트의 실행시간은 6.25ms에서 실행 후 43.54ms로 증가함을 알 수 있었다. 또한 사용자 인터페이스를 통해 모니터링된 값을 실시간으로 전시해주는 것을 확인하였다.

제안 모듈은 로봇 설계단계에서 컴포넌트 분석을 위한 모니터링 도구로써 활용이 가능하고 로봇 시스템의 자기치유 도메인의 가장 근본이 되는 모니터링 분야에 활용될 수 있을 것으로 본다. 향후 모니터링 요소의 확장 및 소형 임베디드 소프트웨어에 적용 가능한 경량화된 모니터링 모듈에 대한 연구가 필요할 것으로 판단된다.

참고문헌

- [1] D. Brugali and P. Scandurra, "Component-based robotic engineering (Part I)," *IEEE Robotics & Automation Magazine*, vol. 16, no. 4, pp. 84-96, Mar. 2009.
- [2] D. Brugali, et al., *Software Engineering for Experimental Robotics: Trends in Robotic Software Frameworks*, Springer, 2007.
- [3] C. Côté, Y. Brosseau, D. Letourneau, C. Raievsky, and F. Michaud, "Robotic software integration using MARIE," *International Journal of Advanced Robotic Systems-Special Issue on Software Development and Integration in Robotics*, vol. 3, no. 1, pp. 55-60, Mar. 2006.
- [4] P. Soetens, *The OROCOS (Open Robot Control Software) Component Builder's Manual*, Ver.1.10.2, FMTC, 2007.
- [5] OMG, *Robotic Technology Component Specification*, Ver. 1.0, Apr. 2008.
- [6] Choulsoo Jang et al., "OPRoS: A new component based robot software platform," *ETRI Journal*, vol. 32, no. 5, pp. 646-656, Oct. 2010.
- [7] Korea Robot Standard Forum, *Components for OPRoS(Open S/W Platform for Robot) Part2: Component*, KOROS 1067-2:2009, 2009.
- [8] KITECH, OPRoS Component Reusability, OPRoS 2nd Phase 1st Workshop, Report, 2011.
- [9] J. M. Hart, *Windows System Programming*, Addison-Wesley, 2004.
- [10] B. G. Shim, B. H. Baek, and S. Y. Park, "A systematic robot fault-tolerance approach," *Proc. of 33rd IEEE COMPSAC*, vol. 1, pp. 624-625, 2009.
- [11] J. E. Cooling, *Software Engineering for Real-Time Systems*, Addison-Wesley, 2003.
- [12] R. Wilhelm, et al., "The worst-case execution-time problem-overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7 Issue 3, April 2008.
- [13] D. Garlan and B. Schmerl, "Model-based adaptation for self-healing systems," *Proc. of the First Workshop on Self-Healing Systems*, pp. 27-32, 2002.
- [14] J. Scott and R. Kazman, "Realizing and Refining Architectural Tactics: Availability," CMU/SEI, U.S., Report, no. 6, 2009.
- [15] D. Sundmark, A. Möller, and M. Nolin, "Monitored software components - a novel software engineering approach," *Proc. of the 11th Asia-Pacific Software Engineering Conference*, pp. 624-631, Nov. 2004.
- [16] F. Lüders, D. Flemström, and A. Wall, "Software components services for embedded real-time systems," *Proc. of the WICSA 2005*, pp. 278-279, 2005.
- [17] D. Garlan, B. R. Schmerl, and J. Chang, "Using gauges for architecture-based monitoring and adaptation," *Proc. Working Conference on Complex and Dynamic System Architecture*, Brisbane, Australia, Dec. 2001.
- [18] G. Edwards, et al., "Architecture-driven self-adaptation and self-management in robotics systems," *Proc. of the SEAMS 2009*, Vancouver, Canada, pp. 142-151, May 2009.
- [19] N. Mahadevan, A. Dubey, and G. Karsai, "Application of software health management techniques," *Proc. of the SEAMS 2011*, Honolulu, USA, pp. 1-10, May 2011.
- [20] C. Szyperski, *Component Software-Beyond Object-Oriented Programming*, Addison-Wesley, ACM Press, 2002.
- [21] A. Makarenko, A. Books, and T. Kaupp, "Orca: components for robotics," *In International Conference on Intelligent Robots and Systems(IROS)*, pp. 163-168, Oct. 2006.
- [22] OPRoS Project Homepage, <http://www.opros.or.kr>, 2011.

김 종 연

1998년 해군사관학교 졸업(학사). 2005년 연세대학교 컴퓨터과학과 졸업(석사). 2009년~현재 국방대학교 국방정보체계 전공 박사과정. 관심분야는 국방 무인화 무기체계, Self-Healing/Adaptive S/W 등.

윤 희 봉

1983년 해군사관학교 경영학사. 1986년 연세대학교 공학사. 1991년 미국 Naval Postgraduate School 전산공학 석사. 1998년 미국 Georgia Institute of Technology 전산공학 박사. 2002년~현재 국방대학교 국방관리대학원 국방정보 체계전공 교수. 2004년~현재 서강대/이주대 외래교수, 임베디드소프트웨어산업협회 운영위원, 국방소프트웨어산업협회 기획이사, 한국정보과학회 편집위원. 관심분야는 무인화 무기체계, 센서융합, 물체인식 및 추적, 미래전투체계, 임베디드 소프트웨어 등.