네트워크 로봇의 서비스 통합을 위한 템플릿 컴포넌트

Template Components for Service Integration of Networked Robot

김 주 형[1], 이 호 동[2], 박 귀 태†

Joo-Hyung Kim[1], Ho-Dong Lee[2], Gwi-Tae Park†

Abstract In a large scale robot system, one of important problems is software integration, which involves three elements: modularity, reusability and stability. By these issues, the degree of convenience of system integration, its required time and the performance of the system stability can be determined. In addition, the convenience of system management can be determined by the degree of completion of service components. This paper explains the template based service component (TBSC) for the integration of service components in networked robot. The important characteristics of TBSC are automatical execution and recovery process by a PnP supporting robot framework, which helps a system operator to manage a robot system comfortably. For easy implementation and system stability, we provide a service component creator and a verification tool to developers.

Keywords : Networked Robot, Middleware, Robot Framework, Template Based Service Component

1. 서 론

네트워크 로봇은 기존의 독립형 로봇이 가지고 있는 기능적 한계를 네트워크로 해결하는 것이 가능하다. 원격 조종을 통해 사람이 접근할 수 없는 위험한 장소로 로봇을 이동하거나, 네트워크를 통해 로봇끼리 서로 통신을 하여 주변 환경 정보를 서로 공유할 수 있고, 공유된 정보를 통해 협동 작업이 가능하다[1]-[4]. 게다가 네트워크의 발달로 인터넷을 통해 로봇에 새로운 기능을 부여할 수 있게 되어 로봇의 기능을 확장할 수 있고 다양한 기능을 처리할 수 있게 되었다. 이러한 특징을 이용하여 네트워크 로봇은 다양한 기능을 네트워크에 존재하는 서버로부터 제공받아 사람에게 서비스를 제공할 수 있는 서비스 로봇의 플랫폼으로써 가능성을 가지고 있다. 따라서 많은 연구자들이 이동 로봇이나 이족 보행 로봇을 네트워크와 연결할 수 있는 서비스 로봇으로 활용하기 위해 다양한 연구를 진행하고 있다[5]-[8].

이러한 서비스 로봇의 연구 분야중 해결해야할 중요한 문제는 로봇과 로봇의 서비스 컴포넌트 및 이들이 서로 연결되어 통신할 수 있는 프레임워크의 설계와, 프레임워크를 기반으로 서비스 컴포넌트들을 통합할 수 있는 방법에 관한 것이다. 지금까지의 연구는 로봇과 서비스 컴포넌트 간의 통신을 위한 프로토콜에 대한 설계나, 로봇과 서비스 컴포넌트간의 데이터 흐름을 관리하는 프레임워크의 설계에 초점이 맞추어져 있었다. 그러나 로봇과 서비스 컴포넌트가 많아진다면 통신 프로토콜 및 데이터의 흐름을 제어 및 관리하는 프레임워크가 복잡하게 된다. 이러한 문제는 미들웨어를 사용하여 이기종간의 통신이 가능하고 통신 프로토콜 문제를 간소화 하도록 하였다[9]-[10]. 그러나 개발자가 미들웨어를 숙지해야 하는 문제가 발생하였다. 게다가 새로운 서비스 컴포넌트를 시스템에 통합할 경우 시스템 전체를 수정 및 변경해야 하는 불편함이 수반된다. 따라서, 로봇 연구자들이 개발자가 안고 있는 미들웨어에 대

Received : Sep.08.2010; Reviewed : Dec.14.2010; Accepted: Dec.22.2010
※ 이 논문은 건설교통부가 출연하고 한국건설교통기술평가원에서 위탁 시행한 2006년도 첨단융합건설기술개발사업[과제번호:06첨단융합D01]의 지원으로 이루어짐.
† 교신저자 : 고려대학교 전기전자전파공학부 교수
[1] 고려대학교 전자전기공학과 박사과정
[2] 한국과학기술연구원 포토닉스센서시스템

한 이해 및 숙지에 대한 부담을 줄이고 개발된 컴포넌트의 재사용성을 향상시키고자 로봇 소프트웨어를 컴포넌트 기반으로 개발하여 서비스 개발자가 쉽고 편리하게 서비스를 개발할 수 있는 방법을 제안하였다. MSRDS(Microsoft Robotics Developer Studio)[11]는 .NET 프레임워크를 기반으로 동작하는 로봇 소프트웨어 개발 툴로써 컴포넌트의 이식성과 재사용성 문제를 해결하였다. 그러나, 고급기능의 로봇 소프트웨어 개발을 위해 개발자가 C# 프로그래밍 언어를 반드시 숙지해야 하는 부담이 있다. ERSP[12]는 순차적, 비동기적 실행을 지원하며 계층적 구조의 모듈을 제공하여 사용이 편리하지만 부족한 컨텐츠로 새로운 컴포넌트의 개발이 쉽지 않다. OROCOS[13]는 C++기반으로 로봇 소프트웨어를 작성할 수 있고 다양한 라이브러리를 제공하고 있지만 제어 구조는 관여하지 않고 있어 컴포넌트의 조합을 통한 시스템 통합에는 쉽지 않다. OpenRTM-aist[14]의 RTM(Robot Technology Middleware)은 국제 표준을 구현한 구현 모델이라는 것이 가장 큰 장점이다. 그러나, 컴포넌트가 로봇 자체의 구동을 위한 규격에 맞추어져 있어 네트워크 로봇으로 확장하기에는 부족하고 컴포넌트의 제어 구조를 모두 개발자가 부담해야 한다. OPRoS[15]는 정보통신부의 지원으로 개발된 로봇 소프트웨어 프레임워크로써 저작도구 및 시뮬레이터와 체계적인 개발 프로세서를 제공하여 다른 프레임워크와 차이를 보인다. 또한 프레임워크의 신뢰성과 안전성을 향상시켜 네트워크 로봇 및 단독 로봇의 동작시 오동작을 최소화하도록 개발되었다.

로봇 소프트웨어 제작을 위한 프레임워크와 소프트웨어 개발 도구 및 시뮬레이터, 유용한 컨텐츠의 제공은 개발자가 로봇 소프트웨어를 쉽고 편리하게 개발할 수 있도록 많은 연구와 개발이 진행되었다. 그러나 개발된 로봇 소프트웨어의 종류와 수가 많아지고 이들을 네트워크 기반 다개체 로봇 시스템에 통합하여 운영할 경우 몇 가지 문제가 발생한다. 첫째, 개발된 서비스 컴포넌트들은 로봇 시스템의 설계에 따라 다양한 연결 관계와 구조를 가지게 된다. 따라서, 특정 컴포넌트에서 오동작이 발생하면 다른 컴포넌트에도 악영향을 끼쳐 시스템 전체가 동작을 할 수 없는 상태가 될 수 있다. 둘째, 분산 컴퓨팅 환경으로 로봇 시스템을 구축하였을 때 특정 컴포넌트에서 발생한 오류가 다른 컴포넌트에도 영향을 끼쳐 시스템이 동작을 하지 않을 수 있다. 이러한 경우 시스템을 운영하는 시스템 관리자가 오류가 발생한 컴포넌트를 찾아서 복구하기가 매우 어려우므로 시스템 관리자의 부담이 매우 크다. 이는 시스템 관리자에게 제공되는 로봇 소프트웨어들의 모니터링 정보 부족이 원인중 하나이다. 셋째, 이러한 문제가 발생하는 이유는 개발된 컴포넌트가 시스템에 통합되기 전 단계에서 오동작을 발생시킬 수 있는지의 여부를 평가하는 과정이 없기 때문이다. 넷째, 시스템 관리자는 로봇과 서비스 컴포넌트를 수동으로 연결해야 하는 번거로움으로 인하여 서비스 컴포넌트에서 오동작이 발생한 경우 시스템을 초기 상태에서 다시 수동으로 연결하여 동작해야 하는 관리상의 어려움이 있다. 다섯째, 네트워크 로봇 개발시 개발자는 네트워크 통신에 대한 이해와 로봇과 서비스 컴포넌트의 제어 및 관리를 위한 상태 다이어그램의 디자인 문제를 안고 있다.

본 논문에서는 개발자의 편리한 서비스 컴포넌트 제작과 시스템 관리자가 적은 부담으로 로봇 시스템을 운영할 수 있도록 안정성을 가지는 네트워크 로봇을 위한 템플릿 기반의 서비스 컴포넌트를 제안한다. 제안하는 서비스 컴포넌트는 본 연구실에서 개발한 로봇 프레임워크[16]를 기반으로 동작한다. 또한, 컴포넌트의 재사용성을 위해 실행파일 형태로 모듈화 시키고, 여러 대의 컴퓨터에서 병렬로 실행가능하다. 제안하는 서비스 컴포넌트의 특징은 다음과 같다.

- **PnP(Plug and Play) 기능의 로봇 프레임워크**
  - 서비스 컴포넌트 실행기가 로봇과 서비스 컴포넌트를 실행시키고 로봇-서비스 컴포넌트 연결기가 구동된 로봇과 서비스 컴포넌트를 연결시킴
  - 오동작을 일으킨 서비스 컴포넌트를 자동 복구 및 구동하여 시스템 운영의 안정성을 향상
- **서비스 컴포넌트의 상태 정보 제공**
  - 컴포넌트의 통신 및 실행 상태 정보를 제공
  - 컴포넌트의 오류 원인 제공
- **서비스 컴포넌트 생성**
  - 모듈화 및 안정성 향상을 위한 통신 및 제어 인터페이스 정의와 실행 절차의 정의
  - 목적에 맞는 서비스 컴포넌트 제공

- 비디오, 오디오 정보의 스트리밍 데이터 전송
- 통신 및 데이터의 리스트 관리, 병렬 처리와 기타 유용한 함수를 라이브러리로 제공

• **서비스 컴포넌트 검증**
- 개발자에 의한 컴포넌트의 자체 테스트를 위한 검증용 컴포넌트 제공
- 검증기에 의한 서비스 컴포넌트의 제어 함수 검증 및 검증 결과 제공

논문의 구성은 다음과 같다. 2장에서는 서비스 통합을 위해 프레임워크와 템플릿 기반의 서비스 컴포넌트가 공용으로 갖추어야할 인터페이스에 대해 설명하고, 3장에서는 공용 인터페이스를 기반으로 생성할 수 있는 서비스 컴포넌트의 종류와 기능에 대해 설명한다. 4장에서는 개발한 서비스 컴포넌트를 프레임워크에 통합하기 이전의 자체 검증에 대한 내용을 기술한다. 5장에서는 프레임워크에 서비스 컴포넌트를 통합하는 방법에 대해 설명하고 6장을 끝으로 결론을 맺고자 한다.

2. 공용 인터페이스 정의

공용 인터페이스는 프레임워크에 서비스 컴포넌트가 모듈단위로 안정성을 갖고 통합될 수 있도록 다음 두 가지 내용을 포함하고 있다. 첫째는 통신을 위해 일반화된 데이터 통신 함수 및 서비스 컴포넌트를 제어하기 위한 인터페이스를 정의하는 것이고, 둘째는 서비스 컴포넌트간의 데이터 통신 상태를 감시하는 메커니즘에 대한 구현 부분을 포함하는 것이다. 또한, 공용인터페이스에서 정의하는 함수들은 미들웨어를 이용하여 구현한 함수를 일반적인 C/C++언어로 감싼 형태로 개발자가 미들웨어에 대한 지식이 없어도 서비스 개발이 가능하도록 하였다. 본 논문에서 정의한 공용 인터페이스는 프레임워크 및 서비스 컴포넌트에 공통적으로 적용된다.

2.1 데이터 통신 함수

공용인터페이스에서 정의하는 데이터 통신 함수는 로봇과 서비스 및 서비스간의 데이터 송수신을 위해 동기화 및 비동기화 통신으로 나누어 표 1과 같이 세 가지 함수로 구성된다.

표 1. 데이터 통신 함수

동기화 통신	void FuncCall(address, input_data, output_data);
비동기화 통신	void FuncSend(destination, input_data); void FuncReceive(source, input_data);

표 1의 함수는 서비스의 주소를 이용하여 원하는 서비스로 데이터를 송신하면, 요청을 받은 서비스 내부에서 개발자에 의해 구현된 알고리즘이 요청받은 데이터를 처리한 후, 데이터를 송신한 서비스로 처리한 결과를 반환한다. 동기화 통신 방식을 이용하여 데이터를 송수신할 경우에는 “FuncCall” 함수를 사용한다. “FuncCall” 함수의 입력은 상대방 서비스의 주소인 “address” 와 송신할 데이터인 “input_data” 이고, 함수의 반환 결과는 “output_data” 이다. “FuncCall” 함수를 이용하면 상대방 서비스가 결과 데이터인 “output_data” 를 반환해야 함수가 종료된다. 반면에, 비동기화 통신 방식을 이용하여 데이터를 송신할 경우에는 “FuncSend” 함수를 이용한다. “FuncSend” 함수의 입력은 상대방 서비스의 주소인 “destination” 과 송신할 데이터인 “input_data” 이다. “FuncSend” 함수가 동기화 통신 함수인 “FuncCall” 과 다른 점은 상대방 서비스에 요청 데이터만 보내고 함수를 종료한 후, 다른 작업을 처리할 수 있다. 대신 “FuncReceive” 함수를 통해 상대방 서비스로부터 결과가 반환되기를 기다린다. 각 함수의 인자에서 서비스의 주소 인자인 “address”, “destination”, “source” 는 문자열로 정의하고, “input_data” 및 “output_data”는 기본적으로 버퍼형식의 데이터와 데이터 크기로 구성된 구조체를 함수의 인자로 한다. 버퍼형식의 데이터는 다양한 형태의 음성 및 영상 정보와 다양한 센서 정보를 표현할 수 있다. 개발자는 다른 서비스의 요청에 따른 처리 결과를 반환할 수 있도록 서비스 컴포넌트의 알고리즘을 “FuncCall” 함수와 “FuncReceive” 함수의 내부에 구현할 수 있다. 함수의 내부를 구현하는 내용은 3장 템플릿 기반 서비스 컴포넌트에서 다루도록 한다.

2.2 서비스 컴포넌트의 제어 인터페이스

서비스 컴포넌트는 독립적인 모듈단위로 동작하는 프로그램이다. 따라서 프레임워크를 통해 서비스 컴포넌트를 제어하기 위해서는 각 서비스 컴포넌트마다 동일한 제어 인터페이스가 제공되어야 한다. 제어 인터페이스의 함수에

대한 설명은 표 2와 같다. 개발자는 표 2의 각 함수 내부에서 처리할 내용을 구현할 수 있다.

그림 1은 서비스 컴포넌트의 제어 흐름을 나타낸다. 각각의 블록은 프레임워크에서 제어할 수 있는 하나의 함수로 구성되어 개발자가 각 블록에 해당하는 함수의 내부를 구현할 수 있다. 또한, 각각의 함수는 입력 및 출력에 대한 인자가 모두 void 형태로 구성되어 있다. 이는 PnP 가능한 프레임워크에 의해 서비스 컴포넌트가 동적으로 구동 및 종료할 수 있고, 프레임워크를 기반으로 하는 서비스 컴포넌트들의 통합을 위해 일관성을 유지하기 위함이다. 또한, 그림 1에서 표현한 제어 흐름은 개발자가 구현한 서비스 컴포넌트의 알고리즘이 프레임워크에서 안정성을 갖고 구동할 수 있도록 알고리즘의 동작을 위한 흐름을 제공한다.

표 2. 제어 인터페이스의 함수들

InitObject	프레임워크에 의해 서비스 컴포넌트가 처음 구동될 때 호출됨. 알고리즘을 구동하기 위한 초기화 작업으로 예를 들면 메모리 할당이 해당됨.
Start	서비스 컴포넌트의 알고리즘을 구동하기 위해 호출됨. 일반적으로 쓰레드(thread)로 처리하는 알고리즘을 시작하기 위해 사용됨.
Pause	서비스 컴포넌트의 알고리즘을 잠시 중단함. 서비스 컴포넌트가 가지고 있는 데이터 및 상태 정보는 백업함.
Resume	중단된 서비스 컴포넌트의 알고리즘을 다시 구동시키기 위해 호출됨. Pause시 백업한 데이터 및 상태 정보를 다시 로드 후 Start 상태로 전이하기 위한 모든 절차가 완료되면 Start를 수행함.
Stop	서비스 컴포넌트의 알고리즘을 완전히 정지시킴으로써 더 이상 기능을 수행할 수 없는 상태로 만듦. 데이터 및 상태 정보는 모두 지워짐.
FiniObject	서비스 컴포넌트를 완전히 종료하기 전에 호출됨. 알고리즘을 수행한 후에 처리해야 할 작업을 위한 함수로써 예를 들면 메모리 해제가 해당됨.

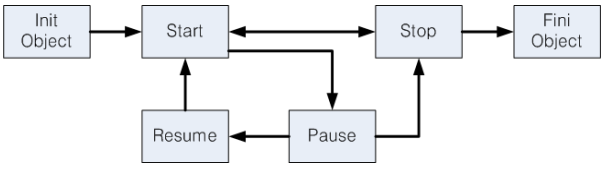

그림 1. 서비스 컴포넌트의 제어 흐름

2.3 통신 상태 감지 메커니즘

통신 상태 감지 메커니즘은 동기화 통신방식에서 오는 통신 오류를 감시하여 서비스 컴포넌트간의 통신에 문제가 발생하지 않도록 하기 위한 것이다. 동기화 통신방식의 문제는 데이터의 요청이 시작되고 상대방의 응답이 반환될 때까지 다음 과정을 수행할 수 없는 차단(blocking)된 상태가 수반된다. 그림 2는 동기화 통신을 통해 발생할 수 있는 통신 오류를 보여주기 위해 서비스 컴포넌트 A가 상대방 서비스 컴포넌트 B에 데이터를 요청하고 결과를 받는 과정을 나타낸다.

서비스 컴포넌트 A는 ①번 과정에서 통신 함수를 통해 데이터를 송신하고 서비스 컴포넌트 B로부터 결과가 반환되어 ⑥번 과정이 완료될 때까지 다음 작업을 진행 할 수 없다. 그러나 서비스 컴포넌트 B가 ③번 과정에서 수신한 데이터를 알고리즘이 처리하는 도중에 결과를 반환할 수 없는 상황에 놓이면 서비스 컴포넌트 B의 알고리즘은 ④번 과정에서 결과를 반환할 수 없고, ⑤번 과정이 발생하지 않으므로 서비스 컴포넌트 A는 서비스 컴포넌트 B의 응답을 계속 기다리게 된다. 반면에, 서비스 컴포넌트 B의 알고리즘이 결과를 반환하여 ④번 과정까지 완료를 하였으나 프레임워크의 통신 함수에서 문제가 발생하여 ⑤번 과정이 이루어지지 않을 수도 있다. 만약, 이러한 서비스가 그림 3과 같이 직렬로 연결되어 있다면 최초에 요청을 한 서비스 컴포넌트는 결과를 무한히 기다릴 수밖에 없다.

통신 상태 감지 메커니즘은 동기화 통신 방식으로 데이

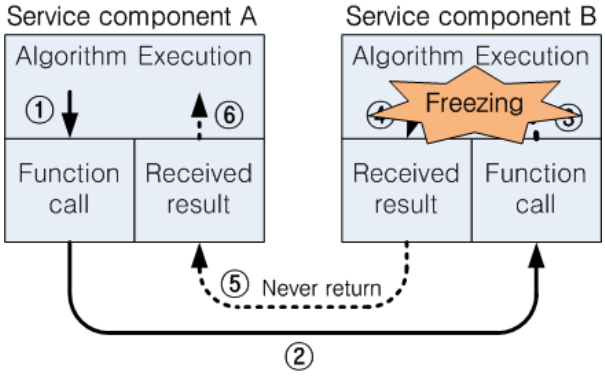

그림 2. 서비스 컴포넌트의 블록화 현상

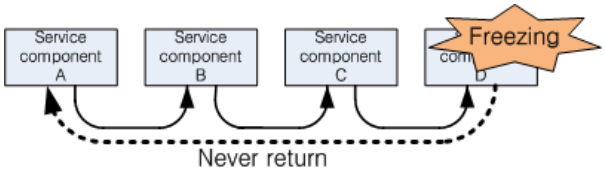

그림 3. 직렬로 연결된 서비스 컴포넌트의 블록화 현상

터를 송수신하는 경우에 적용된다. 그림 4는 블록화 현상을 감지하기 위한 통신 상태 감지 메커니즘을 보여준다. 실선은 데이터의 흐름을 나타내고, 점선은 모니터링 쓰레드 내부에서 주고받는 통신 상태의 로그에 대한 흐름을 나타낸다.

서비스 컴포넌트 A의 “Procedure” 에서 서비스 컴포넌트 B에 요청을 하는 경우, 서비스 컴포넌트 A의 “Monitor 1” 을 통해 데이터가 미들웨어 계층으로 전달되고 “Monitor 4” 로 응답 대기에 대한 감시 정보를 담고 있는 로그를 전달한다. 만약, “Monitor 4” 는 일정 시간이 초과한 후에도 아무런 데이터를 받지 않으면 “Procedure” 와 프레임워크에 데이터 통신 오류를 보고하게 된다. “Monitor 1” 을 통과한 데이터는 네트워크를 통해 서비스 컴포넌트 B로 전달되어 “Monitor 2” 를 통해 알고리즘으로 전달된다. 이때 “Monitor 2” 는 알고리즘의 결과 반환에 대한 감시 정보를 담고 있는 로그를 “Monitor 3” 에 전달한다. 만약, 일정 시간이 초과한 후에도 알고리즘으로부터 아무런 결과가 반환되지 않는다면 “Monitor 3” 은 프레임워크에 데이터 통신 오류를 보고하게 된다. 네 개의 모니터링으로부터 전달받은 통신 상태에 대한 정보는 프레임워크가 서비스 컴포넌트의 오류 발생 여부를 판단하기 위해 사용한다. 만약, 모니터링의 정보에 의해 서비스 컴포넌트에서 오류가 발생하였다고 판단이 되는 경우, 프레임워크가 해당 서비스 컴포넌트를 종료하고 재시작을 한다.

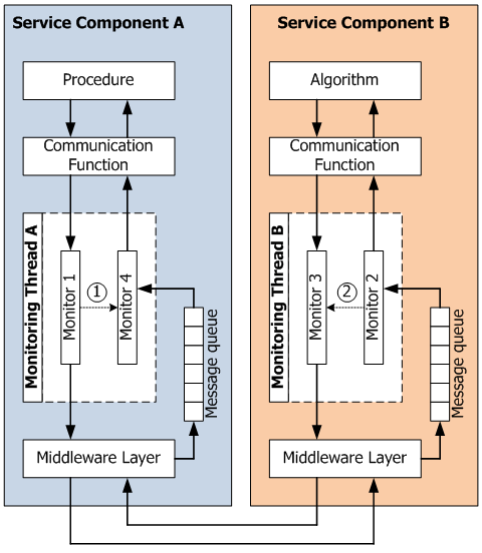

그림 4. 통신 상태 감지 메커니즘

3. 템플릿 기반의 서비스 컴포넌트

3.1 서비스 컴포넌트의 구성

그림 5의 템플릿 기반 서비스 컴포넌트는 공용 인터페이스와 지능 소프트웨어 및 검증 소프트웨어로 구성되어 있다. 지능 소프트웨어는 프레임워크에 통합하기 위해 개발자가 알고리즘을 구현하기 위한 것이고, 검증 소프트웨어는 개발자가 여러 가지 테스트 코드를 구현하여 검증 도구를 통해 지능 소프트웨어의 안정성을 검증하기 위한 것이다. 따라서 지능 소프트웨어와 검증 소프트웨어는 동일한 서비스 컴포넌트로써 구성 요소가 동일하다. 지능 소프트웨어에서 구현 부분은 개발자가 알고리즘을 구현할 수 있도록 사용자 구현 함수를 제공한다. 또한, 개발자에게 필요한 여러 유틸리티를 프레임워크에서 제공하여 개발자의 부담을 덜어주고, 구현한 알고리즘의 결과를 GUI로 확인이 가능하여 개발의 편리성을 향상시킨다. 구현된 서비스 컴포넌트는 실행파일 형태로 프레임워크가 설치된 컴퓨터라면 어디에서나 실행 가능하다.

템플릿 기반 서비스 컴포넌트는 공용 인터페이스와 지능 소프트웨어 및 검증 소프트웨어로 구성되어 있다. 지능 소프트웨어는 프레임워크에 통합하기 위해 개발자가 알고리즘을 구현하기 위한 것이고, 검증 소프트웨어는 개발자가 여러 가지 테스트 코드를 구현하여 검증 도구를 통해 지능 소프트웨어의 안정성을 검증하기 위한 것이다. 따라서 지능 소프트웨어와 검증 소프트웨어는 동일한 서비스 컴포넌트로써 구성 요소가 동일하다. 지능 소프트웨어에서 구현 부분은 개발자가 알고리즘을 구현할 수 있도록 사용자 구현 함수를 제공한다. 또한, 개발자에게 필요한 여러 유틸리티를 프레임워크에서 제공하여 개발자의 부담을 덜

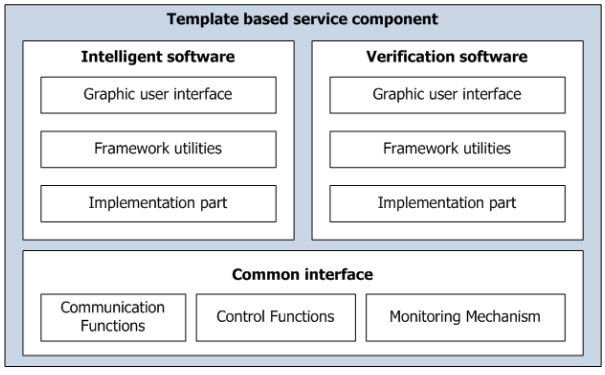

그림 5. 템플릿 기반 서비스 컴포넌트의 구성 요소

어주고, 구현한 알고리즘의 결과를 GUI로 확인이 가능하여 개발의 편리성을 향상시킨다. 구현된 서비스 컴포넌트는 실행파일 형태로 프레임워크가 설치된 컴퓨터라면 어디에서나 실행 가능하다.

3.2 서비스 컴포넌트 생성기

템플릿 기반 서비스 컴포넌트는 서비스 컴포넌트 생성기에 의해 구현 가능한 코드 형태로 생성된다. 개발자는 서비스 컴포넌트 생성기에서 서비스의 이름을 입력하고 서비스의 종류를 선택하여 원하는 형태의 서비스 컴포넌트를 생성할 수 있다. 또한, 생성된 서비스 컴포넌트는 서비스의 종류에 따라 데이터 통신방식이 다르게 설정된다. 서비스 컴포넌트 생성기는 개발자가 입력한 서비스 이름과 종류에 의해 지능 소프트웨어와 검증 소프트웨어를 생성한다. 지능 소프트웨어와 검증 소프트웨어는 완벽하게 분리된 코드 형태로 생성되어, 개발자가 각각의 역할에 맞게 구현한 후에 서비스 검증단계에서 모두 사용된다.

3.3 서비스 컴포넌트의 종류

서비스 컴포넌트의 종류는 네트워크 로봇의 특징을 고려하여 표 3과 같이 네 가지로 구분하여 정의한다. 그림 6은 각 서비스 컴포넌트간의 관계를 나타낸다.

표 3. 서비스 컴포넌트의 종류

로봇 서비스	로봇의 센서 정보를 수집하여 다른 서비스 컴포넌트에 전달하거나 요청한 서비스로부터 결과를 받아 하드웨어를 제어하는 서비스
단일 서비스	간단한 알고리즘으로 구성되어 있고 다른 서비스를 사용하지 않는 독립적인 서비스
복합 서비스	단일 서비스 및 다른 복합 서비스의 결과를 이용하여 새로운 결과를 반환하는 서비스
전역 서비스	프레임워크에 통합된 서비스의 상태를 모니터링하거나 제어하기 위한 목적의 서비스

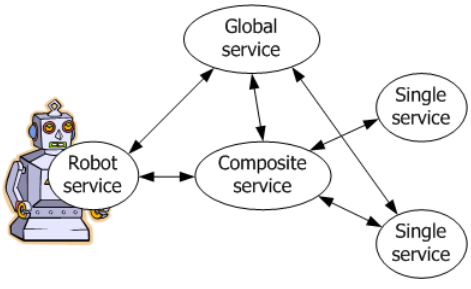

그림 6. 서비스 컴포넌트의 관계

3.4 서비스 컴포넌트의 구현

개발자는 지능 소프트웨어내의 사용자 구현 함수를 통해 알고리즘을 구현할 수 있다. 사용자 구현 함수는 표 1에서 설명한 “FuncCall” 과 “FuncReceive” 함수이다. 서비스 컴포넌트는 C/C++ 언어로 구현 가능한 코드 형태를 제공하므로 개발자가 두 함수의 내부에 구현을 하면 된다. 표 4는 두 함수를 구현하는 예를 나타낸다.

개발자는 필요에 따라 사용자 구현 함수 외에 표 2에 기술된 제어 인터페이스 함수의 내부도 구현할 수 있다.

표 4. 서비스 컴포넌트의 사용자 구현 함수

```
// 동기화 통신을 하는 경우 이 함수를 구현한다.
void FuncCall(address, input_data, output_data) {
    // 버퍼 형태의 input_data를 알고리즘에 필요한
    // 데이터 형태로 변환한다.
    // 알고리즘을 수행한다.
    // 알고리즘의 결과를 버퍼 형태의 output_data로 변환한다.
}
```

```
// 비동기화 통신을 하는 경우 이 함수를 구현한다.
void FuncReceive(source, input_data) {
    // 버퍼 형태의 input_data를 알고리즘에 필요한
    // 데이터 형태로 변환한다.
    // 알고리즘을 수행한다.
    // 알고리즘의 결과를 버퍼 형태의 데이터로 변환한다.
    // FuncSend함수를 이용하여 source로 결과를 반환한다.
}
```

4. 서비스 검증

개발자가 지능 소프트웨어에 알고리즘을 구현하고 실행파일 형태로 서비스 컴포넌트를 생성하면 프레임워크에 통합할 수 있는 상태가 된다. 그러나 프레임워크에 통합하기 이전에 구현된 지능 소프트웨어가 안정적으로 구동하는지 검증할 필요가 있다. 이를 위해 개발자는 검증 소프트웨어에 지능 소프트웨어를 검증하기 위한 절차를 구현할 수 있다. 개발자는 두 가지 소프트웨어의 구현을 마치고 실행파일 형태의 서비스 컴포넌트를 생성하면 서비스 검증도구를 이용하여 지능 소프트웨어의 안정성을 검증할 수 있다. 서비스 검증도구는 네트워크 로봇 시스템을 가상화하여 지능 소프트웨어를 자동으로 구동 및 제어하고 정해진 검증 절차를 수행한다. 그림 7은 서비스 검증도구의 다이어그램을 나타낸다. 검증 환경에서 가상화 로봇 시스템은 서비스 컴포넌트를 동적으로 구동하고 제어하는 부

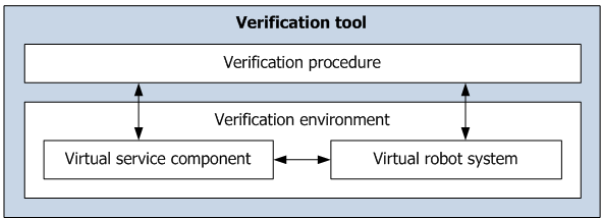

그림 7. 서비스 검증도구의 다이어그램

분으로써 실제 프레임워크의 제어 부분과 동일한 동작을 한다. 가상화 서비스 컴포넌트는 로봇 서비스를 가상화한 것으로 실제 로봇에 탑재되는 서비스 컴포넌트와 동일한 동작을 한다. 검증 절차는 개발자에 의한 데이터 통신 검증과 서비스 검증 도구에 의한 제어 인터페이스 검증으로 구분된다.

4.1 데이터 통신 검증

데이터 통신 검증은 개발자가 구현한 데이터 통신 함수가 오류 없이 동작하는지 검증 소프트웨어를 통해 수동으로 검증하는 단계이다. 서비스 검증 도구에서는 개발자가 구현한 지능 소프트웨어와 검증 소프트웨어가 서로 통신할 수 있도록 연결해주는 역할을 한다. 그림 8은 두 소프트웨어를 이용한 검증 방법을 보여준다. 검증 소프트웨어(그림 8의 상단 프로그램)에서 개발자가 데이터를 보내면 지능 소프트웨어(그림 8의 하단 프로그램)는 알고리즘을 처리한 결과를 다시 검증 소프트웨어로 반환한다. 이 과정에서 알고리즘의 동작 오류 여부 및 통신 상태 판단을 할 수 있다.

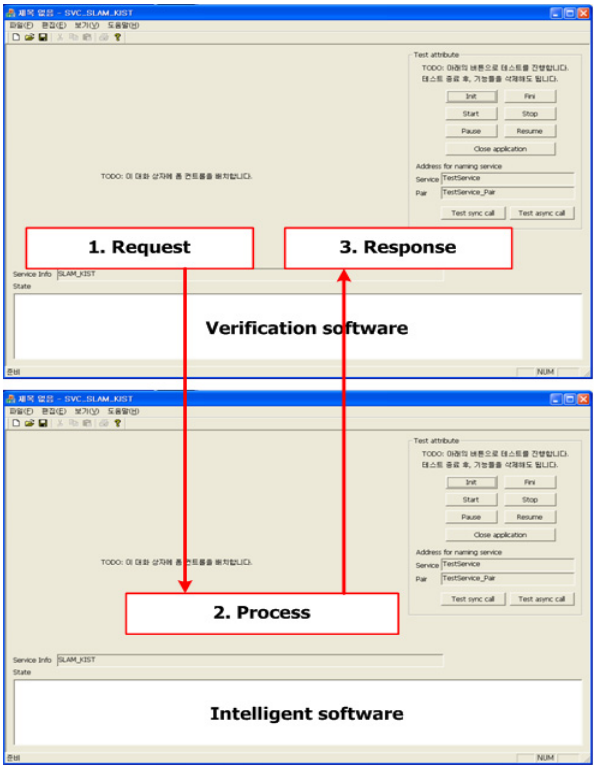

그림 8. 개발자에 의한 서비스 검증

4.2 제어 인터페이스 검증

제어 인터페이스 검증은 서비스 검증 도구가 지능 소프트웨어의 제어 인터페이스 함수를 그림 1의 흐름에 따라 호출하면서 동작의 오류 여부를 검증한다. 검증 도구의 가상화된 로봇 시스템에 의해 지능 소프트웨어를 자동으로 구동하고 제어 흐름에 따라 함수를 호출한 후, 종료까지 오류가 없는지 검증한다. 검증 도구에 의해 검증이 완료된 후, 개발자는 검증 도구가 보고하는 오류 보고서를 확인하여 지능 소프트웨어의 구동에서 종료단계까지 이르는 통신 및 알고리즘 동작 상태에 대한 내용을 보완할 수 있다. 그림 9는 검증 도구의 검증 완료 후 보고서 출력을 나타낸다.

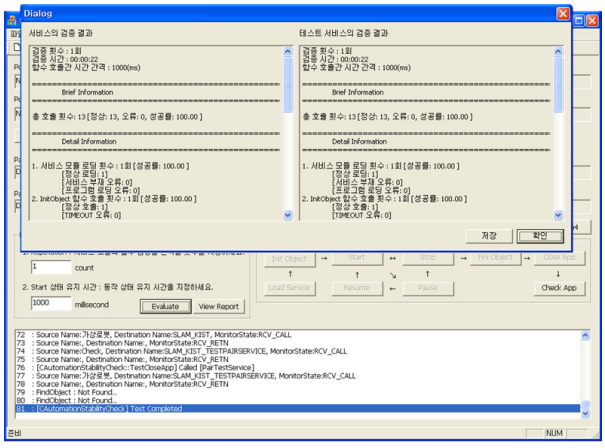

그림 9. 서비스 검증 도구의 보고서 출력 화면

5. 서비스 컴포넌트와 프레임워크의 통합

검증 단계까지 통과한 서비스 컴포넌트는 실행 파일 형태로 프레임워크에 통합할 수 있다. 새로운 서비스 컴포넌트를 사용하기 위해 프레임워크에 서비스 컴포넌트의 정보를 등록해야 한다. 프레임워크의 서비스 컴포넌트 관리자를 통해 XML형태로 서비스의 이름과 실행 위치를 등록

할 수 있다. 또한, 로봇이 새로운 서비스 컴포넌트를 이용하기 위해 로봇 서비스에도 서비스의 이름을 등록해야 한다. 그림 10처럼 개발자는 새로운 서비스 컴포넌트의 정보를 등록만 하면 프레임워크에서 자동으로 로봇과 서비스가 통신할 수 있도록 연결을 해준다.

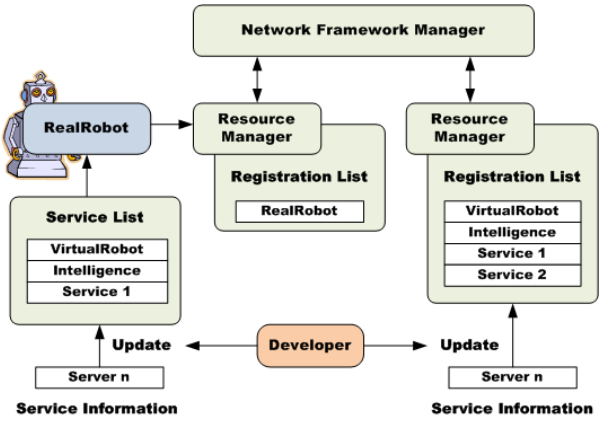

그림 10. 프레임워크에 서비스 컴포넌트의 등록 및 통합

6. 결 론

본 논문에서는 네트워크 로봇의 서비스 통합을 위한 템플릿 기반의 서비스 컴포넌트에 대해 기술하였다. 규모가 큰 로봇 시스템에서 여러 서비스 컴포넌트를 통합하고 시스템 관리자가 시스템을 운영할 때 발생할 수 있는 문제를 해결하고자 PnP 가능한 로봇 프레임워크를 기반으로 서비스 컴포넌트가 동적으로 구동하고 오동작을 일으키는 경우 자동으로 복구가 되도록 하여 시스템 관리자가 시스템 운영 및 관리에 대한 부담을 줄일 수 있을 것으로 기대한다. 또한, 서비스 컴포넌트의 모듈화와 안정성 향상을 위한 검증을 통해 개발자가 구현한 컴포넌트를 충분히 검증함으로써 시스템 통합 및 운영 단계에서 컴포넌트의 수정 및 보완 작업을 줄일 수 있을 것으로 기대된다.

참고문헌

[1] Ping Li, Wenjuan Lu, Zengqi Sun, “Transport layer protocol reconfiguratino for network-based robot control system.” 2005 IEEE Proceedings Networking, Sensing and Control, 19-22 March 2005. pp. 1049-1053.

[2] Young-Shin Kim, Hong-Seong Park, Wook-Hyun Kwon, “An Architecture for a Network Based Robot Control System.” 1999 7th IEEE International Conference on Emerging Technologies and Factory Automation, 18-21 October,1999. Vol.2, pp.875-880.

[3] Kazuhiko Kawamura, Phongchai Nilas, Kazuhiko Muguruma, Julie A. Adams, and Chen Zhou, “An Agent-Based Architecture for an Adaptive Human-Robot Interface.” Preceedings of the 36th Hawaii International Conference on System Sciences, 2002.

[4] A. Julie, Adams, Hande Kaymaz-Keskinpala, “Analysis of Perceived Workload when using a PDA for Mobile Robot Teleoperation.” Proceedings of the 2004 IEEE International Conference on Robotics & Automation, April 2004. Vol.4, pp.4128-4133.

[5] Joo-Ho Lee, G. Appenzeller, Hideki Hashimoto, “Physical agent for a sensored, networked and thinking space.” Proceedings of the 1998 IEEE International Conference on Robotics and Automation, 16-20 May 1998. Vol.1 pp.838-843.

[6] 오상록 정보통신부 지능형서비스 로봇 PM, “네트워크기반 지능형 서비스 로봇”, 로봇 공학회, 2004년, 제1권. 1호.

[7] 조영조 ETRI 지능형로봇연구단장, “지능형 서비스 로봇과 URC(Ubiquitous Robot Companion)”, http://javava79.isloco.com

[8] 유범재 한국과학기술연구원 지능로봇연구센터, “네트워크(IT)기반 로봇”, http://blog.naver.com

[9] Mizukawa M. “Robot technology (RT) trend and standardization” In: Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts, 2005: 249-253.

[10] Gill C, Smart B. “Middleware for Robots?” In: Proceedings of AAAI Symposium Workshop on Intelligent and Distributed Embedded Systems, 2002: 1-5

[11] MSRS, Microsoft, http://www.microsoft.com/robotics

[12] ERSP, Evolution Robotics, http://www.evolution.com

[13] OROCOS, http://www.orocos.org

[14] OpenRTM-aist, http://www.openrtm.org

[15] OPRoS, http://www.opros.or.kr

[16] Ho-Dong Lee, Dongwon Kim, Joohyung Kim,

Gwi-Tae Park, “PnP Supporting Middleware Framework for Network Based Humanoid” The Journal of Korea Robotics Society, 2008. Vol.3, No.3, pp.255-261.

박 귀 태

1975 고려대학교 전기공학과 공학사
1978 고려대학교 전기공학과 공학석사
1981 고려대학교 전기공학과 공학박사
1981~현재 고려대학교 전기전자전파공학부 교수
관심분야 : 지능 시스템
E-mail : gtpark@korea.ac.kr

김 주 형

2006 고려대학교 전산학과 이학사
2008 고려대학교 전자전기공학과 공학석사
2008~현재 고려대학교 전자전기공학과 박사과정
관심분야 : 지능화 공간, 로봇공학, 기계학습
E-mail : proteus99@korea.ac.kr

이 호 동

2002 고려대학교 전산학과 이학사
2002 고려대학교 전기전자전파공학부 공학사
2004 고려대학교 전기공학과 공학석사
2010 고려대학교 전기공학과 공학박사
2010~현재 한국과학기술연구원 포토닉스센서시스템
관심분야 : 지능 시스템, 휴머노이드 로봇, 로봇공학
E-mail : juragic@korea.ac.kr