지능형 로봇 제어를 위한 작업 관리 구조

곽별샘[O] 이재호
서울시립대학교 인공지능 연구실
semix2@naver.com[O], jaeho@uos.ac.kr

A Task Management Architecture for Control of Intelligent Robots

Byulsaim Kwak[O] and Jaeho Lee
University of Seoul

요 약

로봇의 각 기능적 단위인 제어 컴포넌트는 그 자체로는 하나의 독립적인 기능을 수행하기 때문에 복잡한 문제를 해결하기 위해선 여러 제어 컴포넌트를 상황에 따라 제어할 수 있는 태스크 매니저가 필요하다. 상황을 판단하고 그에 따른 적절한 제어를 하기 위해서는 제어 컴포넌트의 실행 상태를 정의하고 이에 따른 정확한 규정이 필요하다. 또한 제어 컴포넌트가 동일한 메모리 공간에만 존재하는 것이 아니기에 네트워크로 분리하고 이들 사이의 통신을 정의하고 구현할 필요가 있다. 본 논문은 제어 컴포넌트의 제어 상태를 정의하고 태스크 매니저와 제어 컴포넌트를 연결할 작업 관리 구조와 이를 이용한 실험 결과를 제시한다.

1. 서론

지능형 로봇의 기능이 점점 복잡해지고 그것이 놓인 환경이 복잡해짐에 따라 통합적인 하나의 기능적 요소와 제한적인 시나리오 기반의 시스템으로는 목적을 제대로 수행하기 힘들다[3]. 로봇의 기능이 높은 시스템 사양을 요구함으로써 로봇은 두 대 이상의 컴퓨터로 구성되기도 하며 이 경우 기능적 요소는 통합된 한 요소로 구현하기 힘들다. 또한 기능적 요소 자체가 주어진 환경을 직접 파악하고 반응하기엔 부담이 크다.

본 논문에서는 환경의 변화에 대해 적절히 수행 절차를 조정하는 태스크 매니저와 그것이 로봇의 기능을 효율적으로 제어할 수 있도록 분산 제어 컴포넌트를 정의한다.

2. 전체 작업 관리 구조

로봇의 전체 작업 구조는 그림 1과 같다.

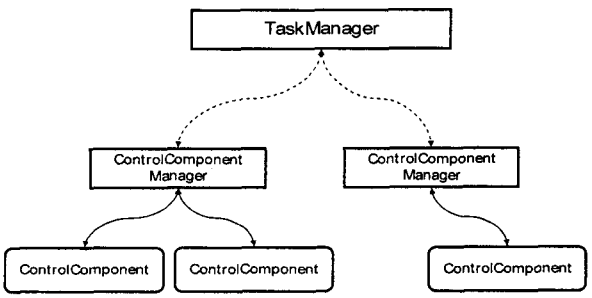

그림 1. 전체 작업 관리 구조

로봇의 세부 기능적 요소인 제어 컴포넌트는 제어 컴포넌트 관리자에 의해 관리되고 그것을 통해 태스크 매니저와 연결된다. 제어 컴포넌트 관리자는 태스크 매니저와 네트워크로 분리되어 있으며 서로 메시지를 XML로 주고 받는다.

태스크 매니저가 특정 제어 컴포넌트를 메모리에 적재할 것을 요청하면 제어 컴포넌트 매니저가 제어 컴포넌트를 메모리에 적재시킨다. 제어 컴포넌트에 대한 다른 실행 요청도 제어 컴포넌트 매니저가 받아 제어 컴포넌트에게 전달한다. 제어 컴포넌트가 태스크 매니저에게 특정 정보를 제공할 경우에도 내부적으로 제어 컴포넌트 매니저를 통한다.

태스크 매니저가 제어 컴포넌트 매니저를 통해 제어 컴포넌트를 실행하고 제어 컴포넌트 매니저가 네트워크로 분리되어있으므로 제어 컴포넌트는 태스크 매니저와 별도로 존재할 수 있다. 이 경우 태스크 매니저는 제어 컴포넌트 매니저와 제어 컴포넌트에 대한 정보가 필요하다. 이 정보는 표 1과 같은 XML 형식으로 표현된다.

표 1. 제어 컴포넌트 관리자와 제어 컴포넌트 명세

```
<Description>
    <ControlComponentManagerList>
        <ControlComponentManager>
            <Name>TestCCManager</Name>
            <IPAddress>localhost</IPAddress>
            <Port>5001</Port>
        </ControlComponentManager>
    </ControlComponentManagerList>

    <ControlComponentList>
        <ControlComponent>
```

```
            <Name>TestControlComponent1</Name>
            <Classpath>tester.TestControlComponent1</Classpath>
            <Location>TestCCManager</Location>
        </ControlComponent>
        <ControlComponent>
            <Name>TestControlComponent2</Name>
            <Classpath>tester.TestControlComponent2</Classpath>
            <Location>TestCCManager</Location>
        </ControlComponent>
    </ControlComponentList>
</Description>
```

태스크 매니저는 표 1의 명세 정보를 갖고 제어 컴포넌트를 제어하게 된다.

3. 태스크 매니저

환경의 변화를 감지하고 그것을 반영해 현재 목적을 수행하기 위해 에이전트 시스템을 사용한다. 여기서 사용된 에이전트는 JAM 에이전트 시스템[2][1]으로 BDI 기반 구조로 설계되어있다.

JAM 에이전트는 내부에 실세계 모델을 갖고 있고 그것을 근거로 상황을 판단한다. 수행해야 할 목표는 내부에 GOAL로서 표현되고 GOAL을 만족시킬 수 있는 적절한 수행 절차를 PLAN으로 갖고 있다. JAM 에이전트는 실세계 모델의 정보와 GOAL을 근거로 적절한 수행 절차를 선택하고 실행하며, 실행 중 현재의 상황이 PLAN이 서술한 제약 조건에 만족하지 못하면 실행 중인 PLAN을 중지하고 다시 상황에 맞는 적절한 PLAN을 찾아 실행한다.

태스크 매니저가 로봇의 기능을 적절히 활용해 주어진 목표를 수행하기 위해서 로봇의 기능은 그 목적 또는 기능 별로 세분화될 필요가 있다. 따라서 태스크 매니저는 세분화 된 제어 컴포넌트를 적절히 제어함으로써 주어진 목표를 완수할 수 있다.

4. 제어 컴포넌트

제어 컴포넌트는 로봇의 세부 기능 요소이다. 이것은 하나의 목적 또는 기능을 수행하며 태스크 매니저와 통신하여 상호 작용한다. 태스크 매니저로부터의 실행 요청을 받아 그것의 기능을 수행하고, 태스크 매니저에 의해 실행 상태가 변할 수 있다. 반대로 제어 컴포넌트는 임의의 시점에서 언제든지 태스크 매니저에게 정보를 제공할 수 있다.

제어 컴포넌트는 다음과 같은 추상 메소드를 가지며 개발자는 제어 컴포넌트의 기능을 구현하기 위해 이 메소드를 구현한다.

void execute(String XMLParameter) 태스크 매니저로부터 실행 요청이 오면 제어 컴포넌트는 이 메소드를 실행한다. 따라서 제어 컴포넌트의 기능은 이 메소드 안에 구현한다.

태스크 매니저가 제어 컴포넌트의 실행 상태를 제어할 수 있으므로 제어 컴포넌트의 실행 상태를 정의할 필요가 있다. (그림 2)

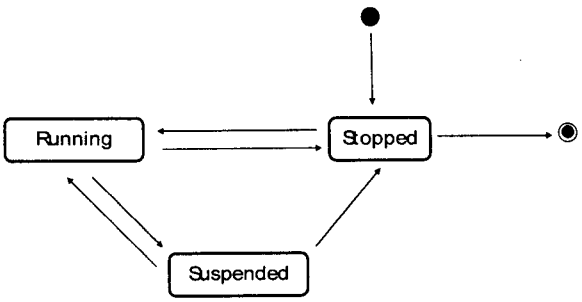

그림 2. 제어 컴포넌트의 실행 상태

제어 컴포넌트는 태스크 매니저의 요청에 의해 메모리에 적재되며 실행 요청이 오기 전까지 그것의 상태는 Stopped 이다. 태스크 매니저가 제어 컴포넌트에게 실행을 요청하면 그것의 execute() 메소드가 호출되고 상태는 Running 이 된다. 제어 컴포넌트가 기능을 수행하고 있는 도중 태스크 매니저의 요청에 의해 일시 중지가 되면 Suspended, 중지 요청에 의해 Stopped 상태가 된다.

태스크 매니저의 실행 제어 요청을 임의의 시점에서 받아들이면 프로그램 구조상 실행 상태의 안전함을 보장할 수 없다. 따라서 제어 컴포넌트의 개발자가 태스크 매니저의 요청을 수락해도 되는 안전한 지점에 그것을 명시할 수 있는 방법을 제공한다.

void tmCheckPoint() 태스크 매니저로부터의 실행 제어 요청을 받아서 제어 컴포넌트의 실행 상태를 변경한다.

이로써 개발자는 태스크 매니저로부터의 실행 제어 요청을 수락할 수 있는 안전한 지점을 명시할 수 있다. 또한 실행 상태가 변화는 과정에서 필요한 별도의 작업이 요구되는 경우 그것을 서술할 방법을 제공한다.

void onStoppedToRunning() void onRunningToStopped() void onRunningToSuspended() void onSuspendedToRunning() void onSuspendedToStopped()

각각의 onXXXToYYY() 메소드는 제어 컴포넌트의 실행 상태가 XXX에서 YYY로 변하기 직전에 호출된다. 상태 변화에 따른 절절한 수행 절차가 필요한 경우 개발자는 onXXXToYYY() 메소드를 구현하여 지정할 수 있다.

효율적인 리소스 관리를 위해 제어 컴포넌트는 항상 메모리에 상주해있을 필요가 없다. 필요한 경우 메모리에 적재하고 경우에 따라 메모리에서 제거될 수 있어야 한다. 태스크 매니저의 이러한 요청에 따라 제어 컴포넌트는 메모리에 적재, 또는 제거되며 개발자가 이 과정에서 필요한 적절한 절차를 서술할 수 있는 방법을 제공한다.

void onLoad() 태스크 매니저의 요청에 의해 제어 컴포넌트를 메모리에 적재할 때 이 메소드를 호출하고나서 적재한다. 적재 후 제어 컴포넌트의 실행 상태는 Stopped 이다. void onUnload() 태스크 매니저의 요청에 의해 제어 컴포넌트가 메모리에서 제거될 때, 실행 중인 제어 컴포넌트는 중지되고 이 메소드가 호출 된 후 메모리에서 제거된다.

제어 컴포넌트는 실행 중 태스크 매니저에게 적절한 정보를 제공할 의무가 있다. 제어 컴포넌트가 제공하는 정보는 태스크 매니저의 실세계 모델에 반영되며 판단의 근거가 될 수 있기 때문이다. 제어 컴포넌트는 실행 중 임의의 시점에서 언제든지 태스크 매니저에게 정보를 제공할 수 있으며 정보를 제공하는 방법은 다음과 같다.

void tmNotify(String XMLMessage) 태스크 매니저에게 정보를 제공한다.

지금까지 제어 컴포넌트가 기능을 수행하고 태스크 매니저로부터의 요청을 처리하며, 태스크 매니저에게 정보를 제공할 수 있는 방법을 정의하고 구현하였다. 다음으로 이것이 분산 환경에 위치하고 태스크 매니저와 적절히 연동할 수 있는 구조를 정의한다.

5. 사용 사례

위의 관리 구조를 이용해 실제 로봇의 제어를 구현해보았다. 로봇엔 네 개의 제어 컴포넌트가 사용되었다. (표 2)

표 2. 로봇의 제어 컴포넌트

제어 컴포넌트 이름	역할 및 기능
Navigation	목표 위치를 입력 받아 지정한 위치로 이동한다.
GUICommand	GUI 입력 창을 통해 사용자의 명령을 입력 받고, 입력 받은 명령을 태스크 매니저에게 전달한다.
GUISpeak	사용자에게 특정 메시지를 GUI 창을 통해 표시한다.
CheckFrontObstacle	레이저 센서를 이용해 로봇 전방의 장애물을 인식하고 인식 결과를 주기적으로 태스크 매니저에게 알린다.

사용자가 GUI 입력 창을 통해 위치를 지정하면 GUICommand 제어 컴포넌트는 태스크 매니저에게 위치 정보를 전달한다. 태스크 매니저는 이 정보를 근거로 그것을 수행할 수 있는 PLAN을 검색하여 실행한다. PLAN은 Navigation 제어 컴포넌트를 실행시켜 목표 위치로 이동할 것을 요청하고, GUISpeak 제어 컴포넌트를 실행시켜 목표 위치를 사용자에게 표시한다. CheckFrontObstacle 제어 컴포넌트는 전방의 장애물은 인식하고 주기적으로 인식 결과를 태스크 매니저에게 알린다. 전방에 장애물이 감지되면 태스크 매니저는 적절한 상황 판단을 통해 Navigation 제어 컴포넌트를 중지시키거나 우회할 수 있는 새로운 좌표를 전달해 재 실행시킨다. 동시에 GUISpeak 제어 컴포넌트를 통해 사용자에게 현재 상태를 표시한다.

6. 결과 및 향후 계획

제시한 작업 관리 구조를 이용해 로봇의 기능적 요소를 세분화하고 각각을 제어 컴포넌트화함으로써 세부적인 기능을 효과적으로 제어할 수 있었다. 또한 제어 컴포넌트를 네트워크로 분리할 수 있기 때문에 사용자의 입력을 받는 제어 컴포넌트와 사용자에게 메시지를 출력하는 제어 컴포넌트를 로봇 외부에 두는 것이 가능했다.

그러나 아직 제어 컴포넌트 명세에 대한 관리와 적절한 저장소 문제가 남아있다. 현재의 컴포넌트 매니저로는 그 역할에 한계가 있다. 하나의 독립적인 저장소를 두고 그것이 제어 컴포넌트 전체에 대한 명세 정보를 관리할 필요가 있다.

또 각 제어 컴포넌트가 서로에게 미치는 영향을 기술할 수 있는 방법이 없다. 둘 이상의 제어 컴포넌트가 동일한 자원을 공유할 때, 태스크 매니저가 그 사실을 알 수 없기 때문에 태스크 매니저는 동시에 실행할 수 있다. 따라서 각 제어 컴포넌트 사이의 관계를 서술할 방법과 그에 따른 적절한 대처 방안이 필요하다.

네트워크로 분리되는 제어 컴포넌트는 그것의 기능을 서비스 측면으로 바라볼 때 웹 서비스 형태를 응용할 수 있다. 서비스의 입출력 및 관계를 OWL[4]등의 표준 방안으로 서술하고 웹 서비스로 확장함으로써 작업 관리 구조의 기능을 확장할 수 있을 것이다.

7. 감사의 글

본 연구는 과학기술특정연구개발사업의 인간기능 생활지원 지능로봇 기술개발사업단의 지원을 받았음.

8. 참고 문헌

[1] Jaeho Lee, Edmund H. Durfee, "Structured circuit semantics for reactive plan execution systems," Proceedings of the twelfth national conference on Artificial intelligence (vol. 2), p.1232-1237, October 1994.

[2] Marcus Huber, "JAM, A BDI-theoretic Mobile Agent," Proceedings of the Third International Conference on Autonomous Agents (Agents-99), May, 1999.

[3] Michael Beetz, "Autonomous Agents and Multi-Agent Systems," Volume 4, Issue 1-2, p.25-55, March-June 2001.

[4] Mike Dean, et al, "OWL Web Ontology Language Reference," http://www.w3.org/TR/owl-ref/.