지능형 서비스 로봇을 위한 품질특성 기반의 소프트웨어 아키텍처 브로커링 방법

(A Quality-Attribute-Driven Software Architecture Brokering Mechanism for Intelligent Service Robots)

서 승 렬 † 구 형 민 † 고 인 영 ††
(Seung-Yeol Seo) (Hyung-Min Koo) (In-Young Ko)

요 약 지능형 서비스 로봇(Intelligent Service Robots)이란 스스로 주변 환경을 모니터링 하고, 문제 상황 발생시 해결 방안을 마련하여 사용자에게 적절한 서비스들을 제공해 주는 로봇을 말한다. 그러나 로봇이 접할 수 있는 다양한 환경과 상황, 문제들을 미리 예측하여 필요한 모든 기능들을 내부에 포함시키기는 어렵다. 로봇 내부에는 환경에 맞는 필요한 기능들만 가지도록 하고, 필요에 따라 새로운 기능들을 증식 시켜 줄 수 있는 자가 성장 소프트웨어(Self-Growing Software)를 개발함으로써 이러한 문제점을 해결할 수 있다. 본 논문은 자가성장 소프트웨어가 외부로부터 필요 기능 수행을 위한 서브아키텍처(컴포넌트의 조합 패턴)의 획득을 브로커를 통해 수행 할 때, 기능적인 측면뿐 아니라 품질특성과 환경을 고려하여 적절한 서브아키텍처를 선택하는 방법에 관한 것이다. 사용자의 품질 요구를 고려하여 기능수행에 가장 적합한 서브아키텍처를 추론하기 위해 사용품질요구(Quality-Attributes In Use) 온톨로지를 구축하였고, 컴포넌트 조합 선택을 위해 추상화된 레벨 선택 및 구체적 레벨 선택의 2단계 선택 기법을 개발하였다. 추상화된 레벨에서는 상위단계 목표를 기반으로 서브아키텍처의 특성들을 추론하고, 구체적 레벨에서는 컴포넌트 자체의 사용품질요구에 부합하는 특성들을 기반으로 실제 서브아키텍처를 검색한다. 이러한 방법을 통해 사용자의 목표나 상황에 따라 달라지는 품질 요구사항을 반영하여 서비스를 제공할 수 있다. 또한, 로봇의 기능 증식 시에 사용자의 목표에 좀 더 부합하는 기능 선택이 가능하고, 사용자의 요구와 소프트웨어 기능 표현 및 해석상의 차이를 극복할 수 있다.

키워드 : 자가성장 소프트웨어, 지능형 서비스 로봇, 사용자 품질요구, 소프트웨어 품질특성

Abstract An intelligent service robot is a robot that monitors its surroundings, and then provides a service to meet a user's goal. It is normally impossible for a robot to anticipate all the needs of its user and various situations in the surroundings ahead, and to prepare for all the necessary functions to cope with them. Therefore, it is required to support the self-growing capability by which robots can extend their functionality based on users' needs and external conditions. In this paper, as an enabler of the self-growing capability, we propose a method that allows a robot to select a component-composition pattern represented in an architectural form(called a sub-architecture), and to extend its functionality by obtaining a set of software components that are prescribed in the pattern. Sub-architecture is selected and instantiated not only based on the functionality required but also based on quality requirements of a user and the surrounding environment. To provide this method, we constructed a quality-attributes-in-use ontology and developed a brokering mechanism that matches quality requirements of users and surroundings against quality attributes of sub-architectures. The ontology provides the common vocabularies to represent quality requirements and attributes, and

· 이 연구(논문)은 산업자원부 지원으로 수행하는 21 세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.
· 이 논문은 2008 한국소프트웨어공학학술대회에서 '로봇소프트웨어를 위한 품질특성 기반의 아키텍처 브로커링 방법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한국정보통신대학교 공학부
syseo@icu.ac.kr
hyungminkoo@icu.ac.kr
†† 정 회 원 : 한국정보통신대학교 공학부 교수
iko@icu.ac.kr

논문접수 : 2008년 4월 17일
심사완료 : 2008년 12월 1일

정보과학회논문지: 소프트웨어 및 응용 제36권 제1호(2009.1)

enables the semantically-based reasoning in matching and instantiating appropriate sub-architectures in supporting services to users. This ontology-based approach contributes to provide a great flexibility in extending robot functionality based on available software components, and to narrow the gap between users' quality requirements and the quality of the actual services provided by a robot.

Key words : Self-growing software, Intelligent service robots, Quality-attributes-in-use, Software quality attributes

1. 서 론

현재의 로봇을 동작시키는 소프트웨어 기술의 개발은 대부분 로봇이 접할 수 있는 환경이나 문제를 미리 예측하고 그에 필요한 기능을 로봇에 추가하는 방식으로 이루어지고 있다[1-3]. 하지만 로봇이 접하게 될 다양한 환경, 상황 그리고 문제들을 미리 예측하여 적합한 소프트웨어 기능을 모두 포함하는 것은 제한된 자원을 가지고 작동하는 로봇에서는 불가능 하다. 이러한 문제점은 외부로부터 소프트웨어 구성(아키텍처, 컴포넌트, 서비스 등)을 실행시간(Run Time)에 획득하여 로봇의 기능을 향상시켜 나가는 자가성장 소프트웨어를 실현함으로써 해결될 수 있다[4].

이와 같은 지능형 서비스 로봇을 위한 자가성장 소프트웨어를 개발하기 위한 노력의 일환 중 아키텍처 기반의 자가성장 소프트웨어가 있으며 이 자가성장 소프트웨어의 목적은 고수준의 소프트웨어 재구성, 재배치 능력을 갖추어 예기치 못한 상황 및 문제를 해결하는 것이 가능하도록 하는데 있다. 이러한 재구성 능력을 갖추기 위하여, 컴포넌트 기반의 소프트웨어 기반 개발 방법론을 지향하고 있는데, 그 이유는 컴포넌트를 기반으로 구성된 소프트웨어는 구성정보의 획득, 저장, 관리가 용이하기 때문이다[5].

본 논문에서는 기존에 개발된 아키텍처 기반의 자가성장 소프트웨어에 대해 간략히 설명하고[6], 컴포넌트의 조합 및 컴포넌트의 검색 및 획득의 역할을 수행하는 브로커가 소프트웨어의 '기능' 뿐만 아니라 '사용자의 품질요구'를 함께 고려해야 하는 필요성을 지적한다. 그리고 기능뿐 아니라 품질특성까지 고려하여 아키텍처를 획득하도록 하기 위한 방법에 대해 설명한다. 사용자의 품질요구를 고려한 컴포넌트의 조합을 선택하기 위하여 추상화된 레벨 선택 및 구체적 레벨 선택의 2단계 선택 기법을 개발하였다. 추상화된 레벨에서는 상위단계 목표를 기반으로 컴포넌트들의 특성을 추론하고, 구체적 레벨에서는 사용품질요구에 부합하는 컴포넌트 조합에 적합한 컴포넌트들을 검색한다. 사용자에 품질요구를 표현하고 추론하기 위하여 사용품질요구(Quality-Attribute-In-Use) 온톨로지를 제작하였다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 2절에서는 현재 연구중인 아키텍처 기반의 자가 성장 소프트웨어에 대하여 간략히 기술하고 품질특성 기반의 아키텍처 선택 방법을 위한 요구사항을 도출한다. 이어지는 3절에서는 로봇 소프트웨어에서의 품질특성에 기반한 아키텍처 브로커에 대해 소개하고 브로커의 작동 절차에 대해 상세히 설명한다. 4절에서는 앞서 기술한 요구사항을 만족시키기 위한 핵심 기술요소들에 대해 설명하고, 5절에서는 기존 연구와의 비교 분석 결과를 설명한다. 그리고 마지막 6절에서는 결론과 향후 연구에 대해 설명한다.

2. 기능 중심의 아키텍처 및 컴포넌트 선택 방법

자가 성장 소프트웨어는 "동적인 상황 판단에 의거해 스스로 자신의 기능과 컴포넌트의 구성 능력을 성장시키는 소프트웨어"라고 정의될 수 있다[7]. 이처럼 자가성장 소프트웨어는 자신이 가진 기능만으로 문제를 해결할 수 없을 때 외부로부터 소프트웨어 구성(아키텍처, 컴포넌트, 서비스 등)을 추가하여 성장을 할 수 있도록 설계된 소프트웨어를 말한다.

그림 1은 아키텍처 기반의 자가성장 소프트웨어 프레임워크인 SHAGE(Self-Healing, Adaptive, and Growing SoftwarE)의 구조이다[7]. SHAGE에서는 새로운 로봇 기능이 필요할 경우 이에 맞는 컴포넌트들의 조합 패턴인 서브아키텍처를 실행시간에 획득하고 이를 기반으로 필요한 컴포넌트를 찾아 조합하여 주어진 기능이 제공될 수 있도록 한다. 특히 여러 개 서브아키텍처가 연결되어 사용자가 원하는 태스크에 필요한 다양한 기

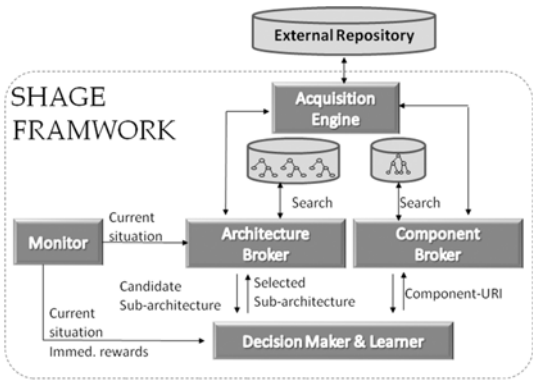


그림 1 자가성장 소프트웨어의 구조

능들을 제공할 수 있도록 한다. SHAGE에 있는 상황 모니터(Monitor)는 다양한 로봇 센서를 통해서 로봇이 처한 상황(사용자의 품질 요구, 행위, 환경 등)을 파악하는 역할을 수행한다. 아키텍처 브로커(Architecture Broker)는 상황모니터를 통해 파악된 정보를 기반으로 적절한 서브아키텍처들을 선택하고, 의사결정 및 학습기(Decision Maker & Learner)는 과거에 학습된 내용을 바탕으로 소프트웨어 내·외부적 속성 측면에서 기능 수행에 가장 적합한 서브아키텍처를 선택하게 된다.

사용자의 상위 단계의 목표(Goal)는 태스크(Task)를 수행함으로써 만족될 수 있고 각 태스크는 태스크를 수행하기 위한 행위들로 세분화되어 SHAGE의 입력으로 사용된다. 그림 2는 태스크를 수행하기 위한 세부 행위들과 각 행위를 지원하는 서브아키텍처, 그리고 실제 컴포넌트와의 연결 관계를 나타낸다. 태스크 내 각 행위는 서브아키텍처가 선택되어 지원되고 각 서브아키텍처에 필요한 컴포넌트가 저장소에서 찾아 조합되어 실행 가능한 형태로 준비된다.

사용품질요구를 고려한 서브 아키텍처의 선택은 태스크뿐만 아니라 사용품질요구, 환경 등을 고려함으로써 사용자를 만족시키지 못하는 서브아키텍처를 선택에서 제외한다. 이로써 서비스를 제공 받는 사용자의 만족도를 보다 높일 수 있는 서브아키텍처의 선택을 할 수 있다. 그림 3은 특정한 행위, 'Move to the Room'에 필요한 서브아키텍처가 사용품질요구와 환경을 고려하여 어떻게 다르게 선택되는지를 보여 준다. 즉, 행위만 고려하여 선택된 서브아키텍처, 'Vision-Based Navigation', 'Laser-Based Navigation', 'Infrared-Based Navigation'은 사용 품질요구, 'Fast'를 추가로 고려함에 따라 그 선택이 변화 될 수 있다. 예를 들면, 'Vision-Based Navigation'의 경우 영상인식시간이 많이 걸리기 때문에 'Detailed but Relatively Slow'라는 특성을 갖게 되고, 'Fast'라는 품질 요구사항이 주어질 경우 이 서브아키텍처는 선택에서 제외되고 대체 가능한 'Laser-Based Navigation' 등이 선택된다. 뿐만 아니라, 태스크가 수행되고 있는 환경은 사용품질요구에 영향을 줄 수 있다. 위 예제에서 좁은 통달거리로 인해 사람과의 충돌을 유

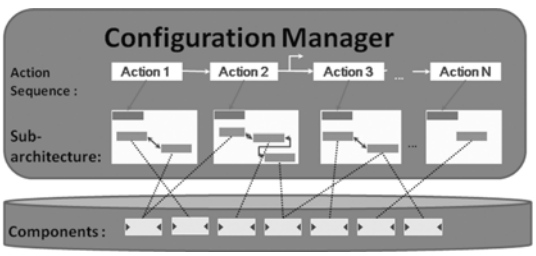
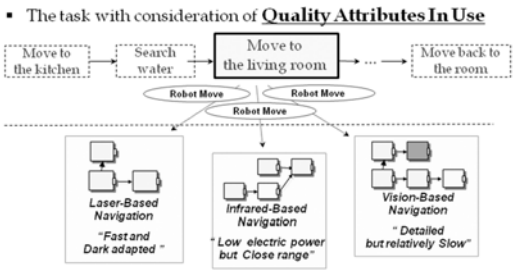


그림 2 서브아키텍처와 컴포넌트 연결



Factors	Action	Architectures Available
Action	Move to the Room	• Vision-Based Navigation • Laser-Based Navigation • Infrared-Based Navigation
Action with QAs In Use	Move fast to the room	• Laser-Based Navigation • Infrared-Based Navigation
Action with QAs In Use And Environment	Move fast to the room in the dark	• Laser-Based Navigation

그림 3 사용자 요구와 아키텍처 선택과의 관계

발할 가능성이 있어 어두운 환경, 'In The Dark'에 적합하지 않은 서브아키텍처, 'Infrared-Based Navigation'은 선택에서 제외된다.

이와 같이 서브아키텍처의 선택이 사용품질요구와 환경에 따라 변할 수 있도록 함으로써 로봇 소프트웨어 기능이 사용자의 요구를 보다 정확히 만족시킬 수 있게 된다. 사용자 요구를 기반으로 서브아키텍처를 선택하기 위해서는 다음과 같은 요구사항을 충족해야 한다.

• 사용자의 상위단계 목표를 만족

서브아키텍처의 획득 및 선택 시에는 사용자의 상위 단계 목표(High-level Goal)를 직접적으로 만족시킬 수 있을 뿐 아니라, 소프트웨어 구성 각각에 대한 상세한 기술적 고려를 최소화할 수 있어야 한다.

• 주변 환경에 대한 고려

환경에 따라 요구하는 소프트웨어 구성의 품질특성이 다를 수 있으므로 환경에 대한 고려는 반드시 수반되어야 한다. 따라서 품질 특성 기반의 아키텍처 브로커링 시 수행 환경과 관련된 주요한 특성들을 고려하여 아키텍처를 선택할 수 있어야 한다.

• 사용자의 요구와 소프트웨어 기능 표현 및 해석상의 차이를 극복

사용자가 원하는 목표 및 요구사항을 표현하는 방법과 로봇 소프트웨어의 구성을 표현하는 방법에는 차이가 있다. 따라서 이러한 차이를 극복하고 사용자의 요구에 부합하는 소프트웨어 구성을 이룰 수 있도록 하기 위한 공통적인 표현 모델과 추론 방법이 필요하다.

• 기능에 대한 확장성

로봇에서 사용될 수 있는 컴포넌트와 그 조합패턴은 수시로 변화되거나 추가될 수 있다. 따라서 태스크, 사용품질요구 그리고 환경에 기반한 소프트웨어 구성의 선택 방법은 새로운 기능을 갖는 소프트웨어 구성의 추

가나 기존의 기능을 향상시키기 위한 변경이 이루어졌을 때 관련 모델 및 추론 방법에 대한 변경 없이 계속 사용될 수 있어야 한다.

- 자가 성장 로봇 소프트웨어를 위한 사용자 품질 요구에 기반의 아키텍처 브로커링

본 절에서는 아키텍처 브로커가 아키텍처를 구성하기 위한 동작 과정과 브로커의 구성 요소들에 대해 기술한다.

그림 4는 아키텍처 브로커링 과정을 도식적으로 보여준다. 앞서 언급하였듯이, SHAGE에서 서브아키텍처의 선택은 크게 두 단계를 거친다. 첫 번째 단계에서는 사용자의 기능요구와 품질요구를 만족하며 환경에 적합한 서브아키텍처가 갖추어야 할 특성을 아키텍처 브로커를 통해 선택하게 된다. 아키텍처의 적합한 특성추론에 대한 상세한 과정은 3.3절에서 기술한다. 선택된 특성은 과거에 학습된 결과를 토대로 의사결정 및 학습기(Decision Maker & Learner)를 통해 최종선택 하게 된다[7].

사용자의 목표를 달성할 수 있는 태스크는 행위들의 조합으로 이루어져 있고 각 행위는 서브아키텍처를 통해 실행된다. 그림 5는 서브아키텍처가 조합되어 전체 아키텍처를 구성하는 모습을 보여준다. 서비스 로봇의 소프트웨어 아키텍처는 서브아키텍처들이 합병 점(Merging point)을 기준으로 결합되어 구성된다[7].

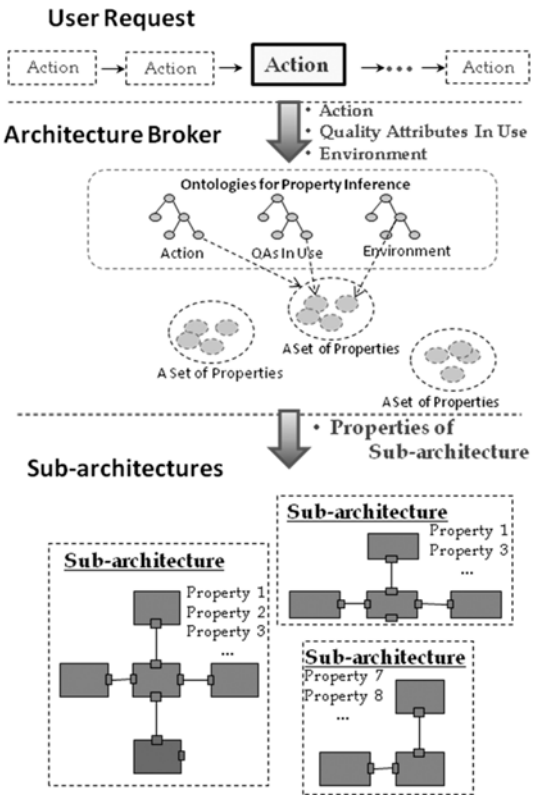


그림 4 아키텍처 브로커링 과정

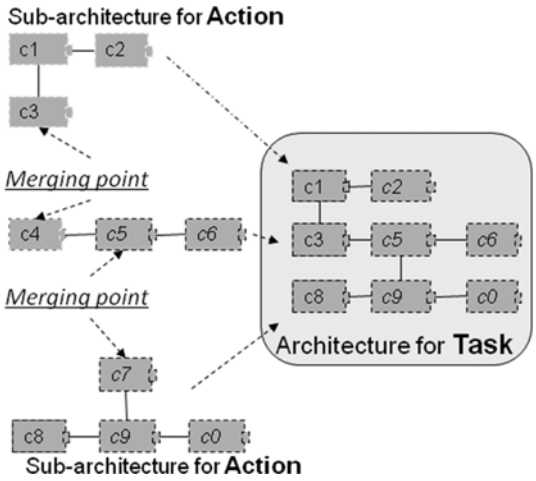


그림 5 서브아키텍처와 전체아키텍처와의 관계

3.1 태스크 및 행위의 정의

본 논문에서 태스크는 사용자의 목표를 달성할 수 있는 일의 단위라고 정의 한다. 독립적으로 달성될 수 있는 일의 단위란, 사용자가 임의로 표현한 요구를 3.2.1에서 설명될 행위들의 유한 집합으로 구성될 수 있는 조합을 말한다. 그림 6은 하나의 태스크와 태스크로부터 나누어진 행위에 대한 예제인데, 그림에서 보듯이 태스크는 사용자의 목표를 달성하기 위한 단위로 'Clean the living room'과 같이 구성될 수 있다. 그리고 태스크를 수행하기 위한 행위들은 'Move to the living room', 'Search Wastes', 'Mop the floor'로 세분화 될 수 있다. 세분화된 행위들은 사용자 품질요구, 'Fast'에 대한 적용을 받아 품질요구가 반영된 형태 'Move fast to the living room', 'Search big wastes', 'Collect big wastes'로 소프트웨어 기능이 구성되어 수행되게 된다.

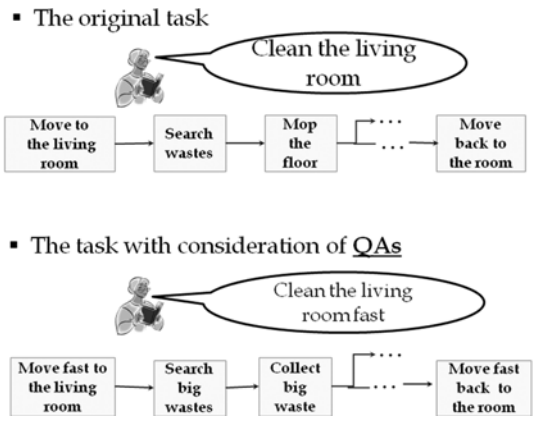


그림 6 품질요구를 고려한 태스크의 변화

3.2 품질 특성 기반 브로커링을 위한 온톨로지

아키텍처 브로커에서는 사용자 요구사항과 그들 간의 관계를 표현하고 고려하기 위하여 온톨로지 기반의 모델링 기법과 추론 방법을 사용한다. 전체 온톨로지는 행위, 사용품질요구, 환경 등 세 부분으로 구성되어 있다. 본 절에서는 각 세부 온톨로지에 대하여 설명한다.

3.2.1 행위(Action) 온톨로지

행위 온톨로지의 구성은 인간의 9가지 기본행위를 기반으로 하였으며, 표 1은 각 대 분류 요소와 이에 대한 설명이다[8]. 위에서 언급하였듯이 태스크는 행위들의 조합으로 구성된다. 행위 온톨로지는 각 행위를 수행하는 소프트웨어의 기능상의 분류에 해당하는 아키텍처의 특성 집합(A Set of Properties)을 추론하기 위해 이용된다. 행위 온톨로지를 통해 추론된 특성집합은 품질 온톨로지와 환경 온톨로지를 통해 각 서브아키텍처가 갖추어야 할 특정한 특성으로 정리되어 이용된다. 이처럼 특성집합의 선택과 특성선택 단계를 나누는 이유와 예제는 3.3절에서 설명될 것이다. 그림 7은 행위 온톨로지의 상위 계층 요소들을 보여준다.

표 1 행위의 분류와 설명

Action List	Explanation
ATRANS	Transfer a relationship e.g. give, have, etc.
PROPEL	Apply physical force to an object e.g. push, spray, etc.
PTRANS	Transfer of physical location of an object e.g. move, is in, etc.
INGEST	Ingest an object by an owner e.g. eat, drink, etc.
MBUILD	Mentally make new information e.g. decide, want, want to know, become, is, find, etc.
...	...

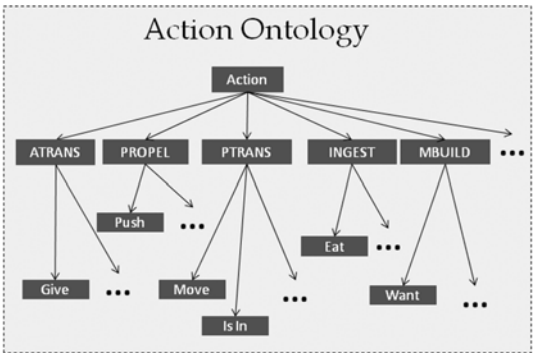


그림 7 행위 온톨로지

3.2.2 사용품질요구 온톨로지

사용자 품질요구의 대 분류는 사용품질 표준인 ISO/IEC 9126-4 Quality-Attributes-In-Use(QAs-In-Use)를 바탕으로 구성되었으며 그림 8은 사용자 품질요구 온톨로지의 주요 상위 계층 요소들을 보여준다. 이 온톨로지는 태스크 행위를 지원하는 서브아키텍처를 선택할 때 기능뿐 아니라 품질 특성을 고려할 수 있도록 하기 위한 아키텍처 특성의 표현과 추론을 위해 사용된다.

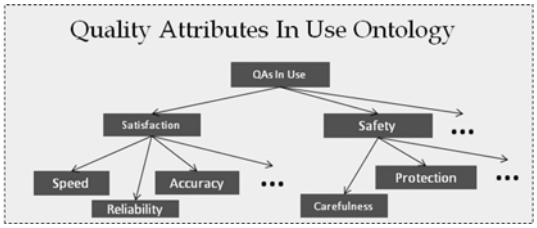


그림 8 사용품질요구 온톨로지

3.2.3 환경 온톨로지

태스크를 수행해줄 수 있는 서브아키텍처를 선택함에 있어서 사용품질요구 이외에 서브아키텍처 선택에 영향을 줄 수 있는 다양한 환경(밝은 빛, 소음, 많은 장애물이 있는 집)에 대한 모델링과 이를 기반으로 한 추론이 필요하다. 그림 9는 환경 온톨로지의 구성 예제이다. 이 환경 온톨로지는 로봇이 배치될 환경에 맞게 세부적으로 구성된다.

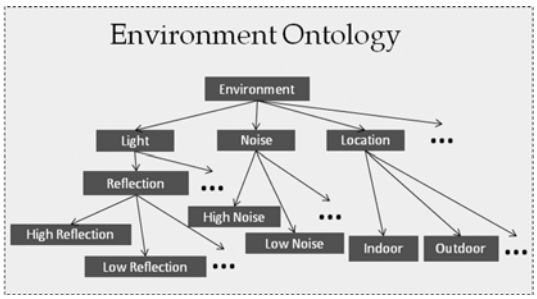


그림 9 환경 온톨로지

3.3 품질 특성 기반의 아키텍처 브로커링 절차

그림 10은 아키텍처 브로커의 수행 절차를 도식적으로 보여준다. 아키텍처 브로커는 태스크를 구성하는 행위, 사용품질요구 그리고 환경 정보를 입력으로 받는다. 아키텍처 브로커는 위의 세가지 정보를 토대로 다음의 추론작업을 수행한다.

i. 행위 온톨로지 기반의 추론: 행위 온톨로지는 행위를 수행하는 기능상의 분류에 사용되는 특성 집합을 표

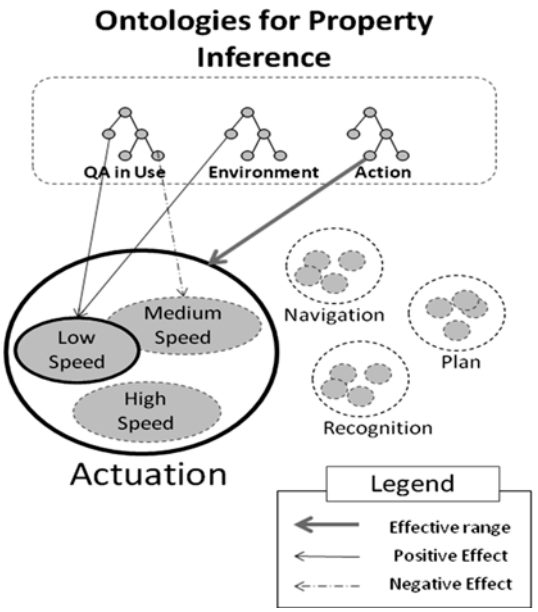


그림 10 아키텍처 브로커링의 예

현하고 추론하는데 사용된다. 특성 집합이란 한가지 행위를 수행하는 서브아키텍처들이 가질 수 있는 다양한 특성들을 분류해 놓은 것이다. 이처럼 행위 온톨로지를 통해 특성 집합을 추론하는 이유는, 필요로 하는 행위와 연관된 품질 요구사항을 서브아키텍처의 특성들과 비교하여 효과적으로 적합한 서브아키텍처를 선택하기 위해서 이다. 예를 들면 'Move'라는 행위를 수행 해 줄 수 있는 'Navigation'기능을 지원하는 서브아키텍처들을 선택 할 경우 그 특성 집합은 'High Speed', 'Medium Speed', 'Low Speed'등을 포함하고 이러한 특성과 사용자 또는 환경으로부터 전달되는 품질 요구사항이 비교되게 된다.

ii. 사용품질요구와 환경 온톨로지 기반의 추론: 로봇의 공통된 특성 집합이 선택된 후에는 사용품질요구 온톨로지와 환경 온톨로지를 통해 표현된 요구사항과 비교되어 그 적합성을 추론하게 된다. 즉, 두 온톨로지로 표현된 요구사항은 서브아키텍처의 특성을 긍정효과(Positive Effect)나 부정효과(Negative Effect) 두 가지 상반된 타입 고려한다. 긍정효과는 사용자의 상위단계 목표를 수행하는데 긍정적인 영향을 주는 특성이고, 부정효과는 그 반대를 나타내는 특성으로서, 최종적으로 긍정효과 대비 부정효과의 비율이 가장 높은 서브아키텍처가 선택된다.

iii. 학습기반 서브아키텍처 선택: 사용자의 상위단계 목표를 수행하는데 적합한 서브아키텍처는 여러 개 선택될 수 있다. 앞서 언급하였듯이 선택된 서브아키텍처들은 로봇 소프트웨어 내외부적인 속성을 고려하여 최종적으로 가장 적합한 한 개의 서브아키텍처만 선택되게 되는데, 그 최종 선택은, 의사결정&학습기를 통해 과거에 학습된 내용(선택되었던 서브아키텍처와 선택될 당시의 환경, 소프트웨어 내·외부 정보 등)을 토대로 이루어진다[7].

앞서 설명한 추론 과정에 대한 예제로서, 그림 10은 '방을 빠르게 청소하라(Clean the room Fast)'라는 태스크를 수행하기 위한 행위 중 '빠르게 거실로 이동(Move fast to the living room)' 행위에 해당하는 서브아키텍처가 선택된 상황을 보여 준다. 집의 환경은 빛의 반사가 심한 바닥 및 벽지를 사용하였다고 가정하였다. 가장 먼저 행위 온톨로지를 통해 로봇에게 적용될 공통적인 특성집합을 선택한다. 그림 10의 예제에서는 행위 온톨로지에 3.3절에서 정의된 추론과정을 통해 'Move' 행위에 해당하는 'Navigation' 특성 집합이 선택된 모습을 보여준다.

행위 온톨로지를 통해 공통된 특성집합이 선택된 후, 사용자 품질요구와 환경 온톨로지를 통해 표현된 요구사항과의 비교가 이루어진다. 앞에서 설명한 바와 같이 긍정효과와 부정효과에 대한 판단을 통해 서브아키텍처를 선택하게 된다. 그림 10에서는 또한 그림 3의 사용자 품질요구와 아키텍처의 관계에서 설명하였던 사용품질요구를 기반으로 소프트웨어 구성을 선택하는 과정을

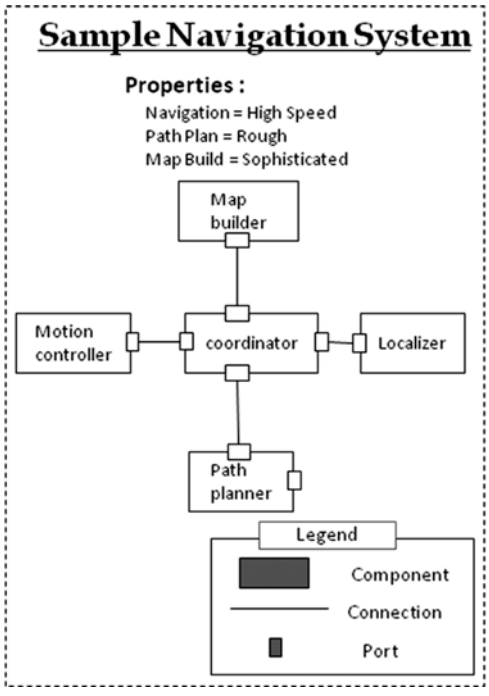


그림 11 서브아키텍처 구성 예제

보여준다. 그림 10에서 'Fast'라는 사용품질요구에 의해 'High Speed Navigation' 특성은 긍정효과를 갖고 되고 주변이 반사가 심한 재질로 이루어져 있다고 가정하였으므로 빛을 통한 영상인식 등이 갖는 'Low Speed Navigation' 특성은 부정효과를 갖게 된다. 이러한 과정을 거쳐 'High Speed Navigation' 특성을 지닌 서브아키텍처들만 남게 되고 이 서브아키텍처들은 의사결정& 학습기에 의해 최종 선택 된다.

3.4 품질 특성 기반의 아키텍처 명세

서브아키텍처는 컴포넌트와 컴포넌트들 사이의 연결정보, 각 컴포넌트가 갖는 포트의 메타정보, 그리고 품질 특성에 대한 명세를 포함한다. 또한 서브아키텍처에서 요구되는 자원 특성(Memory Requirement, CPU Requirement, Average Response Time등)에 대한 명세를 포함한다.

그림 11은 Navigation을 수행하는 서브아키텍처에 대한 예제로서, 그림에서 보여지듯이 5개의 컴포넌트로 구성되어 있고, 그 속성들로는 'High Speed Navigation', 'Rough Path Plan', 'Sophisticated Map Build' 등이 있다. 그림 12는 앞서 언급한 Navigation 서브아키텍처를 아키텍처 명세 언어 중 하나인 ACME를 통해 표현한 것이다[9]. 그림 아래쪽에 1번으로 표시된 부분은

그림 12 서브아키텍처 명세 예제

앞서 기술한 품질 특성을 기반으로 서브아키텍처를 선택하는데 사용된다. 2번으로 표시된 부분은 소프트웨어 구성의 내외부적 속성에 해당되는 부분으로 의사결정& 학습기가 최종선택을 할 때 사용된다.

4. 평 가

본 절에서는 앞서 기술한 품질 특성에 기반하여 로봇 소프트웨어를 구성하기 위한 브로커링의 요구사항을 만족시키기 위한 핵심 기술 요소들에 대하여 설명한다.

• 사용자의 상위단계 목표를 만족

본 논문에서는 태스크를 수행하기 위하여 세분화된 행위들을 온톨로지를 이용하여 모델링함으로써 사용자의 상위단계 목표에 기반으로 하여 서브아키텍처가 가져야 할 특성들을 선택할 수 있도록 하였다. 실제 서브아키텍처의 선택은 행위 온톨로지를 통해 추론된 특성들을 대상으로, 사용품질요구와 환경을 추가적으로 고려하여 이루어진다. 이처럼 소프트웨어 구성 각각에 대한 상세한 기술적인 고려가 아닌, 소프트웨어 구성의 품질 특성을 고려를 통해 서브아키텍처 선택의 기준으로 사용하였다. 따라서 본 브로커링 방법은 사용자의 상위단계 목표 만족을 향상 시킬 수 있고 소프트웨어 구성 각각에 대한 기술적인 고려를 최소화 할 수 있다.

• 주변 환경에 대한 고려

환경을 온톨로지를 기반으로 모델링 하여 태스크를 지원하기 위한 서브아키텍처를 선택할 때 사용품질요구 이외에, 환경까지 고려된 서브아키텍처가 선택될 수 있도록 하였다. 따라서 본 브로커링 방법은 주변 환경을 고려한 서브아키텍처를 선택할 수 있다.

• 사용자의 요구와 소프트웨어 기능 표현 및 해석상의 차이를 극복

행위, 사용품질요구, 환경 온톨로지를 기반으로 서브아키텍처의 특성을 추론하고, 추론된 특성은 서브아키텍처에 명세 되어 있는 품질특성들과 비교된다. 이를 통해 사용자의 요구를 보다 잘 만족시켜줄 수 있는 소프트웨어 구성을 선택할 수 있도록 하였다. 따라서 본 브로커에서는 사용자 요구와 소프트웨어 구성이 가진 서로 다른 표현 및 해석상의 차이를 극복한 브로커링이 가능하다.

• 기능에 대한 확장성

행위, 사용품질요구 그리고 환경 정보 등이 서브아키텍처에 직접 명세 되지 않고 이러한 요구사항과 서브아키텍처의 특성 정보가 필요 시 비교되도록 함으로서 모델의 유연성과 확장성을 높였다. 따라서 아키텍처 정보가 수정이나 추가 되더라도 행위, 사용품질요구 그리고 환경 온톨로지에 대한 수정 없이 아키텍처를 선택할 수 있다.

5. 관련연구

사용자의 품질 요구를 소프트웨어의 품질 특성 결정에 반영하기 위한 몇몇 연구들이 수행되었거나 진행 중에 있다. 서비스기반의 아키텍처에서 다수의 서비스를 선택 및 요청할 때, 다루어져야 할 소프트웨어의 품질 특성을 결정하고 서로 다른 서비스들의 품질특성 간의 절충(Trade-off)을 하기 위한 방법에 대해 조사한 연구가 있었다[1]. 하지만 변화하는 사용자의 품질요구에 대한 고려보다는 미리 정해진 사용자의 품질요구나 소프트웨어 내외부의 품질특성을 그 비교 대상으로 하였으며 이 방법으로는 동적으로 소프트웨어를 구성할 때, 미리 정해진 기준들 이외에는 소프트웨어 구성에 사용하지 못하기 때문에 동적으로 소프트웨어를 재구성해야 하는 자가성장 소프트웨어에서는 적용이 힘들다.

또한, 소프트웨어의 품질특성과 사용자의 품질 요구의 자동화된 관계 추적을 통해, 복잡한 사용자 요구사항을 기반으로 소프트웨어 구성을 선택하고 조합하는 연구가 있었다[2]. 하지만 사용자의 미리 정의된 사용품질요구사항과 소프트웨어 품질 특성 사이의 관계를 추적하는데 연구의 초점이 맞추어져 있어서, 동적으로 변화하는 사용자의 요구를 반영하기는 어렵다.

마지막으로, 서비스 기반의 아키텍처에서 미리 명세된 사용자의 품질요구와 서비스를 실행 순서를 기반으로 실행시간에 연결하여 조합하는 방법에 대한 연구가 있었다[3]. 하지만 이 연구에서는 미리 정의된 정보(소프트웨어가 기능적으로 주고받는 입출력 데이타와 함수의 호출 관계)를 기반으로 하여, 입출력 관계 및 호출관계가 미리 정의되지 않은 아키텍처 기반의 자가성장 소프트웨어에서는 적용이 불가능하다.

지능형 서비스 로봇을 위한 자가성장 소프트웨어에서는 실행시간에 사용자의 품질 특성을 기반으로 외부로부터 소프트웨어 구성을 획득해야만 한다. 하지만 위의 연구들은 대부분 사용자의 품질요구와 소프트웨어 내외부의 품질 특성을 개발 단계에서 개발자나 도메인 전문가가 미리 정해 놓고 한정된 소프트웨어 컴포넌트 또는 서비스를 선택 및 조합하는데 초점을 두고 있다.

6. 결론 및 향후 연구

지금까지 아키텍처 기반의 자가성장 소프트웨어에서 사용자의 품질 요구사항을 기반으로 서브아키텍처를 선택하고 획득하는 방법에 대하여 설명하였다. 본 논문에서는 기능만을 중심으로 서브아키텍처를 획득하는 기존의 아키텍처 기반의 자가성장 소프트웨어의 한계를 극복하기 위해, 기능뿐 아니라 사용품질요구와 환경을 고려하여 외부로부터 소프트웨어 구성 패턴과 관련 컴포넌트를 획득하는 방법을 제시하였다. 이를 위해 사용품질요구 온톨로지를 구축하였고, 아키텍처 선택을 추상화 레벨 선택과 구체적 레벨 선택, 2단계 선택 기법을 개발하여 저장소에 소프트웨어 구성이 추가되었을 때 유연히 대처할 수 있도록 하였다. 추상화된 레벨에서는 사용품질요구를 기반으로 컴포넌트의 조합패턴을 검색하고, 구체적 레벨에서는 컴포넌트 자체의 내외부적 특성을 기반으로 패턴에 맞는 컴포넌트들을 검색한다.

본 논문에서는 사용자 및 환경의 요구 사항을 표현하기 위한 온톨로지 기반 모델을 설명하였고 이러한 요구사항이 소프트웨어 구성의 특성과 비교되어 기능 및 품질 요구사항에 맞는 소프트웨어 구성을 가능하게 하는 방법을 제시하였다. 이렇게 사용자 목표나 상황에 따라 다를 수 있는 품질 요구사항을 반영한 소프트웨어 구성을 선택함으로써 사용자의 서비스 만족도를 향상 시킬 수 있게 하며, 로봇 기능 증식에 있어 사용자의 목표에 좀 더 부합하는 기능의 선택이 가능하게 된다.

현재 지능형 서비스 로봇 도메인에서의 보다 일반화된 사용품질요구 표현을 위한 온톨로지 기반의 모델을 개발 중에 있고, 기능 기반의 브로커 프로토타입을 확장 중에 있다. 또한 실제 로봇 플랫폼에 적용, 실험하면서 본 접근법에 대한 검증 및 증명 작업을 진행 중에 있다.

참 고 문 헌

[1] Liam O'Brien, Paulo Merson, Len Bass, "Quality Attributes for Service-Oriented Architectures," Proceedings of the International Workshop on Systems Development in SOA Environments, 2007.

[2] Barry Boehm, Hoh In, "Identifying Quality-Requirement Conflicts," Proceedings of ICRE 96, 1996.

[3] Quoc Bao Vo, Lin Padghan, "A Component-Based Approach to Automated Web Service Composition," Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, 2006.

[4] Hyung-Min Koo and In-Young Ko, "A Repository Framework for Self-Growing Robot Software," Proceedings of 12th Asia-Pacific Software Engineering Conference(APSEC'2005), 12, 2005.

[5] A. Oreback, "Components in intelligent robotics," Technical report, Royal Institute of Technology, Stockholm, Sweden, 1999.

[6] Yu-Sik Park, In-Young Ko, and Soo-Yong Park, "A Task-based Approach to Generate Optimal Software-Architecture for Intelligent Service Robots," Robot & Human Interactive Communication, 2007.

[7] Dong-Sun Kim, Soo-Yong Park, Young-kyun Jin, Hyeong-Soo Chang, Yu-Sik Park, In-Young Ko, Kwan-Woo Lee, Jun-Hee Lee, Yeon-Chool Park, Suk-Han Lee, "SHAGE: A Framework for Self-managed Robot Software, Self-adaptation and self-

managing systems," 2006.
[8] RC Schank, "Conceptual Dependency: A Theory of Natural Language Understanding," Cognitive Psychology, 1972.
[9] ABLE group at Carnegie Mellon University, http://www.cs.cmu.edu/~acme/, ACME web-site, 2008.

서 승 렬
2007년 한양대학교 컴퓨터공학과(학사) 2007년~현재 한국정보통신대학교 컴퓨터 공학과 석사과정. 관심분야는 컴포넌트기반 소프트웨어 개발, 소프트웨어 신뢰성, 소프트웨어 품질특성

구 형 민
2004년 금오공과대학교 컴퓨터공학과(학사). 2007년 한국정보통신대학교 컴퓨터공학과(석사). 2007년~현재 한국정보통신대학교 컴퓨터공학과 박사과정. 관심분야는 자가적응 소프트웨어, 그룹인지 소프트웨어, 사용자중심 소프트웨어

고 인 영
1990년 서강대학교 전산학과(학사). 1992년 서강대학교 전산학과(석사). 1993년~1996년 공군사관학교 교관(전임강사) 2003년 미국 Univ. of Southern California(박사). 2003년 미국 USC ISI Postdoctoral Research Associate. 2004년~현재 한국정보통신대학교 공학부 조교수. 관심분야는 소프트웨어공학, 웹 공학