큐잉 네트웍 모델을 이용한 소프트웨어 아키텍처 설계 단계에서의 성능 예측 방법론

(The Methodology for Performance Prediction in Architectural Design Stage of Software using Queuing Network Model)

윤 현 상 † 장 수 현 † 이 은 석 ††
(Hyunsang Youn) (Suhyeon Jang) (Eunseok Lee)

요 약 소프트웨어의 개발 초기 단계에서 성능을 예측하는 기법은, 비 기능적 요구사항의 검증 및 소프트웨어의 성능을 향상시키기 위해 중요한 이슈가 되었다. 이를 위해 소프트웨어의 아키텍처를 수학적인 분석 모델로 변환시키는 기법들이 등장하게 되었다. 그러나 에이전트 기반 시스템을 개발하는 경우, 기존 방법들은 에이전트 플랫폼의 성능을 정확하게 반영하지 못하기 때문에, 정확한 성능 예측 및 분석에 적용할 수 없다. 본 논문에서는 정규화된 의미 기술언어를 이용하여 에이전트 기반 시스템 아키텍처의 성능을 예측하는 기법을 제안한다. 본 방식은 UML로 기술된 시스템의 아키텍처를 하드웨어 및 소프트웨어 플랫폼의 성능이 반영된 분석 모델로 변환시킨다. 성능 예측의 정확도를 평가하기 위해, 과거에 연구했던 전자상거래 시스템을 확장한 유비쿼터스 상거래 시스템 시나리오를 기반으로 프로토타입을 구현하여 성능을 측정하고 생성된 분석 모델로부터 측정된 성능 결과와 비교하였다. 그 결과 약 80%의 정확도를 보였다.

키워드 : 성능 예측, 소프트웨어 개발, 아키텍처, 분석 모델

Abstract It is important issue for software architects to estimate performance of software in the early phase of the development process due to the need to verify non-functional requirements and estimation of performance in various stages of architectural design. In order to analyze performance of software, there are many approaches to translate software architecture represented by Unified Modeling Language, into analytical models. However, in the development of agent-based systems, these approaches ignore or simplify the crucial details of the underlying performance of the agent platform. In this paper, we propose performance prediction methodology for agent based system using formal semantic descriptions, and then, we transform the descriptions into queuing network model which model reflects performance of hardware and software platform. We prove the accuracy of proposed methodology using prototype implementation. The accuracy is summarized at 80%.

Key words : Performance, Software Development, Architecture, Analytic Model

1. 서 론

소프트웨어의 설계 초기 단계에서 성능을 예측하는 것은, 소프트웨어 개발 초기에 비 기능적 요구사항의 검증과, 최적화된 소프트웨어의 아키텍처를 결정을 위해, 오늘날 소프트웨어 공학 분야에서 중요한 이슈가 되고 있다.

분석 모델(Analytic model)에 기반한 접근법은 추상화된 모델을 기반으로 시스템의 성능을 측정하는데 매우 유용하다. 시스템의 요구사항을 반영하는 추상화된 초기 설계 모델로부터 수학적인 분석 모델을 얻기 위해서는 정량적인 파라메타 값들이 요구된다. 파라메타 값을 기반으로 분석 모델은 시스템의 행위 및 성능을 표현할 수 있는 다양한 수식을 제공한다. 따라서, 분석 모델을 통하여 설계자는 시스템의 행동을 예측할 수 있고, 시스템을 구현하지 않고도 그 시스템의 성능을 분석 및 예측할 수 있다[1,9].

· 본 연구는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발사업, 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업ITTA-2006-(C1090-0603-0046), 과학기술부의 특정기초연구사업 R01-2006-000-10954-0, 교육인적자원부의 2단계 BK21사업의 연구결과로 수행되었음

† 학생회원 : 성균관대학교 전자전기컴퓨터공학과
wizehack@ece.skku.ac.kr
nickelion@ece.skku.ac.kr
†† 종신회원 : 성균관대학교 전자전기컴퓨터공학과 교수
eslee@ece.skku.ac.kr

논문접수 : 2007년 4월 25일
심사완료 : 2007년 6월 28일

일반적으로 소프트웨어의 성능을 평가하기 위한 분석 모델로 확장형 큐잉 네트워크(Extended QN), 계층형 큐잉 네트워크(Layered QN)와 같은 큐잉 네트워크 기반의 모델을 많이 사용한다[3]. 따라서 오늘날 큐잉 네트워크 기반의 모델을 도출해내는 변환 기술은 소프트웨어 아키텍처를 통하여 분석 모델을 얻는데 유용한 접근 방법으로 활용되고 있다. 그러나 기존 모델 변환 방법을 다중 에이전트 기반의 시스템시스템 개발에 적용하는 것은 에이전트 플랫폼의 특성을 반영하지 못하기 때문에 정확한 성능 평가가 이루어 지지 못한다.

다중 에이전트 기반 시스템을 구성하는 지능형 에이전트들은 로컬 또는 분산 환경에 관계없이 에이전트 플랫폼을 필요로 한다. 에이전트 플랫폼은 에이전트의 동작, 에이전트들이 제공하는 서비스 디스커버리 및 에이전트의 상태 관리를 수행한다. 로컬 또는 분산 시스템에서 동작하는 에이전트는 에이전트 플랫폼에서 제공하는 서비스들을 이용하기 위해 다른 에이전트들과 상호작용한다. 에이전트 기반 시스템의 동작은 에이전트를 구현한 어플리케이션에 특화된 코드와 에이전트 플랫폼을 기반으로 이루어진다. 따라서 다중 에이전트 기반 시스템의 성능은 다음의 사항들에 의존하게 된다.

1) 에이전트 플랫폼의 성능
2) 어플리케이션의 성능
3) 시스템을 구성하는 하드웨어의 성능

본 논문에서 우리는 에이전트 기반 시스템의 성능을 초기 설계 단계에서 측정할 수 있는 방법을 제안하고자 한다. 이러한 접근 방법은 에이전트 기반 시스템의 프로토타입을 구현하지 않고 아키텍처 설계 명세로부터 시스템 각 모듈의 성능을 예측하고, 전체적인 시스템의 성능을 평가할 수 있는 방법이다.

본 논문은 다음과 같이 구성된다. 2장에서는 소프트웨어의 성능을 분석 및 측정하는 다양한 연구에 관해 알아본다. 3장에서는 유비쿼터스 상거래 시스템을 기반으로 다중에이전트 기반 시스템의 성능을 예측하는 방법에 관하여 알아본다. 4장에서는 예측 결과를 검증할 것이다. 끝으로 5장에서는 결론을 내릴 것이다.

2. 기본 개념

성능모델(Performance Model) 기반의 방법들은 소프트웨어의 초기 설계 단계에서 그 소프트웨어의 성능을 분석하는데 유용하게 사용될 수 있다. 소프트웨어 성능공학 커뮤니티(Software Performance Engineering Community)는 소프트웨어의 개발 프로세스에서, 설계와 성능 모델링의 통합을 오랜 기간 동안 주장해왔다. 그러한 설계 방식은 UML에서 제공하는 유스 케이스(Use Case), 객체 모델링(Object Modeling), 기능 모델링(Functional Modeling)을 이용하여 시스템을 모델링하고. 큐잉 네트워크 모델과 같은 분석 모델로 변환하는 방법을 사용한다[4,5,7,10].

UML을 이용한 시스템의 모델로부터 분석 모델을 얻을 수 있는 파라메타 값을 지정하기 위해 스케줄링, 성능, 시간에 관한 UML 프로파일이 OMG의 표준으로 채택되었다. UML 프로파일의 목적은 실행 시간, 논리적 물리적 리소스, 동시적인 리소스 접근, 스케줄링과 같이 UML다이어그램들만으로는 표현하기 힘든 모델의 요소들을 표현하기 위한 것이다[8].

[4]에서 Woodside는 소프트웨어 개발 주기에서 요구사항을 아키텍처화 하는 단계에서의 성능을 분석하는 방법을 보여준다. 이 접근 방법은, UCM (Use Case Map)을 통해 기술한 시스템의 시나리오에서 LQN(Layered Queuing Network)을 도출하는 방식이다. Woodside는 UCM에서 LQN을 도출하는 것을 자동화 시키는 도구인 UCM2LQN을 구현하였다. 이것은 시스템의 시나리오를 정의할 수 있도록 지원하고, UCM에디터와 LQN 생성을 지원한다. 그리고 UCM 명세서를 통해 XML파일을 추출한다.

SPT 프로파일과 UML 명세서를 기반으로 분석모델을 도출해내는 많은 방법들이 있었다. 일반적으로 이러한 방식에서 소프트웨어의 워크플로우는 시퀀스 다이어그램이나 활동 다이어그램을 사용한다. 배포 다이어그램은 하드웨어와 소프트웨어의 자원을 기술하는데 사용한다. Cortellessa 와Mirandola는 다양한 UML 다이어그램으로부터 성능 모델을 생성하는 PRIMAUM을 제안하였다[6]. PRIMAUML에서, 소프트웨어의 아키텍처는 유스케이스 다이어그램, 시퀀스 다이어그램, 배포 다이어그램에 의해서 기술된다.

그러나 기존 컴퓨팅 시스템의 성능을 평가하기 위한 방법들은 에이전트 기반 시스템에 적용하기에는 한계가 있다. 기존 방법들은, 에이전트 플랫폼의 성능에 관한 중요한 사항들을 무시하거나 단순화시키기 때문이다. 기존 접근 방식을 사용할 경우 에이전트와 에이전트 플랫폼간에 상호 작용이 정확하게 모델링 되지 않는다. 또한 에이전트 플랫폼 내부에서 이루어지는 다양한 활동들도 나타낼 수 없기 때문에 분석 모델을 이용한 결과가 다소 부정확하게 된다. 기존 방식의 또 다른 문제점은 모델을 생성하는데 필요한 파라메타 값들이 필요하다는데 있다. 이러한 파라메타들은 설계 단계에서 명확하게 얻을 수 없다. 따라서 일반적으로 프로토타입을 구현하고 이것을 이용해서 모델을 제작하는데 필요한 파라메타 값들을 측정한다. 그러나 이런 방식은 복잡한 시스템을 개발하는 단계에서는 많은 시간적인 비용을 필요로 한다. 따라서 본 논문에서는 에이전트 플랫폼의 성능 모델

과, 다중에이전트 기반 시스템의 초기 설계 명세서를 기반으로 생성된 성능 모델을 이용하여 소프트웨어의 초기 설계 단계에서 성능을 평가하기 위한 방법을 제안한다.

본 논문에서 우리는 UML을 기반으로 에이전트 기반 시스템의 아키텍처설계를 정의하고, 시스템의 아키텍처로부터 분석 모델을 얻기 위한 정규화된 의미 기술언어(Formal Semantic Description: FSD)를 제안한다. 또한 정규화된 의미 기술언어로 기술한 아키텍처를 H/W와 에이전트 플랫폼의 성능이 반영된 큐잉 네트워크 모델로 변환시키는 방법을 제안한다.

본 논문에서 제안하는 방식은 다중 에이전트 기반시스템의 설계 초기 단계에서 소프트웨어의 아키텍처 설계자를 지원한다. 또한, 시스템의 성능관련 요구사항의 검증을 원하는 요구사항 분석가를 지원한다. 추가적으로 본 논문은 재사용 가능한 에이전트 플랫폼을 제공한다.

3. 에이전트 기반 시스템 성능 예측 방법론

본 장에서는 논문에서 제안하는 아키텍처 모델링 및 모델 변환을 설명하기 위해, 이전에 설계 및 구현하였던 모바일 상거래(mobile commerce)시스템[10]을 확장한 유비쿼터스 상거래 시스템을 개발에 적용하는 사례에 적용하는 것을 예를 들고자 한다. 본 시나리오를 통하여 우리는 시스템의 아키텍처와 그것을 변환한 큐잉 네트워크 모델을 명세화할 것이다.

시나리오: 오프라인 상점에 있는 점원은 고객에게제품에 관한 소개를 한다. 고객은 그가 가지고 있는 모바일 단말기에 있는 RFID리더를 통하여 제품에 부착된 RFID 테그를 읽고, 온라인 상점에서 판매되는 동일한 제품의 가격을 알아보고 가장 저렴한 가격으로 판매하는 온라인 상점에서 물건을 구입 한다. 온라인 상점에서는 해당 제품을 설명해준 오프라인 상점에게 일정 금액의 광고료를 지불한다.

3.1 시스템 아키텍처 모델링

본 장에서 우리는 다중 에이전트 기반 시스템의 성능을 평가할 수 있는 분석적 방법을 제안하고자 한다. 주어진 시스템 아키텍처의 성능을 분석하기 위해, 먼저 우리는 UML을 기반으로 에이전트 기반 시스템의 아키텍처설계를 정의한다. 그런 후에, 정규화된 의미 기술언어를 제안한다. 마지막으로, 본 의미 기술 언어로 작성된 시스템 아키텍처를 큐잉 네트워크 모델로 변환시키는 방법을 제안한다.

우리는 시스템의 아키텍처를 표현하고 에이전트 플랫폼과 아키텍처를 연관시키기 위해 구조적 관점(structural aspect)와 행위적 관점(behavioral aspect)으로 다중 에이전트 기반 시스템을 표현한다. 구조적 관점은 플랫폼 독립적인 구조와, 플랫폼 의존적인 구조로 구성된다. 플랫폼 독립적인 구조는 다중에이전트 기반 시스템을 구성하는 각 구성 요소들간의 연관관계를 식별하기 위한 것으로 클래스 다이어그램을 사용하여 표현한다. 플랫폼 의존적인 구조는 다중에이전트 기반 시스템을 실행시키는 에이전트 플랫폼과 하드웨어의 구조를 표현하기 위한 것으로 배포 다이어그램을 사용하여 표현한다. 시스템 아키텍처의 행위는 내적 행위(Internal behavior)와 외적 행위(External behavior)로 구성한다. 외적 행위는 커뮤니티를 구성하는 몇몇 에이전트들의 하위 목표(sub goal)과 역할(role)을 표현하기 위하여 필요하며 메시지 시퀀스 다이어그램을 사용한다. 마지막으로 각 에이전트가 가지고 있는 내적 행위를 표현하기 위하여 활동 다이어그램을 사용한다. 이러한 방법을 통해 시스템의 아키텍처를 표현할 때 우리는 SPT 프로파일을 통하여 큐잉 네트워크 모델을 도출하기 위한 파라메타 정보를 얻는다.

그림 1에서 클래스 다이어그램은, 유비쿼터스 상거래 시스템의 플랫폼 독립적인 구조를 나타낸다. 여기서 우리는 시스템을 구성하는 에이전트 및 데이타 베이스와 같은 다양한 객체를 식별하고, 그들간의 연관 관계를 식별한다. 예를 들면 그림을 통해, 여러 개의 사용자 에이전트(User Agent)가 하나의 주문 에이전트(Order Agent)에 접속할 수 있다는 것을 알 수 있다. 에이전트 기반 시스템의 플랫폼의존적인 구조는 그림 1의 배포다이어그램을 통해 알 수 있다. 본 그림을 통하여 시스템을 구성하는 에이전트들의 통신 환경과, 동작환경을 알 수 있다. 예를 들어 오프라인 샵(Offline-Shop) 에이전트와 주문(Order) 에이전트는 같은 플랫폼에서 동작하지만 다른 서버 상에서 운용되는 에이전트이고, 사용자 에이전트와 온라인 샵(Online-Shop) 에이전트는 동작 플랫폼 및 하드웨어가 모두 다른 환경에서 동작한다. UML다이어그램으로부터 큐잉 네트워크의 생성 및 성능 평가를 위하여 UML을 구성하는 각 구성요소를 위한 스테레오 타입을 정의하였다. 각 스테레오 타입은 <<*device*>>, <<*agent_platform*>>, <<*communicator*>>, <<*Agent*>>이다. 이들은 각각 디바이스, 에이전트 플랫폼, 인터넷이나 랜(LAN)과 같은 통신 인프라, 에이전트를 나타낸다. 시스템을 구성하는 에이전트들의 외적 행위는 그림 1의 시퀀스 다이어그램으로 나타낼 수 있다. 개발자는 메시지 시퀀스 다이어그램을 통해서, 하나의 시나리오를 기반으로 동작하는 에이전트들의 역할을 표현할 수 있다. 시스템의 외적 행위에 관한 기술 작업이 끝나면, 시스템을 구성하는 각 에이전트의 내적 행위를 기술하는 작업을 한다. 이것은 활동 다이어그램으로 기술되며, SPT 프로파일을 이용하여 수치적인 파라메타 값을 입력한다.

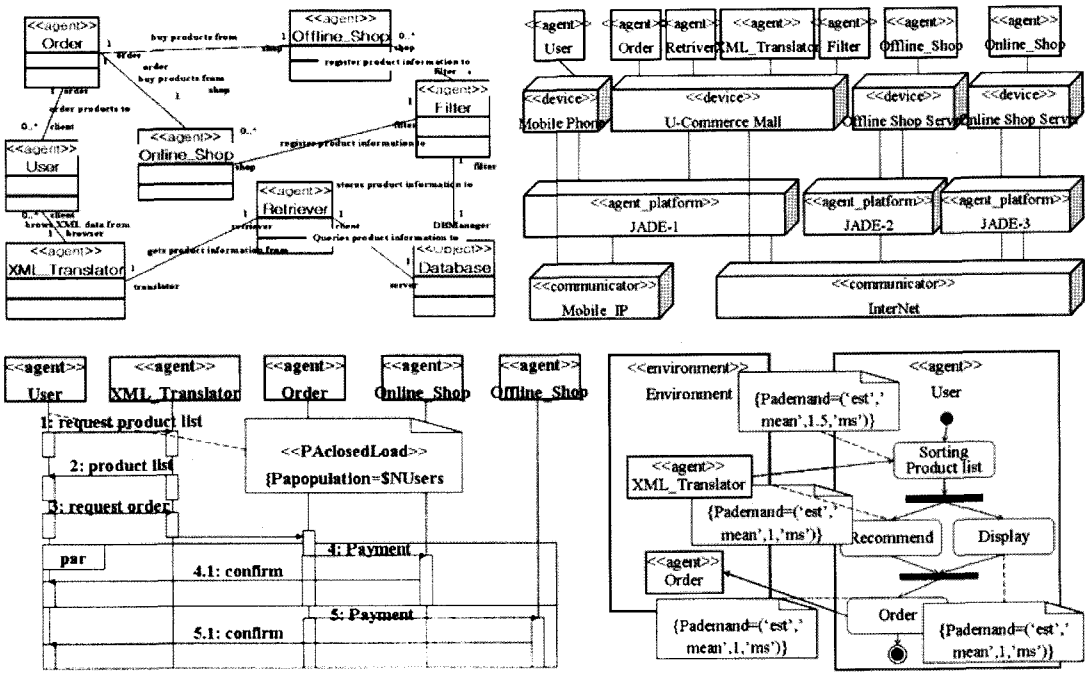


그림 1 UML 기반 U-Commerce System의 아키텍처 모델

그림 1의 활동 다이어그램은 사용자 에이전트가 다른 에이전트들과의 상호 작용을 하면서 내부적인 동작을 수행하는 것을 표현한다. User 에이전트는 XML-Translation 에이전트로부터 상품에 관한 정보를 XML 형태로 입력받아서 그 중 적절한 상품을 사용자에게 추천하거나, 받은 상품에 관하여 디스플레이 한다. 그런 후 사용자가 원하는 상품의 구매를 지원한다. 각 활동에 SPT 프로파일 형식으로 기술한 'PAdemand'를 통해서 우리는 큐잉 시스템의 서비스 시간을 구할 수 있다. 예를 들어 그림 2에서, User 에이전트는 'Sorting'작업을 1.5ms 동안 수행한 수에, 'Recommend'과 'Display'작업을 각각 병렬적으로 수행한다. 수행한다. 그리고 마지막으로 'Order'작업을 1ms동안 수행한다. 따라서 우리는 3.5ms의 서비스 시간을 얻을 수 있다.

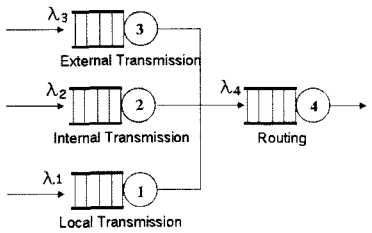


그림 2 FIPA 에이전트 플랫폼의 메시지 전달에 관한 큐잉 모델

네트워크 대역, 암호화 부호화 시간등과 같은 정보는 에이전트 플랫폼과 H/W환경의 특성을 반영한다. 이러한 정보들을 성능 평가에 정확히 반영하기 위해 플랫폼 내부의 구현과 동작 시간을 측정해야 한다. 따라서 우리는 그것의 성능을 측정하기 위한 간단한 어플리케이션을 구현하였다.[1)]

3.2 에이전트 플랫폼 모델링

성능 모델은 에이전트 플랫폼의 행위를 정확하게 반영해야만 한다. FIPA(Foundation of Intelligent Physical Agent) standard specification[7]은 에이전트 플랫폼의 메시지 전송을 위한 행동을 명세화 하였다. 우리는 FIPA에서 제안한 명세서를 기반으로 에이전트 플랫폼을 위한 큐잉 네트워크를 모델링 하였다. 우리는 플랫폼을 구성하는 각 모듈의 상세 내용을 추상화 시켰다. 큐잉 네트워크 모델의 큐와 서버의 특성을 결정 하면, 큐잉 네트워크 이론을 통해서 시스템의 성능에 관한 측정치를 얻을 수 있다.

우리는 에이전트 플랫폼에 관한 QNM(Queuing Network Model)을 작성하였다. 여기서 우리는 에이전트 플랫폼을 구성하는 각각의 모듈간의 상호 작용들을 추상화시켜 표현하였다. 모델을 구성하는 서버와 큐의 특성과 그것들이 필요로 하는 파라메타 값들을 결정 하면,

1) http://selab.sku.ac.kr/wizehack/getPara.html

큐잉 이론을 기반으로 모델로부터 필요로 하는 성능에 관한 측정치(Performance measures)를 얻을 수 있다.

그림 1은 *External, Internal, Local* 기반의 전송 타입을 보여준다. *External* 전송은 에이전트가 다른 에이전트 플랫폼에 있는 에이전트에게 메시지를 전송하는 경우이다. *Internal* 전송은, 에이전트가 같은 에이전트 플랫폼을 공유하지만, 다른 H/W디바이스를 사용하는 에이전트에게 메시지를 전송하는 경우이다. *Local* 전송은 에이전트가 같은 플랫폼 이면서, 동일한 H/W디바이스를 사용하는 에이전트에게 메시지를 전송하는 경우이다. 이것을 기반으로 우리는 어플리케이션 도메인의 큐잉 네트워크와 에이전트 플랫폼의 큐잉 네트워크 모델을 조합한다. 어플리케이션의 큐잉 네트워크 모델은 UML기반 아키텍처에 추가된 SPT 프로파일을 통해서, 모델이 필요로 하는 파라메타 정보를 얻을 수 있다. 에이전트 플랫폼에서 제공하는 encoding/decoding mechanism, network bandwidth등의 정보들은 사용하는 H/W 또는 에이전트 플랫폼의 종류에 따라서 달라질 수 있다.

정확한 성능 측정을 위해, 우리는 H/W 및 플랫폼 벤더들간의 특성에 따라 에이전트 플랫폼의 파라메터 값을 조율할 수 있어야 한다. 그러나 그러한 정보들은 플랫폼의 내부에 숨겨져 있기 때문에 설계 단계에서 결정하는 일은 매우 어렵다. 따라서 우리는 플랫폼의 성능을 측정하기 위해 간단한 프로그램을 제작하였다. 이 프로그램은 플랫폼의 내부 구조에 상관 없이 자바언어로 구현된 시스템의 내부를 구성하는 각 컴포넌트, 객체, 모듈들의 작업 처리 시간을 측정하는 프로그램이다. 이 프로그램을 통해 측정하기 원하는 모듈의 입력 시간과, 출력 시간을 측정하여 수행 시간을 계산한다.

3.3 모델 변환

다중 에이전트 기반 시스템과 큐잉 네트워크 모델을 기술하기 위해, 우리는 그것을 정규화된 의미 기술로 작성해야 한다. 그러나 현 시점에서 UML의 의미구조는 정규화 되어 있지 않다. 따라서 우리는 UML과 큐잉 네트워크를 위한 정규화된 의미구문을 정의한다. 본 논문에서 우리가 제안하는 방법은 UML을 기반으로 작성된 정규화된 의미 구문을 통해서 큐잉 네트워크 모델을 생성하는 것이다. 본 논문에서 제안하는 정규 의미 기술의 문법체계는 다음과 같다.

에이전트 식별정보는 에이전트 이름, 다른 에이전트에게 메시지를 동시에 멀티플랙싱 하는 수, 작업 처리 시간, 플랫폼 식별정보로 구성된다.

$$\xi_{agent}^{id} = (\xi_{agent}^{name}, \xi_{agent}^{nom}, \xi_{agent}^{ptime}, \xi_{env}^{id})$$

에이전트가 아닌 다른 요소들, DBMS, Web application 등을 객체(Object)라고 부른다. 객체 식별정보는 객체 이름, 객체의 멀티플랙싱 수, 작업 처리 시간, 객체가 위치한 디바이스의 식별정보이다.

$$\xi_{obj}^{id} = (\xi_{obj}^{name}, \xi_{obj}^{nom}, \xi_{obj}^{ptime}, \xi_{dev}^{id})$$

에이전트 플랫폼의 식별정보는 에이전트 플랫폼이름, 커뮤니케이터 식별정보, 디바이스 식별정보를 가진다. 커뮤니케이터 식별정보는 유/무선 랜, 블루투스 같은 커뮤니케이션 네트워크 타입을 말한다.

$$\xi_{env}^{id} = (\xi_{env}^{name}, \xi_{env}^{nom}, \xi_{dev}^{id})$$

커뮤니케이션 타입은 에이전트 플랫폼의 통신 방식을 의미한다. 에이전트가 다른 에이전트와 커뮤니케이션을 수행하는 경우, *external communication (ext (ttime, rtime))*, *internal communication (int (ttime, rtime))*, *local communication (local (ttime, rtime))*의 세가지 커뮤니케이션 타입 중 한 가지가 결정되게 된다. 각 커뮤니케이션 타입은 전송시간(*ttime*)과 라우팅(*rtime*) 시간으로 구성된다.

$$ctype = (ext(ttime, rtime) \vee int(ttime, rtime) \vee local(ttime, rtime))$$

커뮤니케이션 식별정보는 *unique, altinative, parallel* 중 하나로 결정될 수 있다. *unique*는 1의 확률로 메시지가 목적지까지 전달되는 것을 의미한다. *alitinative*는 p의 확률로 메시지가 전달되는 위치가 결정되는 것을 말한다. *parallel*은 동시에 여러 곳으로 메시지가 전달되는 것을 의미한다.

$$commid = (unique \vee altinative(p) \vee parallel)$$

이제 마지막으로 우리는 다중에이전트 기반 시스템 아키텍처의 변환기술을 제안 한다. 먼저 우리는 큐잉 네트워크를 위한 최종 결과물을 정의한다. 최종 결과물을 모델링 하면 다음과 같다. 큐잉 네트워크는 큐, 서버; 도착 비율 또는 전체 작업 수; 서비스 비율, 서버에서 큐로의 라우팅 확률, 에이전트 플랫폼을 위한 큐잉 네트워크 모델(AP)로 구성된다.

$$X_{queuing} = (\sigma_q, \sigma_s, (\lambda \vee t), \mu, p, X_{queuing}^{Platform})$$

에이전트가 메시지를 받을 때, 그 에이전트는 FIFO방식으로 메시지를 처리하게 된다. 이러한 메커니즘에 따라서, 우리는 하나의 에이전트나 하나의 객체는 하나의 큐를 가진다고 정의할 수 있다.

에이전트나 객체가 다른 에이전트들에게 메시지를 동시적으로 보낼 때, 우리는 하나의 큐에 많은 서버가 존재하고 이것이 동시적으로 여러 다른 큐에게 전달되는 것으로 모델링을 할 수 있다. 그러나 이러한 방식은 큐에 존재하는 작업의 대기 시간에 영향을 미친다. 따라서 우리는 서버의 수에 따라서 서버의 서비스 율을 조율할 필요가 있다. 먼저 하나의 큐에 있는 작업을 하나의 서

버가 처리한다고 가정했을 때, T_1을 해당 에이전트의 서비스 타임이라 정하고, 멀티케스팅을 위해 메시지를 복사하는 시간을 α라고 하자. 그러나 α는 거의 0에 가깝게 되기 때문에 무시된다. 이때 k개의 메시지를 멀티케스팅하기 위해 k개의 서버를 두었을 때, 각각의 서비스 타임 μ_k는 아래와 같다.

$$T_1 = \xi_{agent}^{ptime}$$
$$T_2 = 2T_1 + \alpha$$
$$\cdots$$
$$T_k = kT_1, (\mu = \frac{1}{T})$$
$$\therefore \mu_k = \frac{1}{kT_1}$$

에이전트 플랫폼을 모델링한 큐잉 네트워크 모델에서 메시지의 라우팅 확률을 결정하기 위해, 먼저 커뮤니케이션 타입을 식별할 필요가 있다. 이것을 위해 우리는 정규화된 기술 구문의 내용을 파악해서 *Internal, External, Local* 커뮤니케이션 타입을 결정하고, 메시지의 라우팅 시간을 파악한다.

$$IF(\xi_{agent}^{id1} \neq \xi_{agent}^{id2}) \wedge (\xi_{env}^{id1} \neq \xi_{env}^{id2})$$
$$X_{queuing}^{Platform} \leftarrow (\sigma_q^{ext}, \sigma_s^{ext}, \omega(\sigma_q^{routing}, 1))$$
$$IF(\xi_{agent}^{id1} \neq \xi_{agent}^{id2}) \wedge (\xi_{env}^{id1} = \xi_{env}^{id2}) \wedge (\xi_{dev}^{id1} \neq \xi_{dev}^{id2})$$
$$X_{queuing}^{Platform} \leftarrow (\sigma_q^{int}, \sigma_s^{int}, \omega(\sigma_q^{routing}, 1))$$
$$IF(\xi_{agent}^{id1} \neq \xi_{agent}^{id2}) \wedge (\xi_{env}^{id1} = \xi_{env}^{id2}) \wedge (\xi_{dev}^{id1} = \xi_{dev}^{id2})$$
$$X_{queuing}^{Platform} \leftarrow (\sigma_q^{rocol}, \sigma_s^{rocol}, \omega(\sigma_q^{routing}, 1))$$

커뮤니케이션 타입으로부터 우리는 에이전트 플랫폼의 큐잉 네트워크 모델에 기반한 큐에서 서버로의 라우팅 확률을 다음과 같이 결정할 수 있다.

$$IF\ X_{queuing}^{Platform} = (\sigma_q^{ext}, \sigma_s^{ext}, \omega(\sigma_q^{routing}, 1))$$
$$\sigma_s^{id1} \leftarrow (\mu, \omega(\sigma_q^{ext}, 1))$$
$$\sigma_s^{routing} \leftarrow (\mu, \omega(\sigma_q^{id2}, 1))$$
$$IF\ X_{queuing}^{Platform} = (\sigma_q^{int}, \sigma_s^{int}, \omega(\sigma_q^{routing}, 1))$$
$$\sigma_s^{id1} \leftarrow (\mu, \omega(\sigma_q^{int}, 1))$$
$$\sigma_s^{routing} \leftarrow (\mu, \omega(\sigma_q^{id2}, 1))$$
$$IF\ X_{queuing}^{Platform} = (\sigma_q^{rocol}, \sigma_s^{rocol}, \omega(\sigma_q^{routing}, 1))$$
$$\sigma_s^{id1} \leftarrow (\mu, \omega(\sigma_q^{rocol}, 1))$$
$$\sigma_s^{routing} \leftarrow (\mu, \omega(\sigma_q^{id2}, 1))$$

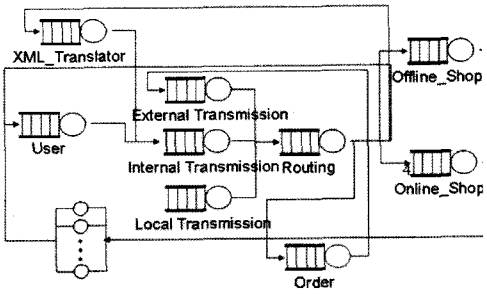


그림 3 유비쿼터스 상거래 시스템의 큐잉 네트워크 모델

위의 변환 방법으로부터 우리는 다음과 같이 그림 1의 시스템 아키텍처를 아래 그림 3과 같은 큐잉 네트워크 모델로 변환시킬 수 있다.

4. 시스템 성능 예측 및 모델 검증

본 논문에서 제안하는 모델 변환 기법을 검증하기 위해 우리는 과거에 수행했던 모바일 커머스 시스템을 기반으로, 유비쿼터스 상거래 시스템의 프로토타입을 구현하였다.

표 1은 유비쿼터스 상거래 시스템을 구성하는 프로토타입의 개발 환경에 관한 정보를 나타낸다. 분석 모델을 이용하여 시스템의 성능을 예측하기 위해, 에이전트 플랫폼의 메시지 전송에 관한 파라메타 정보가 필요하다. 이러한 정보는 JADE에서 제공하는 사용자 인터페이스인 RMA에서 DUMMY에이전트를 생성하여 측정 가능하다. 측정한 에이전트 플랫폼의 메시지 전송에 관한 성능은 표 2와 같다.

표 2 에이전트 플랫폼의 메시지 전송 타입

	Transmission Time
External Communication	12 (ms)
Internal Communication	3 (ms)
Local Communication	1 (ms)

표 3은 분석 모델을 기반으로 예측한 유비쿼터스 상거래 시스템의 평균 응답시간과 프로토타입을 구현하여 측정한 결과를 비교한 결과이다. 표 3을 통해서 우리는 본 논문에서 제안한 방식을 기반으로 생성된 분석 모델의 예측 정확도가 약 80%정도라는 것을 알 수 있다. 비

표 1 유비쿼터스 상거래 시스템 프로토타입

	하드웨어	운영체제	에이전트 플랫폼
Offline Shop Server	Pentium(R) 4 CPU 2.80GHz, 1.00 GB RAM	Windows XP	JADE-Leap
Online Shop Server	Pentium(R) 4 CPU 2.80GHz, 1.00 GB RAM	Windows XP	JADE-Leap
User Device	Pentium(R) 4 CPU 2.80GHz, 512 MB RAM	Windows XP	JADE-Leap
U-Portal Server	Pentium(R) 4 CPU 2.80GHz, 1.00 GB RAM	Windows XP	JADE-Leap

표 3 시스템 평가 및 분석 모델의 검증

시스템 구성 요소	시스템 부하량 (CPU Utilization)			
	분석 모델 기반 결과(%)	시스템 측정 결과 (%)	오차 (ms)	신뢰도(%)
U-Commerce Server	91.24	87	4.24	95.35292
Online Shop Server	57.3	54	3.3	94.24084
Offline Shop Server	60.64	63	-2.36	96.25397
시스템 구성 요소	작업 처리량 (Transaction Throughput)			
	분석 모델 기반 결과	시스템 측정 결과	오차 (ms)	신뢰도(%)
User Agent	30.34	28.28	2.06	93.21028
XML Translator Agent	14.53	13.43	1.1	92.42946
Order Agent	0.62	0.72	-0.1	86.11111
Online Shop Agent	6.79	8.97	-2.18	75.69677
Offline Shop Agent	8.37	11.54	-3.17	72.53033
시스템 구성 요소	평균 시스템 응답 시간 (System Response Time)			
	분석 모델 기반 결과(ms)	시스템 측정 결과 (ms)	오차 (ms)	신뢰도(%)
User Agent	992	1329	-337	74.64258841
XML Translator Agent	534	634	-100	84.22712934
Order Agent	19	28	-9	67.85714286
Online Shop Agent	297	360	-63	82.5
Offline Shop Agent	384	410	-26	93.65853659

록 Order Agent의 응답시간에 관한 정확도가 70% 미만이지만 실질적인 오차는 9(ms)로써 실제 시스템의 성능에 거의 영향을 미치지 않는다.

5. 결 론

본 논문에서, 우리는 시스템의 초기 설계 단계에서의 성능 분석 및 예측을 위해 다음과 같은 세가지 방법을 제안하였다. 첫째로, UML을 이용하여 에이전트 기반 시스템의 아키텍처를 모델링 하는 방법론을 제안하였다. 둘째로, 에이전트의 구조 및 행위를 기술할 수 있는 정규화된 의미 기술언어를 제안하였다. 마지막으로, 이러한 의미 기술 언어를 큐잉 네트워크 모델로 변환시키는 방법에 관하여 제안하였다. 성능 예측의 정확도를 평가하기 위해, 과거에 연구했던 전자상거래 시스템을 확장한 유비쿼터스 상거래 시스템 시나리오를 기반으로 프로토타입을 구현하여 성능을 측정하고 생성된 분석 모델로부터 측정된 성능 결과와 비교하였다. 그 결과 약 80%의 정확도를 보였다.

참 고 문 헌

[1] Simonetta Balsamo, Antinisca Di Marco, Paola Inverardi, Marta Simeonl, Model-Based Performance Prediction in Software Development: A Survey, IEEE TRANSACTION ON SFTWARE ENGINEERING, Vol.30, No.5, MAY (2004).

[2] G. Frank, A. Hubbard, S. Majumdar, D.C. Petrju, J. Rolla, and C.M. Woodside, A Toolset for Performance Engineering and Software Design of Client-Server Systems, Performance Evaluation, Vol.24, No.1-2, pp. 117-135, (1995).

[3] J.A. Rolia and K.C. sevcik, "The Method of Layers," IEEE TRANSACTION ON SFTWARE ENGINEERING, Vol.21, No.8, pp. 682-668, (1995).

[4] C.M. Woodside, C. Hrischuk, B. Selic, and S. Brayarov, "Automated Performance Modeling of Software Generated by a Design Environment," Performance Evaluation, Vol.45, pp. 107-123, (2001).

[5] K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Compter Science Application, John Wiley and Sons, (2001).

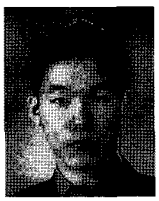
[6] V. Cortellessa and R. Mirandola, "Deriving a Queueing Network Based Performance Model from UML Diagrams," ACM Proc. International Workshop Software and Performance, pp. 58-70, (2000).

[7] C.U. Smith and L.G. Williams, "PASASM: A Method for the Performance Assessment of Software Architectures," Proc. Third Int'l Workshop Software and Performance (WOSP'02) pp. 179-189.

[8] OMG, UML Profile for Schedulability, Performance, and Time Specification, January (2005).

[9] Marco Scarpa, Antonio Puliafito, Massimo Villari, and Angelo Zaia "A Modeling Technique for the Performance Analysis of Web Searching Applications," IEEE Trans. Software Eng., Vol.16, No.11, pp. 1339-1356, November (2004).

[10] Eunseok Lee and Jionghua Jin, "A Next Generation Intelligent Mobile Commerce System," LNCS 3026, Springer-Verlag, pp. 320-331, Apr. (2004).



윤 현 상

2004년 2월 서경대학교 정보통신공학과(공학사). 2006년 2월 성균관대학교 대학원 컴퓨터공학과(공학석사). 2006년 8월~현재 성균관대학교 대학원 컴퓨터공학과 박사과정. 관심분야 유비쿼터스 컴퓨팅, 소프트웨어공학, 다중 에이전트 시스템, 소프트웨어 아키텍처, 소프트웨어 성능 분석

장 수 현

2006년 2월 성균관대학교 정보통신공학과(공학사). 2006년 3월~현재 성균관대학교 대학원 컴퓨터공학과(석사과정). 관심분야는 유비쿼터스 컴퓨팅, 소프트웨어공학, 다중 에이전트 시스템, MDA기반 소프트웨어 개발

이 은 석

1985년 2월 성균관대학교 전자공학과(공학사). 1988년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학석사). 1992년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학박사). 1992년~1994년 일본 미쓰비씨 정보전자연구소 특별연구원. 1994년~1995년 일본 동북(Tohoku)대학교 Assistant Prof. 1995년 3월~현재 성균관대학교 정보통신공학부 교수. 관심분야는 소프트웨어공학, 오토노믹/유비쿼터스 컴퓨팅, 에이전트지향 지능형시스템 등