비용을 고려하고 아키텍처 평가를 지원하는 소프트웨어 아키텍처 비용 추정 기법

(A Software Architecture Cost Estimation Method to Support Architecture Evaluation with Consideration of Cost)

조 시 호 † 이 준 하 † 박 수 용 ††
(Siho Choue) (JunHa Lee) (SooYong Park)

요 약 소프트웨어 제품의 시장 경쟁력을 향상시키기 위해서는 이해관계자의 요구사항에 부합하는 품질을 제공하는 동시에 개발 예산 내에 개발 가능한 아키텍처를 획득할 수 있는 방안이 요구된다. 하지만 아키텍처의 품질 검토 및 선정을 위해 사용되는 기존 아키텍처 평가 기법은 아키텍처 설계를 통해 획득 가능한 품질속성에만 초점을 맞추고 있어 아키텍처가 개발 비용에 미치는 영향을 체계적으로 고려하지 않는다. 본 논문에서는 소프트웨어 비용 추정 모델인 COCOMO II를 적용한 아키텍처 비용 추정 기법을 제안하여 기존 아키텍처 평가와 병행 가능한 비용 분석을 통해 기존 아키텍처 평가 기법을 보완하며 품질과 비용이 함께 고려된 아키텍처 선정을 지원한다. 제안 기법의 정확성 검증을 위해 본 논문의 기법을 RPS (Robot Patrol System)의 아키텍처 후보 목록에 적용한 뒤 비용 추정 결과를 각 아키텍처 후보의 실측 공수와 비교하였다.

키워드 : 아키텍처 평가, 아키텍처 비용 추정, 아키텍처 선정

Abstract Improving the competitiveness of software products in the market involves procuring the means to design software architecture that deliver qualities necessitated by stakeholder requirements within allocated budget, thereby improving the cost-effectiveness of the end product. Currently, software architecture evaluation methods are used to predict and review qualities inherent in software architecture designs and to choose a candidate architecture that delivers desired qualities. Existing software architecture evaluation methods, however, fail to address the cost considerations dependent on the architecture chosen for product implementation. In this paper we suggest a cost estimation method for software architecture which adapts the cost drivers in the software cost estimation model COCOMO II to support cost estimation during architecture evaluation. The suggested method can be performed in coordination with existing software architecture evaluation efforts and supplements existing architecture evaluation techniques with guidelines for identifying and evaluating cost drivers in candidate software architectures without incurring extra overhead. The accuracy of the cost estimation using the suggested method is verified through application of the method to the architecture candidates used in RPS (Robot Patrol System), a surveillance embedded system.

Key words : Architecture evaluation, architecture cost estimation, architecture selection

· 이 논문은 2009 한국 소프트웨어공학 학술대회에서 '비용을 고려한 아키텍처 평가를 지원하는 소프트웨어 아키텍처 비용 추정 기법'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 서강대학교 컴퓨터공학과
josh@sogang.ac.kr
sypark@sogang.ac.kr
†† 정 회 원 : 서강대학교 컴퓨터공학과 교수
zuna.lee@gmail.com

논문접수 : 2009년 3월 3일
심사완료 : 2009년 12월 14일

정보과학회논문지: 소프트웨어 및 응용 제37권 제2호(2010.2)

1. 서 론

현대 산업에서 소프트웨어가 차지하는 역할이 확대됨에 따라 소프트웨어 시장은 보다 양질의 소프트웨어를 요구하게 되었고, 개발조직은 시장의 요구에 부응하여 더 나은 제품을 제공하기 위한 경쟁을 가속화하고 있다. 가열된 경쟁 속에서 개발조직이 생존하기 위해서는 경쟁력을 갖춘 소프트웨어를 개발할 수 있는 역량을 갖추어야 한다. 개발조직의 경쟁력이란 이해관계자가 필요로 하는 기능과 품질을 합리적인 가격에 시기 적절하게 제공하는 능력을 의미한다[1]. 특히 이해관계자의 만족도에 큰 영향을 미치는 소프트웨어 품질속성(quality attribute)을 주어진 예산 내에 설계하고 개발하는 능력은 시장 환경의 변화에 빠르게 대처하고 최대의 ROI (Return-On-Investment)를 획득하여 개발조직이 경쟁 속에서 생존할 수 있게끔 하는 중요한 역량이다[2].

소프트웨어 아키텍처는 완성된 소프트웨어가 발현할 품질을 결정하는 중요한 개발 산출물이다[3-5]. 하지만 품질속성이 갖는 모호성[6,7]으로 인해 아키텍처 설계 상에서 최종 제품의 품질 요구사항 충족 여부를 직접 확인하기 어려우므로, 아키텍트는 아키텍처 평가기법을 사용하여 아키텍처 설계에 내재된 품질속성을 구체화하고 제품이 제공하게 될 품질을 검토한다. 특히 자연어에 기반하여 기술적으로 접근이 쉽고 품질속성의 이해와 구체화가 용이한 시나리오 기반 아키텍처 평가 기법이 아키텍처 평가 수행을 위해 널리 사용되고 있으며, 대표적으로 Carnegie Melon 대학의 SEI(Software Engineering Institute)에서 개발한 ATAM(Architecture Tradeoff Analysis Method)을 들 수 있다[8,9]. 시나리오 기반 아키텍처 평가 기법은 아키텍트가 아키텍처 설계의 품질 요구사항 부합 여부 검토, 대안 아키텍처 후보의 식별, 최적 아키텍처의 선정을 수행할 수 있도록 지원한다[10-12]. 하지만 기존의 시나리오 기반 아키텍처 평가 기법은 아키텍처가 지원하는 품질속성만을 검토 대상 및 아키텍처 선정 기준으로 적용하며 아키텍처 후보의 개발 비용을 고려하지 않고 있다[10,11]. 비용을 고려하지 않는 아키텍처 후보 선정은 품질 요구사항을 충족시키더라도 예산을 초과하게 되는 최종 제품으로 이어질 수 있으며, 이는 완성된 산출물의 경제성을 떨어뜨리고 개발조직의 경쟁력을 약화시키는 결과를 초래할 수 있다[2].

시나리오 기반 아키텍처 평가 기법의 비용 고려 부족 문제를 해결하기 위해 본 논문에서는 소프트웨어 비용 추정 모델인 COCOMO II[13]를 아키텍처 후보의 비용 예측에 적용한 아키텍처 후보 비용 추정 기법을 제안한다. COCOMO II 모델에서 정의된 비용요소(cost driver)인 Effort Multiplier(EM)를 아키텍처 후보의 품질 평가 결과에 적용하여 추정 비용에 대한 체계적인 근거를 도출하며, COCOMO II의 EM 정의에 기반한 비용요소 평가 기준을 제공하여 아키텍처 개발 비용 예측 경험이 없는 아키텍트도 COCOMO II 모델의 신뢰성에 기반한 아키텍처 후보 비용 추정이 가능하도록 하였다. 비용 추정의 정확성을 검증하기 위해 제안 기법을 보안용 실시간 임베디드 시스템인 RPS(Robot Patrol System)의 개발을 위해 설계된 아키텍처 후보에 적용하였으며 추정된 비용을 실제 구현에 소요된 공수와 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서는 아키텍처 평가와 비용 추정을 위한 관련 연구를 설명하고, 3 장에서는 아키텍처 후보 비용 추정 프로세스를 기술한다. 4 장에서는 본 논문의 제안 기법을 RPS에 적용하여 비용 추정의 정확성을 검증하며, 마지막으로 제안 기법의 적용 효과 및 향후 연구를 기술한다.

2. 관련 연구

2.1 시나리오 기반 아키텍처 평가 기법

시나리오 기반 아키텍처 평가 기법은 소프트웨어 개발 중 아키텍처 설계 단계에서 작성된 아키텍처가 이해관계자의 품질 요구사항에 부합하는 품질을 제공하는 최종 제품으로 연계될 수 있는 가능성을 세부 구현에 들어가기 앞서 검토할 수 있는 수단을 제공한다. 아키텍처 설계의 품질 검증을 위해 널리 사용되는 ATAM, SAAM(Software Architecture Analysis Method)[14], CBAM(Cost Benefit Analysis Method)[15] 등이 시나리오 기반 아키텍처 평가 기법에 속한다. Utility Tree [3,8] 등의 기법을 통해 품질 요구사항으로부터 도출할 수 있는 시나리오는 품질 요구사항의 모호함을 제거하고 정량적인 검증이 가능하도록 하는 메커니즘을 제공한다. 시나리오를 매개체로 아키텍처 평가에 참여하는 아키텍트 및 이해관계자는 주어진 아키텍처가 시나리오에 기술된 품질 요구사항을 충족시킬 수 있는가에 대하여 토론하여 아키텍처를 검토한다. 그림 1은 시나리오 기반 아키텍처 평가 기법을 구성하는 활동을 보여준다.

시나리오 기반 아키텍처 평가는 일반적으로 다음과

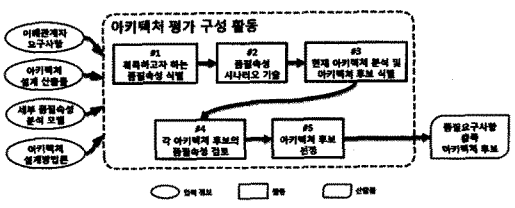


그림 1 시나리오 기반 아키텍처 평가

같은 단계로 구성된다: 1) 아키텍처 설계를 통해 획득하고자 하는 품질을 식별하고, 2) 품질 요구사항을 기술하는 품질속성 시나리오를 작성한 후, 3) 현재 아키텍처를 분석하여 시나리오 충족 여부를 검토한 뒤 시나리오를 충족시킬 수 있는 다른 아키텍처 후보를 식별하고, 4) 식별된 각 아키텍처 후보의 품질을 측정 및 검토하여, 5) 최종적으로 품질속성 시나리오를 가장 효과적으로 충족시키는 아키텍처 후보를 선정한다. 시나리오 기반 아키텍처 평가를 통해 아키텍트는 각 아키텍처 후보가 요구되는 품질을 지원하기 위해 채택한 주요 설계 의사결정을 발견하고, 어떤 설계 의사결정이 품질에 큰 영향을 주며, 해당 의사결정이 품질 요구사항을 충분히 지원하는지를 검토하고, 기존의 아키텍처 설계를 대체할 수 있는 새로운 후보들을 식별할 수 있다. 시나리오 기반 아키텍처 평가는 시스템 개발 초기에 아키텍처 설계 단계에서 수행되므로 최종 제품의 품질을 정량적으로 예측하기 위한 세부 설계 및 구현 산출물이 가용하지 않으며, 따라서 아키텍처의 예상 품질 평가는 아키텍트의 아키텍처에 대한 이해와 기존의 설계 지식에 기반한 예측을 통해 이루어진다. 세부 설계가 수행된 이후엔 측정하고자 하는 품질 별로 구체적인 아키텍처 평가 기법을 적용하여 보다 정확한 품질 예측을 얻을 수 있다[4,7]. 하지만 아키텍처 평가의 초점은 최종 제품의 품질 예측에 맞추어져 있으므로 아키텍처 후보 선정에 따른 비용 산정이 함께 고려되지 않거나 전문가의 경험 및 직관에 의존하여 예측되므로 체계적인 비용 고려가 이루어지지 않는다.

2.2 COCOMO II 비용 추정 모델

COCOMO II는 1981년 Boehm[13]에 의해 개발된 소프트웨어 비용 추정 모델로 소프트웨어 시스템의 코드 규모(size)를 비용 추정의 근간으로 하여 개발에 필요한 공수(effort)를 산출하기 위해 사용된다. 규모 기반으로 산출된 공수에는 시스템의 다양한 특성을 반영하는 비용요소가 적용되어 결과 값이 조정된다. COCOMO II에서 소프트웨어 개발 공수를 산출하기 위해 사용하는 계산식은 다음과 같다:

$$PM = A \times Size^{E} \times \prod_{i=1}^{17} EM_i \ ,$$

where A = 2.94 (COCOMO II. 2000)

위 식에서 PM은 소프트웨어 개발에 소요될 공수를 person-month 단위로 산출한 값이며 해당 값에 개발자의 평균 월 급여를 적용하여 소프트웨어 개발 비용을 산출할 수 있다. $Size$와 E는 각각 소프트웨어의 규모와 규모 측정 시 반영되어야 할 시스템 특성을 나타내는 인자인 Scale Factor들의 합을 의미한다. 그리고 EM_i는 소프트웨어 시스템의 특성을 반영하는 비용요소인 Effort Multiplier(EM)들의 값을 나타내며, 시스템의 제품으로써의 특성과 시스템이 기반하는 플랫폼의 특성, 개발에 참여하는 인력의 특성, 그리고 개발 프로젝트의 특성까지 네 종류의 시스템 특성을 반영하는 EM들로 구성된다. EM은 소프트웨어의 규모와 무관하게 개별적으로 적용되어 시스템의 비용을 조정하며, 시스템의 특성을 다양한 수준의 등급으로 나타내므로 아키텍처 설계의 선택에 따른 시스템 품질의 변화 또한 EM의 등급에 반영되어 시스템 비용 산출에 적용될 수 있다.

3. 아키텍처 후보 비용 추정

본 논문의 제안 기법은 기존 시나리오 기반 아키텍처 평가 기법을 보완하여 아키텍처 후보의 비용 정보를 도출하고 이를 사용하여 후보의 비용을 추정하는 방안을 제시한다. 그림 2는 그림 1에 도시된 아키텍처 평가 기법에 제안 기법을 구성하는 비용 추정 프로세스를 삽입하여 보완한 전체 프로세스를 도시한다. 아키텍처 후보 비용 추정은 비용 추정 프로세스의 Step 1 ~ Step 3을 따라 세 단계에 걸쳐 수행되며, 아키텍처 후보의 정보를 기술하는 기술 모델과 COCOMO II의 EM 정의에 기반한 비용요소 식별 및 평가 기준이 비용 추정을 위해 사용된다.

본 연구에서는 COCOMO II에서 정의된 EM 중 아키텍처 설계 변경에 따른 소프트웨어 시스템 특성의 변경을 반영할 수 있는 EM만을 선택적으로 적용한다. 따라서 비용 추정에 사용할 EM을 다음의 다섯 종류로 한정한다: 1) 시스템이 갖추어야 할 신뢰성을 반영하는 RELY(Required Software Reliability), 2) 소프트웨어의 내부적 복잡성을 반영하는 CPLX(Product Complexity), 3) 설계 및 구성요소의 재사용성을 반영하는

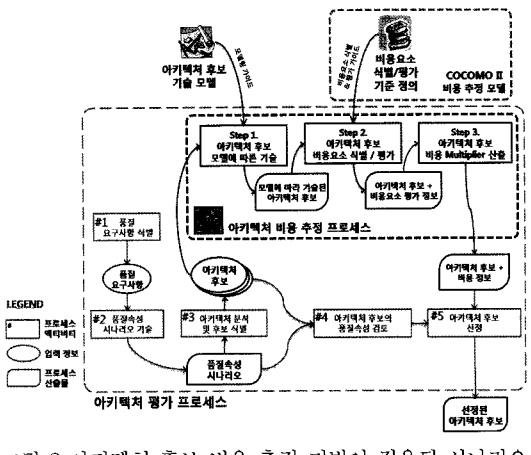


그림 2 아키텍처 후보 비용 추정 기법이 적용된 시나리오 기반 아키텍처 평가 프로세스

RUSE(Developed for Reusability), 4) 연산 처리시간 제한을 반영하는 TIME(Execution Time Constraint), 5) 저장공간 사용 제한을 반영하는 STOR(Main Storage Constraint).

비용 추정이 가능한 아키텍처 후보 또한 사용된 EM을 통해 반영 가능한 시스템 특성을 목표 품질로 하는 아키텍처 후보로 한정된다. 따라서 본 논문의 적용 범위는 성능(Performance), 시스템 가용성(Availability), 그리고 변경용이성(Modifiability)를 위한 아키텍처 후보의 비용 추정으로 한정된다.

3.1 절에서는 제안 기법이 적용되는 문맥과 더불어 기존 시나리오 기반 아키텍처 평가 기법의 산출물을 제안 기법을 위한 입력 정보로 사용하는 방안에 대하여 기술하며, 3.2 절에서는 아키텍처 비용 추정 프로세스를 세부적으로 설명한다.

3.1 기존 아키텍처 평가 산출물의 재사용

아키텍처 후보 비용 추정은 기존의 시나리오 기반 아키텍처 평가 기법을 통해 획득할 수 있는 아키텍처 후보 설계 정보에 기반한다. 아키텍처 후보 비용 추정을 위한 기반 정보는 각 후보의 설계 세부사항(컴포넌트와 커넥터 및 컴포넌트 토폴로지)과 후보 적용이 시스템 특성에 미치는 영향이며, 해당 정보는 시나리오 기반 아키텍처 평가의 수행 결과로 얻어지는 아키텍처 후보 기술 문서로부터 추출할 수 있다. ATAM과 같은 시나리오 기반 아키텍처 평가 기법은 평가 대상 아키텍처 후보의 설계에 채택된 의사결정을 검토하고 해당 후보를 소프트웨어에 적용 시 발생할 품질 및 특성 변화를 분석한다. 이 정보는 아키텍처 후보의 비용 추정 시 비용요소의 식별과 평가를 위한 기초 정보로 활용될 수 있다. 아키텍처 평가 기법의 산출물에 대한 의존성으로 인해 본 논문의 제안 기법은 기존 시나리오 기반 아키텍처 평가 기법과 병행하여 사용함을 전제하며, 대신 기존 아키텍처 평가 기법에 추가적인 아키텍처 설계 검토를 요구치 않고도 아키텍처 후보의 비용 정보를 도출할 수 있다. 그림 3은 시나리오 기반 아키텍처 평가 기법인 ATAM의 입력 정보와 참여자, 그리고 산출물을 보여준다.

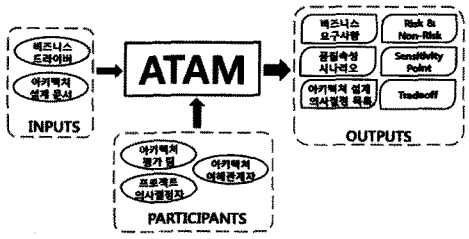


그림 3 시나리오 기반 아키텍처 평가 기법의 입력 정보, 참여자 및 산출물

3.2 비용 추정 프로세스

ATAM 등의 아키텍처 평가 기법을 통해 식별된 아키텍처 후보들의 목록과 각 후보의 적용에 따른 시스템 특성 영향을 기술한 문서가 비용 추정을 위한 입력 정보로 제공되었다면 각 아키텍처 후보의 비용요소를 분석하여 개발에 소요될 비용을 예측할 수 있다. 앞서 살펴본 그림 2의 아키텍처 비용 추정 프로세스를 구성하는 Step 1~Step 3을 통해 아키텍처 후보의 비용 배수(Cost Multiplier) 값이 산출된다. 이는 해당 아키텍처 후보를 제품에 적용할 경우 제품의 규모에 기반한 예측 비용을 조정하는데 적용되는 조정자(modifier)이며, 후보의 상대적 비용을 비교할 수 있는 기준이 된다. 비용 추정 프로세스를 구성하는 단계들을 3.2.1~3.2.3 절에서 상세히 기술한다.

3.2.1 아키텍처 후보 기술

아키텍처 비용 추정의 첫 단계는 아키텍처 후보들을 기술 모델의 형식에 맞추어 기술하는 단계이다. 본 단계를 구성하는 활동을 그림 4에서 확인할 수 있다.

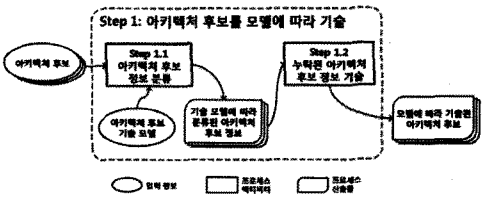


그림 4 비용 추정 Step 1: 아키텍처 후보 기술

아키텍처 후보 기술 단계는 아키텍처 후보 정보를 사용한 비용요소 식별 및 평가가 용이하도록 후보 정보를 전 처리하는 과정으로, 아키텍처 후보가 시스템 특성에 미치는 영향을 EM 정의에 따라 항목 별로 분류하는 것이 목적이다. 그림 5는 [3], [5]에서 정의한 아키텍처 후보 기술 모델과 COCOMO II[13]의 비용요소(cost driver)를 고려하여 아키텍처 후보를 기술하기 위한 모델을 보여준다. 본 논문은 그림 5와 같이 OMT notation[16]으로 나타낸 아키텍처 후보 기술 모델을 제시하여 아키텍처 후보의 정보를 템플릿의 형태로 체계화할 수 있도록 한다. 아키텍처 후보 기술 모델은 아키텍처 후보가 적용될 설계 상의 문맥을 기술하는 '배경/환경', 아키텍처 후보의 설계와 품질 지원 전략을 기술하는 '솔루션 설계', 그리고 시스템 특성에 미치는 영향을 기술하는 '적용 결과'로 구성된다. 적용결과 항목의 정보에 기반하여 아키텍트는 아키텍처 후보의 적용으로 인해 영향 받는 EM을 식별하고 해당 EM의 등급을 평가하는 근거를 구축한다. 만약 아키텍처 후보에 대하여 기술된 정보가 부족하여 비용요소를 위한 근거가 누락되었

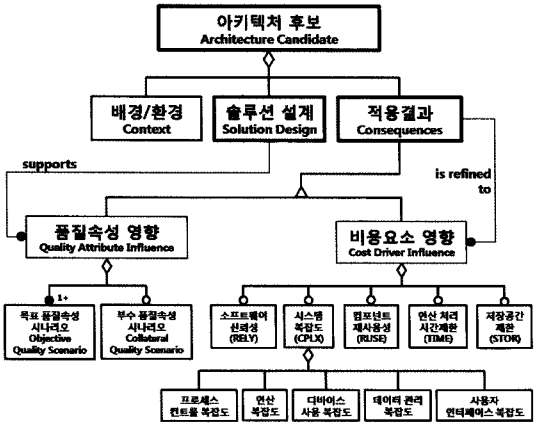


그림 5 비용요소를 고려한 아키텍처 후보 기술 모델

을 경우 아키텍트의 이해를 바탕으로 필요한 정보를 추가적으로 기술하도록 한다.

3.2.2 아키텍처 후보 비용요소 식별 및 평가

아키텍처 비용 추정의 두 번째 단계는 아키텍처 후보의 비용요소를 식별하고 등급을 평가하는 단계이다. 비용요소는 해당 아키텍처 후보가 소프트웨어 시스템의 아키텍처 설계에 적용될 때 품질속성의 변화와 함께 발생하는 시스템 특성의 변화를 반영한다. 비용요소 정보는 앞서 기술한 '적용 결과'항목의 정보를 바탕으로 도출될 수 있다. 비용요소 식별 및 평가 단계의 구성은 그림 6과 같다.

비용요소 영향의 식별과 등급 평가의 분리는 임의의 아키텍처 후보가 반드시 모든 비용요소의 등급에 영향을 주는 것이 아니기 때문이다. 아키텍처 후보의 적용으로 등급이 변경될 가능성이 있는 비용요소를 먼저 선별함으로써 비용요소 등급 평가의 오버헤드를 감소시킬 수 있다. 등급 값 변동이 발생할 것으로 예상되는 비용요소는 표 1의 EM 정의를 기반으로 등급이 평가된다[13]. 표 1에서 확인할 수 있는 바와 같이 RELY, RUSE, TIME, STOR는 단일 평가 기준으로 구성되어 있지만 CPLX는 5 개의 세부 항목으로 구성되어 있다. CPLX의 등급 평가는 각 세부 항목을 개별적으로 평가한 후 평균 등급을 세부 항목의 수로 나눈 값을 적용한다.

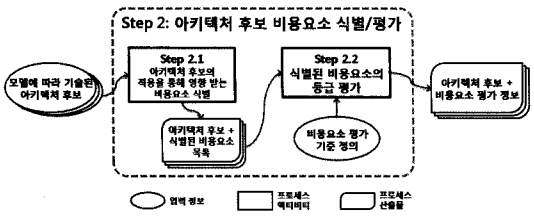


그림 6 비용 추정 Step 2: 비용요소 식별 및 평가

3.2.3 아키텍처 후보의 비용 배수 산출

아키텍처 후보의 비용요소 식별 및 평가가 완료된 후에는 아키텍처 비용 추정 프로세스의 세 번째 단계인 비용 배수 산출이 수행된다. 본 단계는 비용 추정을 위해 등급 평가가 완료된 비용요소들로부터 해당 아키텍처 후보의 비용 배수를 산출하며, 그림 7과 같은 활동으로 구성된다.

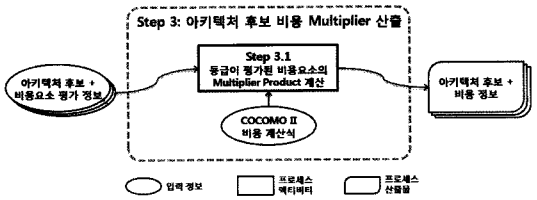


그림 7 비용 추정 Step 3: 비용 배수 산출

아키텍처 후보의 비용 배수 산출은 COCOMO II 모델의 비용 계산식에서 EMi 산출 부분을 차용하여 계산한다.

$$\text{CostMultiplier} = \prod_{i=1}^{5} EM_i \text{, where}$$

$$EM_1 = \text{RELY}, EM_2 = \text{CPLX}, EM_3 = \text{RUSE}, EM_4 = \text{TIME}, EM_5 = \text{STOR}$$

아키텍처 후보의 비용 배수는 비용 추정 프로세스의 최종 산출물로, 비용 배수 값이 아키텍처 후보의 비용 정보로 아키텍처 선정에 사용된다. 나아가 개발 프로젝트가 진행되며 세부 설계 및 구현이 완료되어 소프트웨어 시스템의 규모가 측정 가능해질 경우, 비용 배수는 해당 시스템의 EM값에 해당하므로 COCOMO II의 계산식을 사용한 구체적인 개발 비용을 산출하는데 사용될 수 있다.

4. 제안 기법의 검증

본 논문에서 제안하는 아키텍처 비용 추정 기법의 정확성을 검증하기 위해 여러 개의 아키텍처 설계가 수행되고 각 아키텍처 설계에 따른 구현이 완료된 임베디드 시스템을 대상으로 본 비용 추정 기법을 적용하였다. 구현된 시스템에 소요된 실제 공수를 측정하고 본 논문의 기법으로 추정된 아키텍처 후보의 비용을 비교함으로써 아키텍처 후보 비용 추정 기법의 비용 예측 정확성을 확인할 수 있다.

4.1 실험 적용 대상 및 실험 방법

RPS 시스템은 로봇 보안 시스템으로 지정된 구역 내를 왕복 주행하며 사전에 입력되지 않은 장애물을 탐지하여 침입자로 인지하고 무선 통신을 통해 사용자 콘솔에 경고 메시지를 전송하기 위한 시스템이다. RPS 시스

표 1 아키텍처 후보 비용요소 평가 기준

EM	평가 기준	등급
RELY	오류 발생 시 약간의 불편함 발생 - 운용 중 50% 이상의 가용성을 보장할 것	VeryLow x0.82
	오류 발생 시 간단히 복구 가능한 손실 발생 - 운용 중 70% 이상의 가용성을 보장할 것	Low x0.92
	오류 발생 시 일반적으로 복구 가능한 손실 발생 - 운용 중 85% 이상의 가용성을 보장할 것	Nominal x1.00
	오류 발생 시 대규모의 경제적 손실 발생 - 운용 중 95% 이상의 가용성을 보장할 것	High x1.10
	오류 발생 시 인명 피해 발생 가능 - 운용 중 99% 이상의 가용성을 보장할 것	VeryHigh x1.26
TIME	소프트웨어 실행 시 가용한 전체 연산 시간의 50% 이하를 점유해야 함	Nominal x1.00
	소프트웨어 실행 시 가용한 전체 연산 시간의 30% 이하를 점유해야 함	High x1.11
	소프트웨어 실행 시 가용한 전체 연산 시간의 15% 이하를 점유해야 함	VeryHigh x1.29
	소프트웨어 실행 시 가용한 전체 연산 시간의 5% 이하를 점유해야 함	ExtraHigh x1.63

EM			등급
CPLX: Control (1/2)	코드 네스팅	네스팅 없음	VeryLow
		간략한 네스팅	Low
		대부분 단순 네스팅	Nominal
		고도의 네스팅	High
		재귀적 프로그래밍	VeryHigh
		보다 복잡한 네스팅	ExtraHigh
	모듈 간 컨트롤	모듈 간 제어 없음	Low
		약간의 모듈 간 제어 사용	Nominal
		Task synchronization 수행	VeryHigh
		보다 복잡한 모듈 간 제어 사용	ExtraHigh
	리소스 스케줄링	리소스 스케줄링이 없음	Nominal
		고정된 우선순위의 인터럽트 처리	VeryHigh
		가변 우선순위 인터럽트 처리	ExtraHigh
CPLX: Compute (1/2)	표현식 복잡도	단순한 표현식	VeryLow
		일반적인 표현식	Low
		표준적인 수학 및 통계 루틴 사용	Nominal
		복잡한 수학 및 통계 루틴 사용	High
		보다 복잡한 표현식 사용	VeryHigh
	미분 방정식 복잡도	미분 방정식이 사용되지 않음	Nominal
		일반적인 미분 방정식 사용	High
		편미분 방정식 사용	VeryHigh
		보다 복잡한 미분 방정식 사용	ExtraHigh
CPLX: Data Mgmt	데이터 보존 매개체	메인 메모리에 저장되는 단순 배열	VeryLow
		단순한 파일 서브셋	Low
		다중 파일 입력 및 파일 출력	Nominal
		보다 복잡한 데이터 매개체	High
	데이터 관리	데이터 관리 필요성 없음	VeryLow
		데이터 구조 변경 없음	Low
		단순한 데이터 구조 변경	Nominal
		복잡한 데이터 구조 변경	High
		고도의 커플링이 수반된 객체 구조	ExtraHigh
	데이터베이스 쿼리	단순한 OOTS DB 쿼리	VeryLow
		약간 복잡한 OOTS DB 쿼리	Low
		복잡한 OOTS DB 쿼리	Nominal
		검색 최적화	VeryHigh
		보다 복잡한 DB 쿼리	ExtraHigh
	데이터 트리거	데이터 트리거 없음	Nominal
		데이터 스트림 컨텐츠에 의해 활성	High
		복잡한 데이터 트리거	VeryHigh
		보다 복잡한 데이터 트리거	ExtraHigh

EM	평가 기준	등급
RUSE	소프트웨어 구성요소 설계에 재사용성이 고려되지 않음, 재사용을 위한 문서가 개발 및 관리되지 않음	Low x0.95
	소프트웨어 설계가 단일 개발조직의 특정 프로젝트 내에서 재사용 가능한 수준으로 관리됨	Nominal x1.00
	소프트웨어 설계가 단일 개발조직 내 복수의 프로젝트에 걸쳐 재사용 가능한 수준으로 관리됨	High x1.07
	소프트웨어 설계가 복수의 개발조직에 걸친 단일 프로덕트 라인 내에 재사용 가능한 수준으로 관리됨	VeryHigh x1.15
	소프트웨어 설계가 복수의 개발조직에 걸친 복수의 프로덕트 라인 내에 재사용 가능한 수준으로 관리됨	ExtraHigh x1.24
STOR	소프트웨어 실행 시 가용한 주 저장 공간의 50% 이하를 점유해야 함	Nominal x1.00
	소프트웨어 실행 시 가용한 주 저장 공간의 30% 이하를 점유해야 함	High x1.05
	소프트웨어 실행 시 가용한 주 저장 공간의 15% 이하를 점유해야 함	VeryHigh x1.17
	소프트웨어 실행 시 가용한 주 저장 공간의 5% 이하를 점유해야 함	ExtraHigh x1.46

EM			등급
CPLX: Control (2/2)	Predicate 복잡도	Predicate 사용되지 않음	VeryLow
		단순한 Predicate 사용	Low
		복합적인 Predicate 사용	High
		보다 복잡한 Predicate 사용	VeryHigh
	프로시저 컨트롤	단순한 프로시저 호출 및 스크립트	VeryLow
		단순한 callback 및 메시지 전달	Nominal
		복잡한 callback	VeryHigh
		보다 복잡한 프로시저 제어	ExtraHigh
	분산처리	분산처리가 없음	Low
		미들웨어가 지원하는 분산처리	Nominal
		Homogeneous 프로세스 간 분산처리	High
		Heterogeneous 프로세스 간 분산처리	VeryHigh
		보다 복잡한 분산처리	ExtraHigh
CPLX: Compute (2/2)	수치해석 복잡도	수치해석이 사용되지 않음	Nominal
		기초적인 수치해석 사용	High
		복잡하지만 구조화된 분석	VeryHigh
		복잡하고 비구조적인 수치해석 사용	ExtraHigh
	행렬연산 복잡도	행렬연산이 없음	Low
		기초적인 행렬 및 벡터	Nominal
		Near-singular matrix	VeryHigh
		보다 복잡한 행렬연산	ExtraHigh
CPLX: Device	장치 연산 복잡도	단순한 포맷의 읽기/쓰기 명령	VeryLow
		GET/PUT 수준의 장치 입출력	Low
		디바이스 선택 및 에러 핸들링 필요	Nominal
		물리적 수준의 입출력 제어	High
		인터럽트 서비스 루틴 사용	VeryHigh
		마이크로 프로그램 사용	ExtraHigh
	임베디드 시스템 성능	임베디드 시스템 성능 요구사항 없음	Nominal
		Performance-intensive	VeryHigh
		Performance-critical	ExtraHigh
CPLX: User Interface	사용자 인터페이스 복잡도	단순한 입출력 또는 보고 생성	VeryLow
		단순한 GUI 빌더 사용	Low
		단순한 위젯 사용	Nominal
		위젯 셋 개발 및 확장	High
		보다 높은 복잡도의 UI	VeryHigh
	멀티미디어 사용	멀티미디어 없거나 아주 단순함	Nominal
		단순한 멀티미디어 사용	High
		다이나믹한 그래픽 사용	VeryHigh
		복잡한 멀티미디어 사용	ExtraHigh

※CPLX 세부항목의 등급 값은 모든 세부항목이 동일하게 VeryLow: x0.73, Low: x0.87, Nominal: x1.00, High: x1.17, VeryHigh: x1.34, ExtraHigh: x1.74

템의 품질 요구사항은 시스템 성능, 즉 가능한 빠르게 침입자 탐지 사실을 판단하고 경고 메시지를 사용자에게 전송하는 것이다. 이와 같은 성능 품질 요구사항을 목표로 2007년에 초기 아키텍처가 설계되어 시스템이 개발된 뒤 2008년에 동일한 품질 요구사항에 대하여 같은 방법론으로 아키텍처가 다시 설계되었고 새로운 아키텍처에 기반하여 시스템도 다시 구현되었다. 결과적으로 2007년과 2008년의 아키텍처는 같은 품질 요구사항에 대한 두 가지 아키텍처 후보로 간주될 수 있으며, 상이한 설계로 인하여 서로 다른 품질속성을 제공하며 개발 비용 또한 다르다.

본 논문의 비용 추정 기법이 정확할 경우, 두 아키텍처 후보의 비용 배수 간 비율이 실측 공수의 비율과 동일해야 한다. 만약 비용 배수의 비율과 실측 공수의 비율이 다를 경우 그 차이만큼 비용 추정 오차가 발생한 것으로 간주할 수 있으며, 아키텍처 선정을 위한 정보로 사용될 경우 선정 근거의 왜곡을 초래할 수 있다. 따라서 추정 비용의 비율과 실측 공수의 비율 간 일치하는 정도를 본 논문의 비용 예측 정확성 척도로 규정한다.

RPS 시스템의 구현 산출물로부터 실제 공수를 측정하기 위해 Class Point Analysis[17] 기법을 사용하였다. Class Point Analysis는 기능점수[18] 기법에 비하여 객체 지향 시스템의 공수 측정에 적합한 소프트웨어 규모 측정 기법으로 객체 지향 방법론을 사용하여 설계된 RPS의 규모 측정에 적합하다. 특히 두 아키텍처 후보의 기능 요구사항이 동일하므로 기능점수로서는 반영할 수 없는 아키텍처 설계 상의 차이를 규모 산정에 반영할 수 있기 때문에 Class Point Analysis는 두 RPS 아키텍처 후보의 정확한 공수 측정이 가능하다.

4.2 측정 결과

본 논문의 아키텍처 후보 비용 추정 기법으로 RPS 시스템의 2007년 아키텍처와 2008년 아키텍처 각각의 비용을 추정하였다. 표 2~3에서 2007년 아키텍처의 정보와 비용요소 평가 결과 및 최종 비용 배수를 확인할 수 있다. 2007년 RPS 아키텍처는 주행 시스템과 탐지 시스템 및 경보 시스템을 독자적인 쓰레드로 실행하고 비동기적으로 운용되는 특징을 갖는다. 따라서 2007년 아키텍처는 센서 및 데이터에 대해 여러 쓰레드가 동시 접근하는 상황에 대하여 고려해야 하며, 시스템의 복잡도가 상대적으로 높아질 수 있다.

표 4~5는 2008년 아키텍처 정보 및 비용요소 평가 결과를 나타낸다. 2008년 RPS 아키텍처는 중앙 통제를 적용한 순차적 프로세싱을 통해 주행과 탐지 및 경보 발송이 처리되는 특징을 갖는다. 쓰레드의 수가 적고 센서 등의 자원 공유가 없으므로 상대적으로 낮은 복잡도의 시스템 설계로 구성된다.

표 2 2007년 RPS 아키텍처

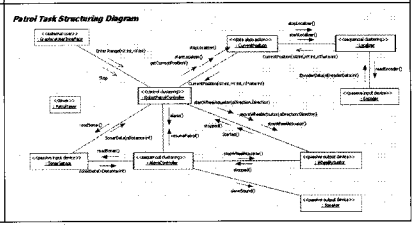
아키텍처 후보: RPS 아키텍처 2007	
배경/환경	RPS가 순찰 중 침입자를 발견했을 때 경보를 울리기까지 소요되는 시간이 최소화되어야 한다.
솔루션 설계	침입자 탐지 알고리즘을 독자적인 coordinator와 timer를 갖는 thread로 운용하고, 중앙 컨트롤과 침입자 탐지 쓰레드 간 Sonar sensor와 Actuator를 비 동기화된 방식으로 공유한다(asynchronous contention).
지원 품질속성	성능(Performance)
목표 시나리오	침입자 탐지 시 경보 응답시간을 최소화한다.
부수 시나리오	N/A
측정 척도	응답 시간(Response Time)
아키텍처 다이어그램	

표 3 2007년 RPS 아키텍처 비용 추정 결과

아키텍처 후보: RPS 아키텍처 2007			
EM 평가	EM 이름	등급(Rating)	배수(Multiplier)
	RELY	Nominal	x1.00
	CPLX:Control	Nominal	x1.00
	CPLX:Comp	**Very Low**	**x0.73**
	CPLX:Device	Nominal	x1.00
	CPLX:Data	**Low**	**x0.87**
	CPLX:UI	Nominal	x1.00
	RUSE	Nominal	x1.00
	TIME	Nominal	x1.00
	STOR	Nominal	x1.00
비용 배수		**x0.920**	

4.3 결과 분석

표 6은 본 비용 추정 기법으로 예측한 비용의 비율과 Class Point를 사용하여 측정한 두 아키텍처 후보의 공수의 비율을 비교한 결과를 나타낸다.

본 논문의 제안 기법을 통해 예측한 아키텍처 후보 비용의 비율 간 차이는 0.059로 실측 공수의 비율 간 차이인 0.76보다 약 22% 적은 차이를 보였으며 이 차이가 비용 예측 정확성 77.63%를 도출하였다. 이는 개발 프로세스 초기의 아키텍처 설계 단계에서 수집할 수 있는 시스템 정보만을 사용한 초기 예측인 만큼 만족할 수 있는 수준의 정확성이라 할 수 있다. Boehm의 비용 추정 정확성에 대한 연구[2,13]에 따르면 개발 프로젝트 중 요구사항 명세 및 아키텍처 설계 단계에서 추정 오차가 70~80% 내인 비용 추정 기법이라면 다음 단계로 진행하기 위한 유용한 가이드라인이 되는 정보를 제공한다고 간주할 수 있으므로 본 논문의 제안 기법은 이

표 4 2008년 RPS 아키텍처

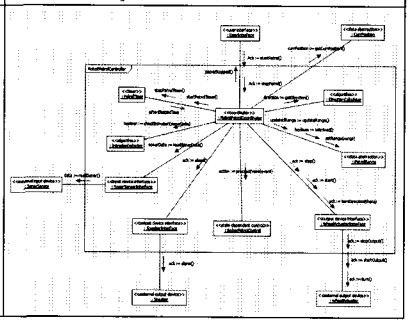
아키텍처 후보: RPS 아키텍처 2008	
배경/환경	RPS가 순찰 중 침입자를 발견했을 때 경보를 울리기까지 소요되는 시간이 최소화되어야 한다.
솔루션 설계	침입자 탐지 알고리즘을 중앙 컨트롤과 timer를 공유하는 sequential 프로세스로 운용하고, Sonar sensor와 Actuator에 대한 접근은 동기화된 방식으로 공유한다(synchronized arbitration).
지원 품질속성	성능(Performance)
목표 시나리오	침입자 탐지 시 경보 응답시간을 최소화한다.
부수 시나리오	N/A
측정 척도	응답 시간(Response Time)
아키텍처 다이어그램	

표 5 2008년 RPS 아키텍처 비용 추정 결과

아키텍처 후보: RPS 아키텍처 2008			
EM 평가	EM 이름	등급(Rating)	배수(Multiplier)
	RELY	Nominal	x1.0
	CPLX:Control	**Low**	**x0.87**
	CPLX:Comp	**Very Low**	**x0.73**
	CPLX:Device	**Low**	**x0.87**
	CPLX:Data	**Low**	**x0.87**
	CPLX:UI	Nominal	x1.00
	RUSE	Nominal	x1.00
	TIME	Nominal	x1.00
	STOR	Nominal	x1.00
비용 배수		**x0.868**	

표 6 비용 예측 결과와 실측 공수의 비교

	RPS 2007	RPS 2008	비율	정확성
비용 배수	x0.920	x0.868	1 : 1.059	77.63%
Class Point 측정 공수	98 UCP	91 UCP	1 : 1.076	100%

기준에 따를 경우 올바른 아키텍처 선정을 위한 유용한 비용 정보를 제공할 수 있는 정확성을 보인다고 할 수 있다. 하지만 시스템의 규모가 커지고 복잡해질 수록 한정된 수의 비용요소로 시스템의 특성 변화를 올바르게 반영하기 힘들게 되기 때문에 본 기법의 정확성은 보다 다양한 규모의 프로젝트에 대하여 추가적인 검증을 요한다.

5. 결론 및 향후 연구

소프트웨어의 발전과 함께 현대 산업의 소프트웨어 의존성이 커가고 있다. 격화되는 소프트웨어 시장 경쟁 속에서 개발조직이 생존하기 위해서는 다양한 품질 요구사항을 충족시키는 동시에 비용에 기반한 세심한 아키텍처 설계를 획득하여 빠르게 변하는 개발환경에 적응할 수 있는 역량을 갖추어야 한다.

본 논문에서는 소프트웨어 아키텍처의 품질을 검토하고 선정하는 기존의 아키텍처 평가 기법에 비용 추정 프로세스를 보완하여 품질과 비용에 대한 체계적인 검토가 함께 수행될 수 있도록 하는 기법을 제안하였다. 본 비용 추정 기법을 기반으로 아키텍트는 품질 획득을 위해 설계된 아키텍처의 상대적 비용을 산출하고 이를 활용하여 비용을 고려한 아키텍처 후보 선정을 수행할 수 있으며, 결과적으로 이해관계자의 품질 요구사항과 개발 조직의 ROI 요구사항을 함께 만족시키는 최적의 아키텍처 설계를 지원할 수 있다.

향후 비용 추정 정확도에 대한 추가적인 검증을 수행하여 규모가 큰 시스템 아키텍처를 대상으로도 높은 수준의 비용 추정 정확성을 보일 수 있도록 기법을 정제할 필요가 있다.

참 고 문 헌

[1] A. Davis, *Software Requirements: Objects, Functions and States*, Prentice Hall, 1993.

[2] B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.

[3] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice 2nd Edition*, Addison Wesley, 2003.

[4] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2002.

[5] C. Hofmeister, R. Nord, and D. Soni, *Applied Software Architecture*, Addison Wesley, 2000.

[6] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.

[7] M. Barbaci, M. Klein, T. Longstaff, and C. Weinstock, *Quality Attributes*, Technical Report CMU/SEI-95/TR-21, 1995.

[8] R. Kazman, M. Klein, M. Barbaci, T. Longstaff, H. Lipson, and J. Carriere, *The Architecture Tradeoff Analysis Method*, Proceedings of the 4th ICECCS, pp.68-78, 1998.

[9] L. Bass, R. Nord, W. Wood, and D. Zubrow, *Risk Themes Discovered Through Architecture Evaluations*, Technical Report CMU/SEI-2006-TR-012, 2006.

[10] M. Ali Babar and I. Gorton, *Comparison of Scenario-Based Software Architecture Evaluation*

Methods, Proceedings of the 11th Asia-Pacific Software Engineering Conference, pp.600-607, 2004.

[11] M. Svahnberg, C. Wohlin, L. Lundberg, and M. Mattsson, *A Quality-Driven Decision-Support Method for Identifying Software Architecture Candidates*, International Journal of Software Engineering and Knowledge Engineering, Vol.13, Issue 5, pp.547-573, 2003.

[12] M. Ionita, D. Hammer, and H. Obbink, *Scenario-Based Software Architecture Evaluation Methods: An Overview*, Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering, 2002.

[13] B. Boehm et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.

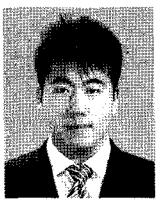
[14] R. Kazman, Len Bass, G. Abowd, and M. Webb, *SAAM: A Method for Analyzing the Properties Software Architectures*, Proceedings of the 16th International Conference on Software Engineering, pp.81-90, 1994.

[15] R. Kazman, J. Asundi and M. Klein, *Quantifying the Costs and Benefits of Architectural Decisions*, Proceedings of the 23rd ICSE 2001, pp.297-306.

[16] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-oriented Modeling and Design*, Prentice Hall, 1991.

[17] G. Costagliola and G. Tortora, *Class Point: An Approach for the Size Evaluation of Object-Oriented Systems*, IEEE Transactions on Software Engineering, Vol.31, Issue 1, pp.52-74, 2005.

[18] IFPUG, *Function Point Counting Practice Manual Release 4.1.1*, International Function Point User Group, 2000.



조 시 호

2007년 서강대학교 컴퓨터공학과(공학사). 2009년 서강대학교 컴퓨터공학과(공학석사). 관심분야는 요구공학, 소프트웨어 아키텍처, 소프트웨어 비용, 애자일 프랙티스

이 준 하

2007년 8월 단국대학교 컴퓨터과학과(공학사). 2008년 2월~서강대학교 컴퓨터공학과 석사과정 관심분야는 요구공학, 소프트웨어 아키텍처, 소프트웨어 프로세스

박 수 용

1986년 2월 서강대학교 컴퓨터공학과(공학사). 1988년 8월 Florida State University, Computer and Information Science(공학석사). 1995년 5월 George Mason University, Information Technology(공학 박사). 관심분야는 요구공학, 동적 아키텍처