韓國 컴퓨터情報學會 論文誌
第10卷 第6號, 2005. 12.

2005-10-6-1-17

소프트웨어 아키텍처의 성숙 평가 모델에 관한 연구

김 경 희*

A study on the Maturity Appraisal Model of Software Architecture

Kyung-Hee Kim*

요 약

소프트웨어 아키텍처는 소프트웨어 집중적인 시스템의 가장 주요한 부분으로, 아키텍처 평가는 시스템에서 매우 중요한 과정이고 소프트웨어 재앙을 예방할 수 있는 가장 저렴한 방법이다. 본 논문은 성숙모델을 통하여 아키텍처를 평가하는 새로운 방법을 소개한다. 기존의 성숙모델들은 소프트웨어나 소프트웨어 프로세스에 관한 것으로 지금까지 빠른 속도로 발전되어 왔다. 본 논문에서는 이러한 기법들을 소프트웨어 아키텍처에 적용한 SAMM(Software Architecture Maturity appraisal Model)을 제안한다. SAMM은 여섯 등급으로 구성되어 있다. 제안한 성숙 모델 SAMM은 시스템 개선을 관리하는 발전된 아키텍처를 구성하고 소프트웨어 아키텍처 설계를 개선하기 위해 필수적으로 요구되는 아키텍처 요구사항들을 명세한다. 또한, 아키텍처 설계를 위한 노력을 줄이고, 질 높은 아키텍처를 구성할 수 있는 지침을 마련해주며, 아키텍처를 평가하여 등급을 매긴다.

Abstract

The software architecture is an essential part of a software-intensive system. In addition, the architecture evaluation is a very important process and a cheap way to avoid a software disaster. This article introduces a new method to evaluate architecture by maturity levels. Maturity Models which are about software and software process, have gained wide scale acceptance over the last decade. We are applying these techniques to the software architecture and propose SAMM(Software Architecture Maturity appraisal Model). SAMM consists of six-levels. We expect that our maturity model SAMM describe the requirements that any architecture must have in order to improve its software architecture design and constitute a proven architecture within which to manage the improvement efforts. We can reduce our effort to design architecture, have a guideline to construct the high quality architecture with SAMM, and evaluate architecture and make architecture level.

▸ Keyword : 소프트웨어 아키텍처(Software Architecture), 성숙모델(Maturity Model), SAMM(Software Architecture Maturity appraisal Model)

• 제1저자 : 김경희
• 접수일 : 2005.10.28, 심사완료일 : 2005.12.15
* 백석대학 디지털정보학부 교수

Ⅰ. 서 론

현대 사회에서 소프트웨어는 의료, 서비스, 제조, 교육, 기계, 전기전자, 항공, 항만 등 사회전반에 걸쳐 산업 전 분야에서 중대한 영향을 미치고 있다. 소프트웨어가 산업전반에 끼치는 영향력과 중요성이 인식되면서, 좋은 소프트웨어를 개발하기 위한 노력과 소프트웨어의 오류를 줄이기 위한 노력이 극대화 되고 있는 것이 현실이다. 특히 소프트웨어의 오류는 사회적으로 심각한 문제를 발생시키는 것 뿐 아니라 인류의 재앙으로까지 발전할 수 있다. 현대 사회에서 소프트웨어가 미치는 영향력을 감안할 때, 소프트웨어가 사용 목적에 적합하게 개발되고 원하는 결과물을 쉽게 산출하도록 설계하며, 오류가 발생하지 않도록 유지하는 일은 매우 중요하다.

소프트웨어 시스템의 토대는 아키텍처이다[1]. 아키텍처는 시스템의 모든 질적 속성들을 허용하거나 막는다[2]. 좋은 아키텍처는 소프트웨어를 개발할 때 성공의 첫걸음이지만, 잘못된 아키텍처는 큰 재해를 불러일으킬 수 있다. 따라서, 아키텍처의 평가는 매우 중요한 과정이고 소프트웨어 재해를 피할 수 있는 가장 저렴한 방법이다[3].

아키텍처 평가모델(architecture appraisal model)을 통하여 아키텍처가 개발하려고 하는 소프트웨어에 적합하게 잘 설계되었는지 여부를 밝히고, 성공적인 결과물과 산출물을 순조롭게 제공할 수 있는지의 여부를 밝혀내는 일은 좋은 소프트웨어를 개발하는데 필수조건이지만, 아키텍처를 평가하는 것은 간단하고 쉬운 문제가 아니다.

기존의 연구들은 아키텍처 평가모델들을 통해 아키텍처를 평가하기 위한 과정을 정의하고, 그 과정에서의 산출물들을 명세하는 것에 관한 연구이거나, 여러 단계의 과정을 통하여 산출물을 분석하고 토론하여 책임자가 대략적으로 아키텍처를 평가하는 방법에 관한 연구가 대부분이었다.

본 연구에서는 소프트웨어 프로세스의 성숙도(maturity level)를 측정하기 위한 성숙모델을 아키텍처 평가 모델에 접목하여 아키텍처의 성숙도에 따라 등급을 매기고 아키텍처를 평가하는 방법을 제시한다.

기존의 소프트웨어 프로세스(software process) 성숙모델에 대한 연구로는 People CMM(People Capability Maturity Model)[4][5], 시스템 공학 CMM(System Engineering Capability Maturity Model)[6], 소프트웨어 산출물 CMM과 CMMI(Capability Maturity Model Integration)[7] 등이 있다. 이러한 성숙모델들은 소프트웨어 프로세스를 평가하기 위한 방법으로 아키텍처 평가 방법과는 달리 연구되었다.

아키텍처 평가 모델에 대한 연구로는 ATAM(Architecture Tradeoff Analysis Method)[2], CBAM(Cost Benefit Analysis Method)[3]등이 있다. 이러한 아키텍처 평가 방법들은 아키텍처를 평가하기 위한 평가 과정을 정의하고, 각 단계를 거친 후에 직관적으로 아키텍처의 총책임자가 평가를 내리는 방식을 취한다. 따라서, 이러한 아키텍처 평가 모델들은 전문가들의 주관적인 평가에 의존함으로 객관적인 평가가 이루어지지 않고 있다.

객관적인 평가의 예는 소프트웨어나 소프트웨어 프로세스를 평가하는 모델로 CMMI[7]나 People CMM[4][5]에서 찾아 볼 수 있다. 이러한 모델들은 객관적인 등급별 속성을 제시함으로 제시된 조건에 부합될 때 해당하는 등급을 부여하여 소프트웨어나 소프트웨어의 프로세스를 평가한다.

본 논문에서는 기존의 아키텍처 평가 모델에 소프트웨어 프로세스 성숙 모델을 접목하여, 소프트웨어 아키텍처 성숙 평가 모델을 제시한다. 소프트웨어 아키텍처가 갖추어야 하는 속성들을 정리하여 성숙 등급에 따른 속성들을 정의·제시하고, 성숙 등급에서의 제시된 속성을 만족하는지의 여부를 가지고 아키텍처를 평가하여 등급을 매기는 새로운 아키텍처 평가 방법인 SAMM(Software Architecture Maturity appraisal Model)을 제안한다. SAMM은 모두 여섯 등급으로 나뉜다. SAMM은 아키텍처 설계가 지속적으로 발전되기 위해서 갖추어야하는 요구사항과, 발전적인 아키텍처로 증명된 아키텍처에서 필수적으로 요구되는 구성요소들을 등급별로 명세한다. 또한, 질 좋은 아키텍처를 만드는 지침을 제공함으로써 아키텍처 설계에 드는 노력을 절감시키고, 아키텍처의 등급을 매김으로써 아키텍처의 평가를 객관화 시킨다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어, 2장에서는 관련연구로 기존의 아키텍처 평가모델인 ATAM과 ATAM을 기반으로 하여 비용 모델을 세운 CBAM을 소개하고, 기존의 성숙도 측정모델로 소프트웨어 프로세스 성숙모델 CMMI와 노동집단의 업무 성숙도를 평가하는 People CMM을 설명한다. 3장에서는 본 논문에서 제안한 아키텍처 성숙 평가 모델인 SAMM을 설명하며, 마지막으로 4장에서 결론과 향후 연구과제들을 살펴본다.

II. 관련연구

2.1 기존의 아키텍처 평가모델

2.1.1 ATAM

ATAM은 다양한 질적 속성과 장단점 특성을 발생시키는 다양한 속성들에 민감한 수많은 아키텍처들에 초점을 맞춘다. 또한, ATAM은 질적 속성의 목표에 권위를 부여하여 아키텍처의 확정된 책임자들을 활용한다[2].

ATAM은 다음의 〈표 1〉과 같은 단계를 거친다. 그러나 ATAM에서의 절차는 엄격한 폭포수형 프로세스(Waterfall Process)는 아니고, 필요에 따라서 전단계로 회귀(regress), 다음단계로 건너뛰기, 단계 간에 반복 등이 가능하다[8].

표 4. ATAM의 7단계[8]
Table 1. ATAM process has 7 Steps

단계	내용
1. ATAM 소개	평가 선임자가 참석자들에게 평가 방법 설명, 기대치 정함, 질의응답 시간을 갖음
2. 사업 소개	프로젝트 대표자(프로젝트 관리자 혹은 고객)가 사업 목표와 가장 중요한 아키텍처적인 요구사항을 설명
3. 아키텍처 소개	아키텍트가 사업 목표를 달성하기 위한 아키텍처를 설명
4. 아키텍처 접근법 식별	아키텍트가 아키텍처적인 접근법들 찾음(분석은 하지 않음)
5. 품질속성 만들기	시스템이 획득해야 하는 품질속성을 도출, 정의
6. 아키텍처 접근법 분석	5단계에서 식별한 높은 우선순위의 시나리오를 기반으로, 이것에 적합한 아키텍처적 접근법을 도출 및 분석
7. 토론과 시나리오 우선순위 매김	전체 책임자 그룹으로부터 시나리오를 도출, 투표에 의해서 이것의 우선순위 매김
8. 아키텍처 접근법 분석	단계7에서 도출한 높은 우선순위의 시나리오를 중심으로 단계6의 반복
9. 결과보고	평가팀은 ATAM 평가를 통해 수집한 정보를 관련 책임자에게 보고

ATAM 분석을 수행함으로써 우리는 다음과 같은 장점들을 갖게 된다. 즉, 요구사항들이 명료화되고 아키텍처 문서들과 아키텍처적 결정에서 사용되는 문서의 기초가 마련되며, 초기 문제가 확인되고 책임자들의 의사소통이 증가된다. 그 중에서도 ATAM을 사용하는 가장 큰 이점은 소프트웨어 아키텍처가 발전된다는 것이다. ATAM은 특별하게 정의된 질적 속성들을 평가한다.

ATAM의 단점은 ATAM에서 산출된 분석·평가 점수가 정확하지 않다는 것이다. 그 점수는 아키텍처에서 높은 위험영역을 발견하기 위한 것이고 아키텍처적인 매개변수들 사이의 연관성의 경향을 찾는 것이다. 이러한 영역들은 위험을 줄이기 위한 것에 집중되어 있다. 예를 들면, 미래 설계, 미래 분석, 표본 등이 그것이다.

반면, 본 논문에서 제시하는 SAMM방법은 아키텍처의 등급을 매기기 위한 등급별 속성을 제시하고 아키텍처가 어느 등급에 속하는지를 확인 할 수 있도록 함으로써 아키텍처를 객관적인 등급을 통하여 평가하도록 한다.

2.1.2 CBAM

CBAM은 결정분석구조와 결합된 반복적인 도출 과정이다. ATAM을 기초하여 시작되었고, ATAM이 산출하는 결과물에서 시작한다[3].

다음의 〈표 2〉는 CBAM의 과정을 간략하게 설명한 것이다.

표 5. CBAM의 9단계
Table 2. CBAM process has 9 Steps

단계	내용
1. 시나리오 수집	시나리오의 3분의 1을 선택하기 위해 우선순위 매김
2. 시나리오 정비	현재 시나리오의 비람직한 경우들을 질적 속성 응답 단계 최상, 최하로 나눔
3. 시나리오 우선순위 매김	우선순위를 매겨 시나리오의 절반 정도 삭제
4. 유틸리티 분배	각 시나리오의 현재와 바람직한 레벨을 위해 유틸리티를 분배
5. 아키텍처 전략 발전	질적 속성 응답 레벨을 결정하고 시나리오를 위한 아키텍처 전략을 발전시킴
6. 예상되는 활용 값 결정	보간법을 사용하여 아키텍처 전략의 예상되는 활용 값을 결정
7. 전체적인 이점 계산	아키텍처 전략에서 얻어진 전체적인 이점을 계산
8. 아키텍처 전략 선택	비용조건의 ROI(Return On Investment)를 기반으로 한 아키텍처 전략을 선택
9. 결과 결정	직관적으로 결과를 결정

CBAM의 과정은 책임자들이 발전된 상태에서의 명확한 시나리오를 만들도록 강요한다. 이 과정은 아키텍처 설계에서 중요한 이점이 되는 시나리오와 요구사항들을 명확하게 하는 결과를 낳는다. 그러나, CBAM은 복잡한 소프트웨어 시스템의 발전을 위한 많은 장점을 가지고 있음에도 불구하고, ATAM과 유사하게 정확하고 객관적인 분석과 평가보다 주관적이고 직관적인 결과를 산출한다. 또한, 아키텍처 설계의 질을 높이기 위한 점차적인 전략과 갖추어야 하는 조건이 제시되지 않는다.

SAMM은 소프트웨어 아키텍처의 객관적인 등급 속성을 제시하고 해당하는 조건을 갖춘 아키텍처를 구분하여 등급을 매김으로써 객관적인 아키텍처의 등급을 제시할 뿐 아니라 소프트웨어 아키텍처의 등급별 조건을 제시함으로써 아키텍처들이 전체적인 아키텍처의 질을 높이기 위한 속성들을 준비할 수 있도록 한다.

2.2 기존의 성숙도 측정모델

2.2.1 CMMI

CMMI는 미 국방성의 후원으로 SEI(Software Engineering Institute)에서 개발된 CMM의 전환된 형태로, CMM 인증 이후 CMMI로 변경되어 국내의 여러 기업들이 사용하는 소프트웨어 성숙도 측정 방법이다. 세계의 여러 나라에서 CMMI의 인증심사를 실시하고 있고 소프트웨어 성숙도를 측정하기 위한 표준으로 떠오르고 있다.

CMMI는 소프트웨어 개발에 대한 성숙도를 측정하는 기준을 제시한다. 다음의 〈표 3〉은 CMMI를 단계별로 간략하게 보인 것이다[7]

표 6. CMMI의 단계
Table 3. The Stage of CMMI

등급	내용
1 초기 (Initial)	뛰어난 개발자에 의해 소프트웨어가 개발됨, 그 개발자가 없으면 개발 못함
2 관리 (Managed)	계획을 세워 계획에 따라 소프트웨어 개발에 관련된 실행을 준행
3 정의 (Defined)	소프트웨어 개발 과정의 표준이 나옴, 표준에 따라 모든 개발 과정을 진행
4 양적관리 (Quantitatively Managed)	과거의 자료를 측정하고 축적하여 새로운 개발에 활용함, 자료의 축적으로 지속적인 소프트웨어 개발에 발전이 이루어짐
5 최적화 (Optimizing)	원인 분석과 해결점을 찾음으로 조직적인 혁신과 발전이 지속됨

SAMM은 소프트웨어 프로세스의 성숙도 측정에 사용되었던 CMMI를 아키텍처에 적용한다. CMMI가 소프트웨어 개발에 대한 성숙치를 단계별로 나누고, 그 단계에서 갖추어야 하는 사항들을 제시한 것을 아키텍처에 적용하여, SAMM은 아키텍처의 성숙도를 나누고 등급별로 갖추어야 하는 아키텍처의 등급별 속성을 제시하며 속성조건을 갖추었는지의 여부에 따라 등급을 나눈다.

2.2.2 People CMM

People CMM[4][5]은 노동자와 노동 집단의 업무성숙도를 나타내는 여러 가지 모델들[9][10] 중의 하나로 업무수행자가 업무를 발전적으로 처리 하도록 돕고 그 처리과정이 효과적으로 진행되도록 한다. People CMM은 5단계로 성숙도를 나누고, 각 단계마다 처리 영역을 두어 영역별 목표와 이를 달성하기 위한 내용으로 구성되어 있다. 다음의 〈표 4〉는 People CMM의 단계를 보인 것이다.

표 7. People CMM의 단계
Table 4. The Stage of People CMM

등급	내용
1 초기 (Initial)	업무수행은 업무 영향력의 분석 없이 진행된다.
2 관리 (Managed)	관리자는 업무 수행자들을 관리하고 개발시키는 책임을 갖는다.
3 정의 (Defined)	업무집단과 업무자의 능력을 발전시키고, 사업 전략을 나눈다.
4 예측됨 (Predictable)	업무 수행자들에게 수행 능력을 주고 통합하며, 양적으로 일을 분배한다.
5 최적화 (Optimizing)	지속적으로 발전된다.

People CMM은 인력의 성숙도를 측정하기 위한 모델로 관리자가 채택하여 사용할 수 있다. 성숙모델은 여러 분야에서 활용되고 있으며, 실제로 산업 현장에서 성숙도를 측정하고 성숙레벨을 높이기 위한 기준으로 사용되고 있다.

SAMM은 이러한 성숙모델을 소프트웨어 아키텍처의 성숙도를 평가하는데 적용하는 시도이다.

III. 제안한 아키텍처 평가모델

3.1 SAMM

기존의 아키텍처 평가모델은 질 좋은 아키텍처를 작성하기 위한 작업 과정을 나열하고, 각 단계를 거친 아키텍처를 책임자가 주관적인 판단에 의해 평가를 수행함으로써 객관적인 평가가 이루어지지 않는다는 단점이 있다.

SAMM은 이러한 기존의 아키텍처 평가모델의 단점을 극복하여 객관적인 평가가 이루어지도록 하고 아키텍처를 등급으로 나누며 질 좋은 아키텍처가 갖추어야 하는 조건들을 등급별로 제시하는 새로운 아키텍처 평가 방법이다. SAMM 방법은 소프트웨어 프로세스의 성숙도 측정에 사용되었던 CMMI를 아키텍처에 적용한 것으로, 기존의 아키텍처 평가모델인 ATAM과 CBAM에서 보인 아키텍처 개발 과정과 조건들을 적용하여 제안되었다.

SAMM은 아키텍처가 갖추어야 하는 조건들을 성숙도에 따라 제시하고 제시된 조건들의 만족여부에 따라 아키텍처를 6 등급으로 나눔으로써 아키텍처의 객관적인 평가가 수행되도록 한다.

3.2 SAMM의 6등급

SAMM은 아키텍처가 발전하는 것을 단계별로 정의하고 등급화 한다. 이는 모두 여섯 등급으로 나뉘며, 각 단계들은 다음 단계에서 아키텍처의 효과적인 설계를 위해 필요한 기초를 제공한다. 다음의 〈표 5〉는 SAMM의 여섯 등급을 간략하게 설명한 것이다.

표 8. SAMM의 6등급
Table 5. The six levels of SAMM

등급	내용
0 아키텍처 설계 없음	소프트웨어 아키텍처 설계가 없고, 언급할만한 소프트웨어 아키텍처가 없음
1 비공식적 아키텍처 설계 존재	소프트웨어 아키텍처 설계가 비공식적인 형태로 진행 중
2 아키텍처 표준에 만족	5개의 소프트웨어 아키텍처 뷰들이 아키텍처 표준에 만족되나 통합되지는 않았음
3 아키텍처 콤포넌트들이 통합됨	아키텍처 콤포넌트들이 잘 통합되고 통신이 잘 됨
4 소프트웨어 콤포넌트들 관리용이	소프트웨어 아키텍처와 소프트웨어 콤포넌트를 유연하게 수정하기 용이
5 아키텍처의 지속적 발전 가능	측정 가능하고 소프트웨어 아키텍처의 지속적인 발전이 가능

여섯 등급은 아키텍처 뷰들의 공통점을 고려하고, 공통 소프트웨어 아키텍처 뷰들은 다음을 포함한다[1].

- 함수뷰(Functional views)
- 동시뷰(Concurrency views)
- 코드뷰(Code views)
- 개발뷰(Development views)
- 물리뷰(Physical views)

SAMM은 각 등급별로 일반적인 목표인 GG(Generic Goals)를 제시하고 GG를 이루기 위한 GP(Generic Practices)를 명시한다. 또한, SAMM의 각 등급에서 중점을 두는 처리 영역인 PA(Process Area)를 제시하고, PA에서 다루는 특정 목표인 SG(Special Goals)를 보이며, SG를 달성하기 위한 특수 실행들인 SP(Specific Practices)를 제시한다.

다음의 (그림 1)은 SAMM에서 요소들 사이의 관계를 보인 것이다.

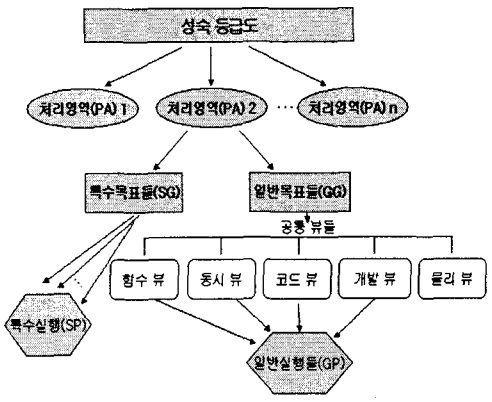

그림 1. SAMM 요소들의 관계도
Fig 1. The relationship diagram of SAMM's elements

(그림 1)에서 보이는 것과 같이 성숙 등급의 SG와 SP들은 각 PA별로 구성된다. PA는 아키텍처의 어떤 영역을 개선하기 위해 중요하게 고려되는 목표들의 집합을 만족하는 그 영역에 연관된 실행들의 집합이다. 또한, 각 PA는 자신들의 특수 목표인 SG들을 갖는데, SG는 해당되는 PA를 만족하기 위해 구현되어야 하는 목표들을 설명하는 유일한 특징들을 나타낸다.

각 SG는 자신을 위한 특수 실행을 나타내는 SP를 갖는다. SP는 자신과 연관되고 완수된 중요하게 생각되는 행동들을 나타낸다.

다음의 〈표 6〉은 SAMM의 각 등급에서 만족되어야 하는 GG와 GG를 위한 GP, 그리고 중점적으로 다루는 PA와 그에 따른 SG와 SP를 보인 것이다.

표 9. SAMM의 6 등급에서의 목표와 실행들
Table 6. The goals and practices of six levels on the SAMM

등급	GG	GP	PA	SG	SP
5	GG4 GG5	GP4.1G P4.2 GP5.1	측정 발전 (Measure Development)	SG5.1	SP5.1.1 SP5.1.2 SP5.1.3
				SG5.2	SP5.2.1 SP5.2.2 SP5.2.3 SP5.2.4
			조직 혁신과 전개 (Organization Innovation & Deployment)	SG5.3	SP5.3.1 SP5.3.2 SP5.3.3
				SG5.4	SP5.4.1 SP5.4.2 SP5.4.3 SP5.4.4
4	GG3 GG4	GP3.1 GP4.1 GP4.2	구성 관리 (Module Management)	SG4.1	SP4.1.1 SP4.1.2 SP4.1.3
				SG4.2	SP4.2.1 SP4.2.2 SP4.2.3 SP4.2.4
			증명 (Verification)	SG4.3	SP4.3.1 SP4.3.2 SP4.3.3
				SG4.4	SP4.4.1 SP4.4.2 SP4.4.3
				SG4.5	SP4.5.1 SP4.5.2 SP4.5.3
3	GG2 GG3	GP2.1 GP2.2 GP2.3 GP3.1	뷰들을 통합 (Integrated Views)	SG3.1	SP3.1.1 SP3.1.2 SP3.1.3
				SG3.2	SP3.2.1 SP3.2.2
			조직적 구성들 (Organizational Modules)	SG3.3	SP3.3.1 SP3.3.2
				SG3.4	SP3.4.1 SP3.4.2 SP3.4.3
			산출물 통합 (Product Integration)	SG3.5	SP3.5.1 SP3.5.2 SP3.5.3
				SG3.6	SP3.6.1 SP3.6.2 SP3.6.3
				SG3.7	SP3.7.1 SP3.7.2
			측정관리 (Measure Management)	SG3.8	SP3.8.1 SP3.8.2
				SG3.9	SP3.9.1
2	GG1 GG2	GP1.1 GP1.2 GP1.3 GP2.1 GP2.2 GP2.3	요구발전 (Requirements Development)	SG2.1	SP2.1.1 SP2.1.2 SP2.1.3 SP2.1.4
				SG2.2	SP2.2.1 SP2.2.2 SP2.2.3
			뷰들을 만족시킴 (Satisfy Views)	SG2.3	SP2.3.1 SP2.3.2
				SG2.4	SP2.4.1 SP2.4.2 SP2.4.3
				SG2.5	SP2.5.1 SP2.5.2
				SG2.6	SP2.6.1 SP2.6.2
				SG2.7	SP2.7.1 SP2.7.2 SP2.7.3
			형상관리 (Configuration Management)	SG2.8	SP2.8.1 SP2.8.2
				SG2.9	SP2.9.1 SP2.9.2 SP2.9.3
			프로젝트 모니터링과 감독 (Project Monitoring and Control)	SG2.10	SP1.10.1 SP1.10.2 SP2.10.3
				SG2.11	SP2.11.1 SP2.11.2 SP2.11.3
1	GG1	GP1.1 GP1.2 GP1.3	요구관리 (Requirement Management)	SG1.1	SP1.1.1 SP1.1.2 SP1.1.3
			프로젝트 계획 (Project Planning)	SG1.2	SP1.2.1 SP1.2.2
				SG1.3	SP1.3.1 SP1.3.2 SP1.3.3 SP1.3.4 SP1.3.5 SP1.3.6
				SG1.4	SP1.4.1 SP1.4.2 SP1.4.3
0					

3.3 SAMM의 등급별 요구조건

SAMM은 각 등급에 따라 갖추어야 하는 조건을 명세한다. 명세된 조건에 부합될 때 해당하는 소프트웨어 아키텍처는 명세된 등급으로 인정된다. SAMM에서 등급별 조건은 각 등급에서 추구하는 일반 목표(GG)와 특수 목표(SG), 그리고 그 목표들을 이루기 위한 사항일반실행(GP)과 특수실행(SP)들로 구성된다.

다음은 〈표 6〉에서 보인 등급별 GG와 GP, 그리고 각 PA가 갖는 SG와 SP들을 SAMM의 등급에 따라 설명한 것이다.

등급 0은 어떤 조건도 갖추지 않은 상태로 제시된 조건이 없다. 따라서, GG, GP, PA, SG, 그리고 SP가 없다.

등급 1은 조건이 있는 등급의 첫 번째로 아키텍처에 기본적인 조건들이 제시된다. 다음의 〈표 7〉은 등급 1에서 갖추어야 하는 GG와 그에 따른 GP를 보인 것이다.

표 10. SAMM에서 등급 1의 GP
Table 7. The GP of SAMM in level 1

GG	GP	GP의 내용
GG1 : 아키텍처는 비공식적인 소프트웨어 아키텍처 설계로 시작됨	GP1.1	요구사항들 수집
	GP1.2	프로젝트 계획을 세움
	GP1.3	소프트웨어 아키텍처에 요구사항들을 할당

다음의 〈표 8〉은 PA별로 등급 1에서 갖추어야 하는 SG와 그에 따른 SP를 보인 것이다. 아키텍처가 〈표 7〉과 〈표 8〉에서 제시하는 GP와 SP의 조건들을 모두 갖추었을 때 우리는 그 아키텍처가 등급 1에 해당된다고 할 수 있다.

표 11. SAMM에서 등급 1의 SP
Table 8. The SP of SAMM in level 1

PA	SG	SP	SP의 내용
요구 관리	SG1.1 요구들을 관리	SP1.1.1	이해할만한 요구사항들 획득
		SP1.1.2	요구들의 수행을 얻음
		SP1.1.3	아키텍처와 요구사항들 사이의 불일치를 확인
프로젝트 계획	SG1.2 계획 수행	SP1.2.1	아키텍처에 영향을 미치는 계획 점검
		SP1.2.2	계획을 수행
	SG1.3	SP1.3.1	아키텍처를 위한 자원과 스케줄 세움
	아키텍처 계획 세움	SP1.3.2	아키텍처 위험 확인
		SP1.3.3	책임자들 개입을 위한 계획
		SP1.3.4	지식과 기술의 필요성에 대해 계획
		SP1.3.5	아키텍처의 우선순위 분석
		SP1.3.6	아키텍처 문서를 운영
	SG1.4 계획을 평가	SP1.4.1	아키텍처의 상수들 평가
		SP1.4.2	아키텍처 생명주기 정의
		SP1.4.3	노력과 비용 평가 정의

등급 2는 아키텍처가 구조적이다. 또한, 등급 1이 갖춘 조건을 모두 만족하고 그 이외에 다음의 조건을 더 만족하여야 한다. 다음의 〈표 9〉는 등급 2에서 갖추어야 하는 GG와 그에 따른 GP를 보인 것이다.

표 12. SAMM에서 등급 2의 GP
Table 9. The GP of SAMM in level 2

GG	GP	GP의 내용
GG2 : 구조적인 소프트웨어 아키텍처 설계로 시작	GP2.1	구조적인 소프트웨어 아키텍처와 명세를 만듦
	GP2.2	구조적 소프트웨어 아키텍처의 결과를 측정
	GP2.3	사람들을 훈련시킴

다음의 〈표 10〉은 PA별로 등급 2에서 갖추어야 하는 SG와 그에 따른 SP를 보인 것이다. 아키텍처가 〈표 9〉와 〈표 10〉에서 제시하는 GP와 SP의 조건들을 모두 갖추었을 때 우리는 그 아키텍처가 등급 2에 해당된다고 할 수 있다.

표 13. SAMM에서 등급 2의 SP
Table 10. The SP of SAMM in level 2

PA	SG	SP	SP의 내용
요구 발전	SG2.1 고객요구들을 발전	SP2.1.1	고객요구 발전
		SP2.1.2	필요성들 도출
		SP2.1.3	요구사항에 아키텍처 모듈 할당
		SP2.1.4	요구들의 형상 관리
	SG2.2 요구들을 분석하고 검증	SP2.2.1	요구분석
		SP2.2.2	운영적 개념과 방법제공
		SP2.2.3	요구에 맞는 방법 검증
뷰들을 만족	SG2.3 함수뷰 만족	SP2.3.1	함수뷰 아키텍처를 제공
		SP2.3.2	관리할 수 있는 함수단위로 모듈들을 분산(11)

시킴	SG2.4 동시뷰 만족	SP2.4.1	동시뷰 아키텍처를 제공
		SP2.4.2	실행상태를 갖는 요소들로 구성된 모델 정의 (11)
		SP2.4.3	다른 뷰들과 매핑
	SG2.5 코드뷰 만족	SP2.5.1	코드뷰 아키텍처를 제공
		SP2.5.2	다른 뷰들과 매핑
	SG2.6 개발뷰 만족	SP2.6.1	개발뷰 아키텍처를 제공
		SP2.6.2	다른 뷰들과 매핑
	SG2.7 물리뷰 만족	SP2.7.1	물리뷰 아키텍처를 제공
		SP2.7.2	소프트웨어 환경구조로 소프트웨어 구조를 도식함(11)
		SP2.7.3	다른 뷰들과 매핑
형상 관리	SG2.8 기준 마련	SP2.8.1	형상관리 시스템을 세움
		SP2.8.2	형상 항목을 확인
	SG2.9 수정을 추적·조정	SP2.9.1	수정 요청을 추적
		SP2.9.2	형상 항목 조정
		SP2.9.3	형상 항목의 수정을 문서화
프로젝트 모니터링과 감독	SG2.10 소프트웨어 아키텍처설계 감시	SP2.10.1	아키텍처 위험을 감시
		SP2.10.2	책임자 투입을 감시
		SP2.10.3	모듈아키텍처의 참여 감시
	SG2.11 고객요구들을 발전	SP2.11.1	수행의 결과를 관리
		SP2.11.2	아키텍처의 예측된 결과 제공
		SP2.11.3	아키텍처의 정확한 결과 분석

등급 3은 아키텍처의 구성요소들이 모두 통합적이다. 또한, 등급 2가 갖춘 조건을 모두 만족하고 그 이외에 다음의 조건을 더 만족하여야 한다. 다음의 〈표 11〉은 등급 3에서 갖추어야 하는 GG와 그에 따른 GP를 보인 것이고, 〈표 12〉는 PA별로 등급 3에서 갖추어야 하는 SG와 그에 따른 SP를 보인 것이다. 아키텍처가 〈표 11〉과 〈표 12〉에서 제시하는 GP와 SP의 조건들을 모두 갖추었을 때 우리는 그 아키텍처가 등급 3에 해당된다고 할 수 있다.

표 14. SAMM에서 등급 3의 GP
Table 11. The GP of SAMM in level 3

GG	GP	GP의 내용
GG3 : 잘 통합된 아키텍처로 시작	GP3.1	통합된 아키텍처를 세움

표 15. SAMM에서 등급 3의 SP
Table 12. The SP of SAMM in level 3

PA	SG	SP	SP의 내용
뷰들을 통합	SG3.1 아키텍처 뷰들의 통합을 준비	SP3.1.1	다섯 개 뷰들의 통합 결정
		SP3.1.2	다섯 개의 뷰들의 통합을 위한 환경 만듦
		SP3.1.3	통합의 패턴 만듦
	SG3.2 아키텍처 콤포넌트들 모아 결과 산출	SP3.2.1	아키텍처 결과를 모음
		SP3.2.2	아키텍처 콤포넌트의 모아진 산출물을 평가
조직적 구성들	SG3.3 통합을 위해 뷰들을 관리	SP3.3.1	다섯 개의 뷰들의 균형을 위한 기법 세움
		SP3.3.2	다섯 개의 뷰들을 위한 통합규칙 정함
	SG3.4 통합된 다섯 개의 뷰들에 아키텍처 제공	SP3.4.1	통합된 다섯 개 뷰들에 어울리는 콤포넌트 정의
		SP3.4.2	인터페이스 관리
		SP3.4.3	다섯 개 뷰들의 의사소통 관리
산출물 통합	SG3.5 산출물 통합을 준비	SP3.5.1	통합순서 결정
		SP3.5.2	산출물 통합 환경 만듦
		SP3.5.3	다섯 개 뷰들을 가진 산출물 통합하는 아키텍처를 위한 기준 만듦
	SG3.6 인터페이스 호환성 확보	SP3.6.1	완전함을 위한 인터페이스 명세를 검토
		SP3.6.2	인터페이스 관리
		SP3.6.3	다섯 개 뷰들의 의사소통 관리
	SG3.7 뷰들의 산출 콤포넌트들 모음	SP3.7.1	뷰들의 산출 콤포넌트들 수집
		SP3.7.2	다섯 개 뷰들의 수집된 산출 콤포넌트들 평가
측정 관리	SG3.8 측정 준비	SP3.8.1	아키텍처의 통합 콤포넌트 위한 측정순서 결정
		SP3.8.2	통합된 아키텍처 측정 세움
	SG3.9 통합된 아키텍처 콤포넌트 측정	SP3.9.1	아키텍처의 통합된 콤포넌트 측정

등급 4는 아키텍처 구성요소들을 관리하기 용이하도록 구성되어 있다. 또한, 등급 3이 갖춘 조건을 모두 만족하고 그 이외에 다음의 조건을 더 만족하여야 한다. 다음의 〈표 13〉은 등급 4에서 갖추어야 하는 GG와 그에 따른 GP를 보인 것이다.

표 16. SAMM에서 등급 4의 GP
Table 13. The GP of SAMM in level 4

GG	GP	GP의 내용
GG4 : 잘 통합된 아키텍처로 시작	GP4.1	통합된 아키텍처를 세움
	GP4.2	관리하는 것을 조정

다음의 〈표 14〉는 PA별로 등급 4에서 갖추어야 하는 SG와 그에 따른 SP를 보인 것이다. 아키텍처가 〈표 13〉와 〈표 14〉에서 제시하는 GP와 SP의 조건들을 모두 갖추었을 때 우리는 그 아키텍처가 등급 4에 해당된다고 할 수 있다.

표 17. SAMM에서 등급 4의 SP
Table 14. The SP of SAMM in level 4

PA	SG	SP	SP의 내용
구성 요소 관리	SG4.1 아키텍처 설계관리	SP4.1.1	아키텍처 성능 관리
		SP4.1.2	구조적 아키텍처 조립
		SP4.1.3	콤포넌트 통계적 관리
	SG4.2 아키텍처설계 통계적 관리	SP4.2.1	분석 기법과 측정을 선택
		SP4.2.2	모듈 관리를 위한 이해 변수에 통계적 기법 적용
		SP4.2.3	선택된 콤포넌트의 성능감시
		SP4.2.4	통계적 모듈 관리 자료 기록
증명	SG4.3 검증 준비	SP4.3.1	검증을 위한 아키텍처 콤포넌트 선택
		SP4.3.2	검증 환경 만듦
		SP4.3.3	검증 순서와 영역 정함
	SG4.4 peer review 수행	SP4.4.1	peer review 위한 소프트웨어 아키텍처 준비
		SP4.4.2	아키텍처의 peer review 수행
		SP4.4.3	peer review 결과 분석
	SG4.5 선택 콤포넌트 아키텍처 검증	SP4.5.1	아키텍처의 검증 수행
		SP4.5.2	검증 결과 분석과 아키텍처에 정확한 처리를 규정
		SP4.5.3	검증 순서와 결과 기록

등급 5는 지속적으로 발전할 수 있도록 구성된 아키텍처이다. 또한, 등급 4의 모든 조건을 모두 만족하고 그 이외에 다음의 조건을 더 만족하여야 한다. 다음의 〈표 15〉는 등급 5에서 갖추어야 하는 GG와 그에 따른 GP를 보인 것이다.

표 18. SAMM에서 등급 5의 GP
Table 15. The GP of SAMM in level 5

GG	GP	GP의 내용
GG5: 아키텍처는 측정된 소프트웨어 아키텍처로 시작됨	GP5.1	아키텍처의 결과를 측정하고 아키텍처의 구성요소를 교정

다음의 〈표 16〉은 PA별로 등급 5에서 갖추어야 하는 SG와 그에 따른 SP를 보인 것이다. 아키텍처가 〈표 15〉와 〈표 16〉에서 제시하는 GP와 SP의 조건들을 모두 갖추었을 때 우리는 그 아키텍처가 등급 5에 해당된다고 할 수 있다.

표 19. SAMM에서 등급 5의 SP
Table 16. The SP of SAMM in level 5

PA	SG	SP	SP의 내용
측정 발전	SG5.1 측정과 행동 결과를 조절	SP5.1.1	콤포넌트 측정을 세움
		SP5.1.2	아키텍처에 적용할 측정 명세
		SP5.1.3	결과 측정과 행동 교정
	SG5.2 결과 측정	SP5.2.1	측정 결과 수집
		SP5.2.2	측정 결과 분석
		SP5.2.3	연관된 행동과 결과 기록
		SP5.2.4	결과들과 의사소통
조직 혁신과 전개	SG5.3 발전의 기반조직을 제공	SP5.3.1	조직의 목표를 세움
		SP5.3.2	통합된 작업 환경 조성
		SP5.3.3	통계적 콤포넌트 관리
	SG5.4 아키텍처의 변경을 관리	SP5.4.1	변경 인식
		SP5.4.2	변경 관리 기법 세움
		SP5.4.3	변경에 의한 콤포넌트 영향 분석
		SP5.4.4	변경 관리

Ⅳ. 결론과 향후 연구과제

본 논문은 아키텍처의 발전시키기 위한 관리능력과 개선된 조직의 아키텍처 설계를 향상시키기 위한 지침을 제공하

고 아키텍처의 성숙도에 따라 아키텍처를 등급별로 평가하는 SAMM을 제안한다. 기존의 평가 모델인 ATAM과 CBAM이 아키텍처의 구성을 위한 과정을 제시하고 주관적으로 아키텍처를 평가하였던 것에 반해, SAMM은 아키텍처를 등급별로 나누고 등급별 조건을 제시하며 그 조건에 부합되는지의 여부에 따라 등급을 부여함으로써 객관적인 평가가 이루어지도록 한다. 또한, 효과적으로 아키텍처가 설계되었는지를 시험하도록 돕고 개선된 아키텍처를 구현하며 아키텍처 향상을 지원한다.

향후 연구과제로는, 성숙모델의 정량적 평가를 위한 GP와 SP의 가치 값을 지정하는 것과 이를 통해 SAMM의 각 레벨별로 성숙도의 범위 값을 얻는 것에 관한 연구가 요구된다. 또한, 정량적인 성숙도 값을 통해 용이하게 소프트웨어 아키텍처의 성숙도를 측정하는 도구에 관한 연구가 요구된다. 이러한 도구와 각 레벨간의 레벨 값 범위를 통해, 명시된 수치를 가지고 소프트웨어 아키텍처의 레벨을 측정함으로 소프트웨어 아키텍처의 정량적 평가가 용이하게 진행될 수 있을 것이다.

참고문헌

[1] Paul Clements, Rick Kazman, Mark Klein, "Evaluating Software Architecture Methods and Case Studies," Addison Wesley, pp. 19-42, 2002.

[2] Mary Shaw, David Garlan, "Software Architecture perspectives on an emerging discipline," Prentice Hall, 1996.

[3] Len Bass, Paul Clements, Rick Kazman, "Software Architecture in Practice Second Edition," Addison Wesley, pp.307-325, 2003.

[4] Bill Curtis, William E. Hefley, Sally Miller, "Overview of the People Capability Maturity Model," SEI Carnegie Mellon University, 1995.

[5] Sally A. Miller, "People Capability Maturity Model Version 2," Carnegie Mellon University, 2003.

[6] Jack Cooper, Matthew Fisher, "Software Acquisition Capapbility Maturity Model," Version 1.03, Technical report, CMU/SEI-2002-TR-010, ESC-TR-2002-010, 2002.

[7] CMMI Product Team, "Capability Maturity Model Integration Version 1.1," Carnegie Mellon University, 2002.

[8] Paul Clements, Rick Kazman, Mark Klein, "Evaluating Software Architectures: Methods and Case Studies," Addison Wesley, 2001.

[9] 노형진, "품질경영을 위한 소집단활동의 활성화 지원 모델 개발에 관한 연구," 한국컴퓨터정보학회 논문지, vol. 5 no. 1, 2000.

[10] 유은숙, 도경화, 정기원, "정부문서유통시스템의 효율성 측면에 대한 성과평가모델 분석," 한국컴퓨터정보학회 논문지, vol. 9, no. 1, pp. 111-121, 2004.

[11] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord Judith Stafford, "Documenting Software Architectures - Views and Beyond," Addison Wesley, pp.35-184, 2003.

저 자 소 개

김 경 희

1999년 8월 숙명여자대학교 전산학박사

1999년 3월~현재 백석대학 조교수

〈관심분야〉 소프트웨어 아키텍처, 소프트웨어 테스팅, 무선 프로그램, 디지털 콘텐츠, e-learning