http://dx.doi.org/10.3745/KTSDE.2013.2.10.691

An Evaluation of Software Development Methodology Applicability at Medium and Small Business through AHP

Kyueok Kim† · Haeyoung Yoo††

ABSTRACT

To develop of a new software, software development methodology is offering the most efficient development methods and management methods. But, it require a lot of time, cost and software engineering specialist. For this reason, we are awaking to the need of it, but it has its problem that is only applied to large-scale software. In this paper, we suggest optimized software development methodology that you can apply to a lot well-used small software development methodology in present, and we prove it through AHP(Analytic Hierarchy Process). This helps small and business not to introduce specialist can efficiently develop and manage software.

Keywords : Software Engineering, Methodology, Software Development, Small Software, AHP

AHP를 통해 소규모 소프트웨어 개발을 위한 소프트웨어 개발방법론 적합도 평가

김 규 억† · 유 해 영††

요 약

소프트웨어 개발방법론은 새로운 소프트웨어를 개발하는데 가장 효율적인 개발 방법과 관리 방법 등을 제시하고 있다. 그러나 소프트웨어 개발방법론은 많은 비용과 시간을 필요로 하고 있으며, 소프트웨어공학 전문가를 필요로 하고 있다. 이러한 이유로 필요성은 인정받고 있으나 대규모 소프트웨어에서만 적용되는 문제점들을 안고 있다. 본 논문에서는 현재 많이 사용되고 있는 소프트웨어 개발방법론을 전문가 없이 소규모 소프트웨어를 개발하는데 적용할 수 있도록 반드시 필요한 단위 일(Task)을 선별하고 대안으로 정의하여 AHP(Analytic Hierarchy Process) 기법을 통해 대안에 대한 적합도를 평가하였다. 이를 통해 중소기업에서도 평가된 소프트웨어 개발방법론을 사용하여 소프트웨어를 효율적으로 개발하고 관리할 수 있다.

키워드 : 소프트웨어공학, 방법론, 소프트웨어 개발, 소규모 소프트웨어, AHP

1. 서 론

산업의 발달에 따라 소프트웨어의 복잡성과 필요성은 점점 높아져 가고 있으며, 이렇게 다양한 소프트웨어 개발을 위해 소프트웨어 개발방법론(Software Development Methodology)은 반드시 필요한 도구로 인식되고 있다. 또한 소프트웨어 개발방법론은 소프트웨어를 개발하는 것에만 그치지 않고 소프트웨어 자산화를 통한 재사용과 체계적인 소프트웨어 관리를 위해서도 많은 개발방법론 연구가 진행되고 있다[1]. 소프트웨어 개발방법론은 소프트웨어를 개발하는데 필요한 프로세스, 모델링, 코딩방법, 테스트 등과 같이 개발 팀원이 해야 할 일들을 정의하고 있으며, 또한 개발팀이 개발방법론에 따라 개발하고 있는지를 확인할 수 있는 품질 관리, 프로젝트 관리 등을 함께 정의하고 있다[2].

이와 같이 소프트웨어 개발방법론은 소프트웨어를 개발하는데 필요한 많은 도구들을 포함하고 있고, 소프트웨어를 개발하는 많은 개발 팀들이 적용하려 하고 있다. 하지만, 소프트웨어 개발방법론은 주로 대규모 소프트웨어를 구축할 수 있는 대기업 위주로 사용되고 있다. 소프트웨어 개발방법론은 소프트웨어를 개발하는 방법과 방향을 제시하지만 적용에 따른 비용과 시간이 반드시 필요하기 때문이다. 이러한 이유로 대규모 소프트웨어를 개발하는 대기업 위주로 소프트웨어 발전시켜 나가기 시작했고, 최근에는 해당 기업의 경험과 프로젝트의 특성을 고려한 독창적인 소프트웨어 개발방법론을 별도로 만들어 사용하고 있다[3].

비용과 시간, 그리고 전문인력 부족 때문에 소프트웨어 개발방법론을 고려하지 않던 중소 규모의 소프트웨어를 개발하는 곳에서도 최근에는 소프트웨어 개발방법론에 대한 필요성을 인정하고 있다[4]. 본 논문에서는 이를 증명하기

† 준 회 원 : 단국대학교 소프트웨어학과 박사과정
†† 정 회 원 : 단국대학교 소프트웨어학과 정교수
논문접수 : 2013년 8월 12일
수 정 일 : 1차 2013년 9월 9일, 2차 2013년 9월 23일
심사완료 : 2013년 9월 30일
* Corresponding Author : Kyueok Kim(kyueok.kim@gmail.com)

위해 중소기업의 소프트웨어 개발방법론 필요성에 대한 설문조사와 인터뷰를 실시 하였고, 도입을 하지 않은 이유에 대해서도 분석하였다. 설문과 인터뷰에 참여한 중소기업에서는 소프트웨어 개발방법론을 도입하기 가장 어려운 이유로 소프트웨어 규모에 비해 프로세스와 산출물이 너무 많고 복잡하다는 결과가 도출되었다. 또한 인터뷰를 통해 소프트웨어공학 전문가들은 표준 소프트웨어 개발방법론을 개발팀의 특성에 맞게 수정하여 사용하도록 가이드하고 있으나 중소기업에서는 전문인력보다는 중소규모의 소프트웨어를 개발하는데 반드시 필요한 부분만 포함하는 최적화된 소프트웨어 개발방법론만을 요구하고 있었다.

본 논문에서는 2장에서 많이 사용되고 있는 표준적인 소프트웨어 개발방법론과 본 논문에서 적합성을 검증할 AHP에 대해 살펴보고, 3장에서는 소규모 소프트웨어를 개발할 때 필요한 소프트웨어 개발방법론의 대안을 제시한 후 AHP를 통해 대안들의 적합도를 평가하고자 한다. 이러한 평가 결과를 바탕으로 기존의 소프트웨어 개발방법론에서 가장 필요한 부분만을 찾아내어 소규모 소프트웨어 개발을 위한 최적의 소프트웨어 개발방법론을 제시하고자 한다. 마지막으로 4장에서는 결론과 향후 연구에 대하여 서술하였다.

2. 관련 연구

2.1 표준적인 소프트웨어 개발방법론

소프트웨어 개발방법론은 1960년대 이후부터 나타나기 시작했다. 당시에는 프로그램 로직 중심으로 필요한 기능과 데이터의 흐름을 정의하는 정도의 방법론을 정의했다[5]. 그 이후 80년대 중반 이후부터는 데이터의 중요성이 강조되면서 정보공학 방법론이 나타나기 시작했다. 대규모의 시스템을 구축하는 경우에 많이 사용되었으며, 현재까지도 정보공학 방법론을 활용하여 소프트웨어를 개발하는 많이 활용하고 있다[6]. Table 1은 60년대 이후 현재까지 가장 많이 사용되던 소프트웨어 개발방법론을 정리한 것이다.

90년대 중반 이후부터는 객체지향 개념을 적용한 객체지향 방법론이 나타났다. 지금도 잘 알려진 UML 기반의 다이어그램을 사용하기 시작했다. C++, JAVA 등 객체지향 기반의 프로그래밍 언어로 개발되는 개발 팀에서 거의 사용되었다[7]. 그 이후 소프트웨어 재사용의 중요성이 알려진 90년대 후반부터 컴포넌트 기반의 방법론이 나타났으며, 현재도 많이 사용되고 있는 CBD(Component Based Development: 컴포넌트 기반) 개발방법론이 대부분의 소프트웨어 개발 프로젝트에서 사용되었다[8, 9].

Table 1에서 보는 바와 같이, 소프트웨어 개발방법론은 프로그램 로직, 데이터, 객체, 그리고 컴포넌트와 같이 중심이 되는 것이 무엇이냐에 따라 발전되었다. 다시 말해, 개발팀이 코딩만 고려하면 되었던 것을 객체나 컴포넌트까지도 고려하다 보니 객체나 컴포넌트를 분석하고 설계하는 부분까지 방법론에 포함을 시키면서 방법론의 규모가 점점 커지게 되었다. Table 1과 같이 소프트웨어 개발방법론이 점점 복잡해지고 작성해야 하는 산출물도 많아지면서 개발 팀원들이 해야 되는 일도 점차 늘어났다. 이렇게 규모가 커진 소프트웨어 개발방법론은 다양한 형태로 개발되는 대규모 시스템에는 효율적으로 적용할 수 있었다. 하지만 소규모 소프트웨어를 개발하는 곳은 많은 프로세스와 산출물 때문에 적용하기가 매우 어려웠다[10]. 삼성SDS의 개발방법론인 이노베이터(Innovator)에서는 표준화된 프로세스와 산출물을 개발하고자 하는 소프트웨어의 규모에 맞춰 다시 정의할 것을 권장하고 있다[11].

2.2 AHP (Analytic Hierarchy Process)

AHP(Analytic Hierarchy Process: 계층화 분석법)는 한 명 혹은 여러 명의 의사결정자가 참여하는 의사결정 문제를 여러 개의 평가 기준(Evaluation Standard)과 대안을 계층적인 구조로 파악하여 최적의 대안(Alternative)을 선택하는 방법이다. 평가 기준이 다수 존재하는 경우, 계층적 구조화(Hierarchical Structuring)하고 주요 평가 기준과 세부 평가 기준들로 구분한다. 이렇게 구분된 평가 기준들을 쌍대 비교(Pairwise Comparison)을 통해 중요도를 산출한다. AHP는 자료가 충분히 완비되지 않았거나 시간적으로 촉박한 상황, 또는 의견 대립이 있는 상황에서 여러 가지 대안들을 순위화하고, 그 가중치를 비율 척도(Ratio Scale)로 도출하는 방법을 제시하고, 비교 수행자가 얼마나 일관성을 가지고 결과를 적었는지를 일관성 지수(Consistency Index: C.I), 일관성 비율(Consistency Ratio)로 확인할 수 있다. 이러한 단계의 결과를 바탕으로 대안을 선택한다. Fig. 1은 AHP를 진행하는

Table 1. Kinds of Software Development Methodology

Categories	Structural Methodology	Information Engineering Methodology	Object Oriented Methodology	Component Based Methodology
Characteristics	- Program logic oriented	- Data model oriented - Enterprise integration data model	- Object that is mixed data with logic oriented - Reuse by descent	- Interface oriented - Composed interface is component
Main products	- Function chart - Data flow diagram	- Table definition - Application structural map	- Class Diagram - Sequence Diagram - Usecase Diagram	- Component Diagram - Reuse Plan
The number of main products	- 10~15	- 12~20	- 20~25	- 25~30
Languages	- COBOL, C, VB, etc.	- COBOL, C, VB, etc.	- C++, JAVA, C#, etc.	- Irrelevant to program language

단계를 정리하여 나타내고 있다. 문제를 정의하고 목표를 설정하며, 평가 기준을 계층화하고, AHP는 정량적인 분석이 곤란한 상태에서 정성적인 요소를 반영하여 가중치 또는 중요도를 구하는데 유용하게 응용 될 수 있다는 점에서 평가에 대한 일관성 여부를 추론할 수 있다는 장점이 있다[12, 13].

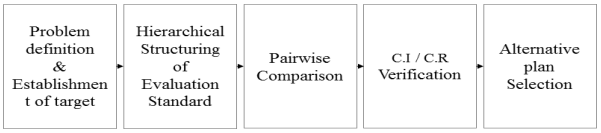

Fig. 1. Stages of AHP

3. AHP를 통한 소프트웨어 개발방법론 적합도 평가

소규모 소프트웨어 개발을 위한 소프트웨어 개발방법론의 적합성(Applicability)을 평가하기 위해서는 기존에 많이 사용되는 표준적인 소프트웨어 개발방법론의 여러 가지 요소들에 대한 분석과 정성적인 요인을 고려해야 한다. 따라서 T.L.Satty가 제안한 다기능 문제 의사결정방법인 AHP (Analytic Hierarchy Process)를 적용하여 가중치 추정(Weight Estimation)을 통한 소프트웨어 개발방법론의 적합도를 평가하였다[14]. 적합도를 평가하는 순서는 Fig. 1에 따라 진행하였다.

3.1 문제 정의 및 목표 설정

본 논문의 서론에서 제시된 것처럼, 문제 정의(Problem Definition)는 소규모 소프트웨어를 개발할 때, 표준적인 소프트웨어 개발방법론에서 최적의 단위 일(Task)을 선택하는 것으로 Table 2와 같이 정의하였고, 목표 설정(Establishment of Target)은 다수의 단위 일에서 소규모 소프트웨어를 개발할 때 적합한 최적의 단위 일(Task) 선택으로 정의하였다.

Table 2. Problem Definition and Establishment of Target

Problem Definition	Developing Small-scale software, What will you selects TASK of software development methodology? You choice of Evaluation Standards.
Establishment of Target	Developing Small-scale software, Choice of Applicability Task

평가 기준 선정을 위해 총 3개의 집단을 선정하여 설문을 실시하였다. 첫번째는 삼성SDS와 SK C&C, IBM에서 소프트웨어 개발방법론을 관리하는 개발방법론 전문가 집단, 두번째는 대규모, 소규모 SI(System Integration)사업을 경험한 PM(Project Manager)와 PL(Project Leader), 그리고 IT 컨설턴트들을 대상으로 하는 집단, 마지막으로 중소기업에서 소프트웨어 개발을 총괄하는 PM과 PL, 개발팀장을 대상으로 하였다. 총 3개의 집단으로 구성한 이유는 개발방법론에 대한 각 단위 일들의 중요성을 가장 잘 아는 전문가 집단과 개발방법론의 필요성은 알고 있지만 적은 단위 일로도 최대의 효과를 요구하는 중소규모 소프트웨어 개발 팀 집단을 우선 선정하였다. 그리고 이 두 집단을 보완하기 위해 대규모와 중소규모의 소프트웨어 개발 팀을 모두 경험한 집단을 선정하였다. Table 3은 설문 대상 집단과 인원수를 나타낸 것이다. 설문 대상자는 각 집단별 20명씩 총 60명으로 하였으며, 설문 당시 설문자들이 상호 토론을 할 수 없도록 일대일 인터뷰를 실시하였다.

Table 3. Survey Groups

Survey Group	Number of People
Expert of Software Development Methodology	20
PM, PL, and Consultant who experience ALL of Large, Medium and Small Scale Project	20
PM, PL who experience Medium and Small Scale Project	20

소프트웨어 개발방법론 전문가 관련 논문 및 연구 보고서 등의 리서치를 통해 일반적으로 잘 알려진 정보공학(Information Engineering) 개발방법론을 기준으로 단위 일을 1차적으로 추출하였고, Table 3에서 정의된 세개의 집단에게 중소규모의 소프트웨어 개발 팀에서 반드시 필요한 단위 일을 선정하도록 설문을 실시 하였다. 세개의 설문 집단에서 모두 선정하지 않은 단위 일은 제외하여 Table 4와 같이 총 15개의 단위 일을 최종 선정하였다. 이때, 각 집단이 선정한 단위 일을 대안A(Alternative A), 대안B(Alternative B), 그리고 대안C(Alternative C)로 정의하였다.

3.2 계층적 구조화

3.1에서 정의된 평가 기준과 대안을 계층적(Hierarchy)인 구조로 파악하여 최적의 대안을 선택하기 위해 Fig. 2와 같이 계층적 구조도(Hierarchical layer)를 나타내었다. 목표를 달성하기 위해 각각의 평가 기준을 선택하여 포함하는 대안이 어떤 것이 있는지를 확인할 수 있다[13].

3.3 쌍대 비교

각 평가 기준에 대해서 항목 간에 어느 쪽이 얼마만큼의 가중치를 갖는지를 AHP에서 사용하는 일대일 비교치 목록을 기준으로 쌍대 비교(Pairwise Comparison)를 통해 중요성의 정도에 따라 상대적 중요도의 설정을 위해 앞에서 정의한 평가 기준들의 가중치(Weight)를 도출한다[13]. 각 평가 기준들의 평가 자료를 종합하는 방법으로는 기하평균법(Geometric Average)을 이용하였다[15].

각 평가 기준을 으로 두고, 복잡도 정도를 이라 할 때, AHP에서 사용하는 일대일 비교치 목록인 Table 5를 참고하여 비교한 값을 정방행렬 [A]로 배열하는 방법은 Table 6과 같다. 는 자신에 비교한 것으로 그 값은 1이다. 는 에 비교한 의 복잡도의 정도를 나타낸 값이고, 정리하면 은 에 비교한 쌍대 비교값을 의미한다[13].

쌍대 비교값을 정의한 Table 5를 3.1에서 나타난 세개의 집단에게 알려주고 설문을 실시하였다. 이때, 집단을 구분하

Table 4. Evaluation Standard and Alternative

Phase	Task	Alternative A	Alternative B	Alternative C
Requirement Definition	(1) Business Process Definition	◎	◎	◎
	(2) Requirement Definition	◎	◎	◎
Analysis	(3) Current System Analysis	◎	◎	
	(4) Application Function structure Analysis	◎	◎	
	(5) Architecture Analysis	◎		
	(6) Logical Data Modeling	◎		
	(7) Application Function Design	◎	◎	◎
Design	(8) Screen Design	◎	◎	◎
	(9) Physical Data Modeling	◎	◎	◎
	(10) Integrated test Design	◎	◎	
Development	(11) Integrated test Implementation	◎	◎	◎
	(12) Release and Emergency planning	◎		
Implementation	(13) Acceptance test Implementation	◎	◎	◎
	(14) Manual Writing	◎	◎	
	(15) Test operation planning	◎		

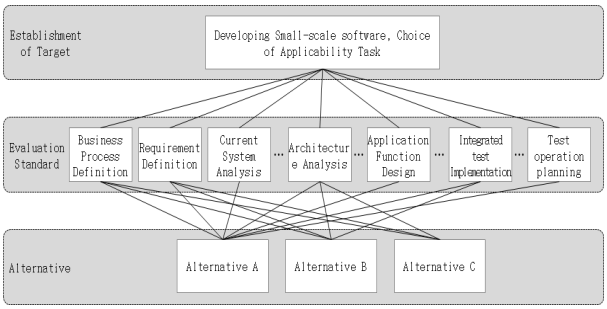

Fig. 2. Hierarchical layer for AHP analysis

Table 5. Pairwise comparison matrix for AHP

Value of	Interpretation
1	Objectives i and j are of equal importance
3	Objectives i is weakly more important than objective j
5	Experience and judgment indicate that objective i is strongly more important than objective j
7	Objective i is very strongly or demonstrably more important than objective j
9	Objective i is absolutely more important than objective j
2, 4, 6, 8	Intermediate values
reciprocal	Objectives j is more important objective i

Table 6. Array of Pairwise comparison for AHP

	A_1	A_2	⋯	A_n
A_1	v_1/v_1	v_1/v_2	⋯	v_1/v_n
A_2	v_2/v_1	v_2/v_2	⋯	v_2/v_n
⋯	⋯	⋯	⋯	⋯
A_n	v_n/v_1	v_n/v_2	⋯	v_n/v_n

지 않고 설문을 하도록 하여 다른 집단이 선정한 대안에 대한 평가도 실시될 수 있도록 하였다. 정리하면, Table 4에서 나타난 15개의 평가 기준 항목들에 대해 세개의 집단이 모두 Table 5 기준으로 쌍대 비교를 실시하여 나온 결과를 Table 6의 가중치 구하는 방법을 통해 Table 7과 같은 결과가 산출되었다. Equation (1)을 이용하여 각 평가 기준들의 기하평균을 산출한 다음, Equation (2)와 같이 전체의 합으로 각 항목의 기하평균 값을 나누어 각 평가 기준들의 가중치를 계산한다[13].

Table 7. Pairwise Comparison for Evaluation Standard

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	Mean	Weight	Rank
(1)Business Process Definition	1.000	1.000	3.000	3.000	5.000	5.000	0.500	0.500	0.500	3.000	0.500	5.000	2.000	3.000	7.000	1.833	0.090	6
(2)Requirement Definition	1.000	1.000	3.000	3.000	5.000	5.000	1.000	0.500	0.500	3.000	0.500	5.000	2.000	3.000	7.000	1.920	0.094	4
(3)Current System Analysis	0.333	0.333	1.000	1.000	3.000	3.000	0.333	0.250	0.250	1.000	0.250	3.000	0.500	1.000	5.000	0.806	0.039	11
(4)Application Function structure Analys	0.333	0.333	1.000	1.000	3.000	3.000	0.333	0.250	0.250	1.000	1.000	3.000	0.500	1.000	5.000	0.884	0.043	8
(5)Architecture Analysis	0.200	0.200	0.333	0.333	1.000	1.000	0.200	0.200	0.167	0.333	0.250	0.500	0.250	0.333	1.000	0.342	0.017	14
(6)Logical Data Modeling	0.200	0.200	0.333	0.333	1.000	1.000	0.200	0.167	0.200	0.333	0.200	1.000	0.250	0.333	1.000	0.353	0.017	13
(7)Application Function Design	2.000	1.000	3.000	3.000	5.000	5.000	1.000	0.500	0.500	3.000	0.333	3.000	2.000	3.000	6.000	1.872	0.092	5
(8)Screen Design	2.000	2.000	4.000	4.000	5.000	6.000	2.000	1.000	0.500	3.000	2.000	4.000	2.000	3.000	7.000	2.625	0.128	2
(9)Physical Data Modeling	2.000	2.000	4.000	4.000	6.000	5.000	2.000	2.000	1.000	3.000	2.000	4.000	2.000	3.000	7.000	2.879	0.141	1
(10)Integrated test Design	0.333	0.333	1.000	1.000	3.000	3.000	0.333	0.333	0.333	1.000	0.333	3.000	0.500	1.000	5.000	0.853	0.042	9
(11)Integrated test Implementation	2.000	2.000	4.000	1.000	4.000	5.000	3.000	0.500	0.500	3.000	1.000	5.000	2.000	3.000	6.000	2.192	0.107	3
(12)Release and Emergency planning	0.200	0.200	0.333	0.333	2.000	1.000	0.333	0.250	0.250	0.333	0.200	1.000	0.250	0.333	0.500	0.381	0.019	12
(13)Acceptance test Implementation	0.500	0.500	2.000	2.000	4.000	4.000	0.500	0.500	0.500	2.000	0.500	4.000	1.000	3.000	6.000	1.393	0.068	7
(14)Manual Writing	0.333	0.333	1.000	1.000	3.000	3.000	0.333	0.333	0.333	1.000	0.333	3.000	0.333	1.000	4.000	0.818	0.040	10
(15)Test operation planning	0.143	0.143	0.200	0.200	1.000	1.000	0.167	0.143	0.143	0.200	0.167	2.000	0.167	0.250	1.000	0.288	0.014	15
Total	12.576	11.576	28.200	25.200	51.000	51.000	12.233	7.426	5.926	25.200	9.567	46.500	15.750	26.250	68.500	20.457	1.000	

$$A = \{_{k=1}^{n} a_k\}^{\frac{1}{n}} = (a_1 a_2 \dots a_n)^{\frac{1}{n}} \quad (1)$$

$$W_i = \frac{GA_k}{\sum_{k=1}^{n} GA_k} \quad (2)$$

Table 8. Pairwise Comparison for Alternative

(1)Business Process Definition

	(1)	(2)	(3)	Mean	Weight
(1)Alternative A	1.000	1.000	1.000	1.000	0.333
(2)Alternative B	1.000	1.000	1.000	1.000	0.333
(3)Alternative C	1.000	1.000	1.000	1.000	0.333
Total	3.000	3.000	3.000	3.000	1.000

:

(15)Test operation planning

	(1)	(2)	(3)	Mean	Weight
(1)Alternative A	1.000	2.000	2.000	1.587	0.529
(2)Alternative B	1.000	1.000	1.000	1.000	0.333
(3)Alternative C	1.000	1.000	1.000	1.000	0.333
Total	3.000	4.000	4.000	3.634	1.211

또한, 각 평가 기준 항목에 대해 대안별로 중요도를 체크하기 위해 Table 7을 산출하는 방법으로 쌍대 비교를 실시하였다. Table 8이 각 평가 기준 항목별로 대안의 가중치를 나타내고 있다.

3.4 일관성 검증

쌍대 비교 행렬은 의사결정자의 주관에 따라 결정되므로, 전문가가 주관적으로 판단한 요소간의 중요성을 얼마나 일관성 있게 응답했는가를 논리적으로 판단하기 위해 일관성 검사(Consistency Verification)를 수행한다.

이것은 쌍대 비교 행렬로 계산한 가중치를 사용하여 실제로 얻어진 행렬과의 차이를 수량적으로 검토하는 것이다. 일관성 검사에는 일관성 지수(Consistency Index: CI)와 일관성 비율(Consistency Ratio: CR) 값이 필요하다. 일관성 지수 CI는 , 비교 요소의 수를 n이라고 했을 때, Equation (3)과 같이 계산한다.

$$CI = \frac{r_{max} - n}{n - 1} \quad (3)$$

일관성 비율 CR은 확률 지수(Random Index)와 CI 값을 이용하여 Equation (4)와 같이 계산하며, 일관성이 작을수록 CR 값이 크다. 일반적으로 CI와 CR이 0.1 이하일 경우 논리적 일관성을 유지한 것으로 판단한다. 확률 지수 RI는 쌍대 비교의 대상 개수 n에 따라 Table 9와 같다[13, 16].

$$CR = \frac{CI}{RI} \quad (4)$$

Table 7의 쌍대 비교 행렬 값과 도출한 가중치의 일관성 검사를 위해 구했더니 15.818이 나왔고, 이것을 Equation (3)에 대입하면 Equation (5)와 같이 CI는 0.053이 계산되었다.

Table 9. Value of the random index

n	2	3	4	5	6	7	8
RI	0.00	0.58	0.90	1.12	1.24	1.32	1.41
n	9	10	11	12	13	14	15
RI	1.45	1.49	1.51	1.48	1.56	1.57	1.59

두 번째 일관성 검사를 위해 Equation (4)에 CI와 RI를 대입하면 Equation (6)과 같이CR값이 0.037로 계산되었다[13].

$$CI = \frac{15.818 - 15}{15 - 1} = 0.058 \quad (5)$$

$$CR = \frac{CI}{RI} = \frac{0.058}{1.59} = 0.037 \quad (6)$$

평가 기준 항목 간 쌍대 비교 행렬의 CI가 0.058, CR이 0.037로 두개 모두 0.1보다 작으므로 일관성에 문제가 없음을 검증하였다. 위와 같은 방법으로 Table 8에서 나타난 쌍대 비교값의 일관성을 검증하였고, 각 항목별 CI, RI는 모두 0.1 이하로 측정되어 일관성이 있는 것으로 확인되었다.

3.5 대안 선택

3.4에서 평가 기준 항목과 각 평가 기준 항목의 대안에 대한 쌍대 비교는 모두 일관성을 유지한 것으로 나타났다. 이번 절에서는 3.3에서 측정된 가중치로 AHP 점수(AHP Point: AP)를 측정하였다. Table 10은 Table 7과 Table 8에서 측정된 가중치이다.

Table 10. Weight of Evaluation Standard and Alternative

	(1)	(2)	(3)	...	(13)	(14)	(15)
Weight of Evaluation Standard	0.090	0.094	0.039	...	0.068	0.040	0.014
(1)Weight of Alternative A	0.333	0.333	0.231	...	0.333	0.420	0.529
(2)Weight of Alternative B	0.333	0.333	0.231	...	0.333	0.420	0.333
(3)Weight of Alternative C	0.333	0.333	0.333	...	0.333	0.333	0.333

각 대안별 AHP 점수는 평가 기준 항목 가중치와 대안 가중치의 곱의 전체 합으로 나타낸다. 대안 A의 AP = (0.090X0.333) + …… + (0.014X0.529)로 나타내며 대안B, 대안C도 같은 방법으로 측정할 수 있다. Table 11은 각 대안의 AHP 점수를 나타낸다. 표에서 보는 것처럼 적합도는 세 개 모두 유사하게 나왔으나 AHP 점수가 가장 큰 대안3이 적합도가 가장 높다고 볼 수 있다. 이러한 결과로 보아 소규모 소프트웨어 개발에 적합한 소프트웨어 개발방법론은 Table 4의 대안3(Alternative 3)으로 볼 수 있다.

Table 11. AHP Point of Alternative

	AHP Point
(1)Weight of Alternative A	0.311
(2)Weight of Alternative B	0.313
(3)Weight of Alternative C	0.317

대안3이 AHP점수가 가장 높다는 의미는 대안A를 선정한 개발방법론 전문가 집단과 대안 B를 선정한 대규모, 중소규모 소프트웨어 개발 팀을 경험한 집단들이 중요하다고 생각되는 단위 일들 중에도 경우에 따라 제외하고 사용할 수 있다는 것을 의미한다. 본 결과가 나타내고 있는 것을 정리하면, 소프트웨어 개발방법론은 현대의 소프트웨어 개발에 있어 반드시 필요한 도구이다. 하지만 정의된 모든 단위 일을 할 수는 없기 때문에 경우에 따라 필요한 단위 일을 선정해야 한다. 본 논문에서는 기존의 정보공학 개발방법론에서 정의된 단위 일 중에 반드시 필요한 것으로 15개를 선정하였다. 이 15개는 전문가와 PM, PL, 컨설턴트 등이 동의했던 사항이다. 따라서 가급적 15개 모두 사용하는 것을 기본으로 하는 대안 A를 권장하지만 경우에 따라 대안 B와 대안 C를 선택해도 무방할 것으로 보인다. 정보공학 개발방법론의 각 단위 일들은 모두 검증되어 사용 중이기 때문이다. 이러한 이유로 AHP점수가 대안B > 대안 A > 대안 C 순으로 나왔다면, 단위 일이 15개 모두 있는 대안 A를 기준으로 대안 B와 대안 A는 활용이 가능하지만 대안 C는 어렵다고 판단할 수 있다.

4. 결 론

급변하는 현대의 비즈니스 환경에 적응하면서 서비스해야 하는 지금의 소프트웨어들은 제한된 사용자들이 원하는 형태로 개발했던 이전의 소프트웨어와는 달리 체계적인 개발, 재사용, 품질과 유지보수 등을 생각할 수밖에 없다. 이러한 이유로 대규모 소프트웨어 개발 프로젝트에서나 사용하던 소프트웨어 개발방법론을 소규모 소프트웨어를 개발하는 곳에서도 필요성을 인식하기 시작했지만 너무나 양이 많고 복잡한 소프트웨어 개발방법론을 도입하기는 쉽지 않았다.

본 논문에서는 중소규모 소프트웨어 개발 프로젝트에서도 쉽게 적용할 수 있도록 정보공학 개발방법론에서 가장 필요한 단위 일을 선정하였고, 선정된 단위 일을 대안으로 정리하여 적합성을 AHP 기법으로 검증하였다. 먼저, 개발방법론 전문가를 비롯한 개발방법론 경험자를 대상으로 단위 일에 대한 선정을 진행하였으며, 설문과 인터뷰를 거쳐 3가지의 대안을 도출하였다. 적합도를 검증하기 위해 AHP를 사용하여 각 대안을 평가하였고, 본 논문에서 선정된 15개의 단위 일을 모두 사용하는 것이 좋으나 경우에 따라 중요한 단위 일만 선택하여 사용하는 것도 문제없다는 결론이 나왔다.

소프트웨어 개발방법론은 소프트웨어의 품질과 재사용 등 다양한 이점이 있지만 적용에 따른 시간과 비용이 많이 들고 반드시 전문가가 필요하기 때문에 적용에 어려움이 많았다. 하지만 본 논문에서 제시된 매우 필요한 단위 일만 포함하는 최소 규모의 소프트웨어 개발방법론을 적용함으로써 최소 비용과 시간을 들여 체계적인 개발과 품질 등의 효율성이 향상될 것으로 기대된다. 본 논문의 결과를 바탕으로 공공, 금융, 제조서비스와 같은 특화된 소프트웨어 개발을 위한 소프트웨어 개발방법론의 적합도도 추가로 평가해볼 계획이다.

참 고 문 헌

[1] Ian Sommerville, “Software Engineering (9th Edition)”, Pearson Education, 2010.

[2] Roger S. Pressman, “Software Engineering: A Practitioner's Approach”, McGraw-Hill, 2009.

[3] Ikhwan Kim, “Dreaming of Global Software.” ISBN. 9788979147452, Hanbit Soft, 2010.

[4] Tom Poppendieck, “Lean Software Development”, Addison-Wesley, 2003.

[5] Stephen R Schach, “Object-Oriented and Classical Software Engineering”, McGraw-Hill College, 2005.

[6] Sungrak Kim, “System Analysis and Design” ISBN 9788980813926, Hyunwoosa, 2013.

[7] Booch, “Object- Oriented Analysis and Design with Applications 3/E”, Pearson Education Asia, 2006.

[8] Kim Younghee, Jin Byeongwoon, Yang Taeyeon, “A Comparison Study on Software Development Methodologies”, The Korean Institute of Information Scientists and Engineers, 1998.

[9] Samsung SDS KPEA, “Software Engineer” ISBN 9788 946052697, Hanwool Academy, 2010.

[10] Dean Leffingwell, “Scaling Software Agility: Best Practices for Large Enterprises”, Addison-Wesley Professional, 2007.

[11] Samsung SDS Software Engineering Center, “Software Development Methodology”, Samsung SDS, 2003.

[12] Zahedi, F., “The Analytic Hierarchy Process-A Survey of the Method and its Applications,” Interfaces, Vol.16, No.4, pp.96-108, 1986.

[13] Gwangyeul Yun, “An Evaluation of the Service Implementation Complexity of SOA and WOA through AHP”, Korea Computer Congress Vol.37, 2010.

[14] T.L.Satty. “The Analytic Hierarchy Process,” McGraw-Hill, 1980.

[15] Hannan, E.L., “An Eigenvalue Method for Evaluating Contestants”, Computer and Operations Research, 1983.

[16] P.J. Petit, “What is the best energy-delivery system for hand-held stope drilling and associated equipment in narrow-reef hard rock mines?”, Journal of the Southern African Institute of Mining and Metallurgy, 2013.

김 규 억

e-mail : kyueok.kim@gmail.com

1997년 단국대학교 전산통계학과 (석사)

1998년~현 재 삼성SDS 수석컨설턴트

관심분야 : S/W Engineering, IT Policy, Architecture

유 해 영

e-mail : yoohy@dankook.ac.kr

현 재 단국대학교 소프트웨어학과 정교수

관심분야 : S/W Engineering, S/W Testing, Web Engineering