The Journal of Korean
Institute of CALS/EC
Vol 6, No. 2, August 2001

Use Case에 의한 소프트웨어 규모 예측 방법에 대한 실증적 연구

서예영*, 이남용*

An Empirical Study of Software Size Estimation Techniques by Use Case

Ye-Young Suh, Nam-Yong Lee

Abstract

There has been a need for predicting development efforts and costs of the system during the early stage of the software process and hundreds of metrics have been proposed for computer software, but not all provide practical support to the software engineer. Some demand measurement that is too complex, others are so esoteric that few real-world professionals have any hope of understanding them, and others violate the basic intuitive notions of what high-quality software really is. It is worthwhile that metrics should be tailored to best accommodate specific products and processes after grasping their good and no good point.

This paper describes two size estimation techniques, the Karner technique and the Marchesi technique, and compares and analyzes them with proposed evaluation criteria. Both techniques are to estimate software size analyzed by use case that is mainly described during the object-oriented analysis phase. We also present an empirical comparison of them, both are applied in the Internet Medicine Prescription System. We also propose some guidance for experiments based on our analysis. We believe that it should be facilitating project management more effective by adjusting software metrics properly.

* 숭실대학교 컴퓨터학부

1. 서론

소프트웨어 프로젝트 관리 측면에서 소프트웨어 개발과 유지보수의 비용을 줄이고 제어하려는 시도가 일어나고 있으며, 이를 위한 기법에 대한 연구가 진행되고 있다. 소프트웨어 개발과 유지보수의 정확한 비용과 작업일정 예측은 많은 경영 의사결정, 예산, 인력 배치와 믿을 수 있는 계약 입찰을 위한 경쟁을 지원한다는 점에서 매우 높은 가치가 있다. 따라서 시스템 개발 주기의 초기 단계에서 비용과 노력 등을 미리 예측할 수 있는 소프트웨어 규모 예측 방법이 요구되고 있다.

이에 따라 소프트웨어 규모 예측을 위한 수백개의 다양한 소프트웨어 메트릭스가 제안되고 있지만, 제안된 모든 소프트웨어 메트릭스가 현재 소프트웨어 엔지니어에게 유용하게 지원되지 않고 있는 실정이다. 어떤 소프트웨어 메트릭스는 너무 복잡한 측정을 요구하고 또 어떤 소프트웨어 메트릭스는 너무 난해해서 소수의 실제 전문가만이 이해할 수 있고 고품질의 소프트웨어에 대한 기본적인 통찰력을 어지럽히기 때문이다 [14]. 사용자가 이용하고자 하는 소프트웨어 메트릭스의 장점과 단점을 정확히 파악하고 적용시켜야 시행착오를 피할 수 있다. 따라서 다양한 소프트웨어 규모 예측 방법에 대한 비교 분석이 필요하다.

이에 따라 'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측 방법에 대한 연구[1]'에서는 Use Case 다이어그램 분석을 통한 소프트웨어의 규모 예측을 위해 제안된 Karner[7] 방법과 Marchesi [11] 방법을 비교 분석하였다.

본 논문에서는 'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측 방법에 대한 연구'를 인터넷 의료 처방전 시스템(Internet Medical Prescription System, IMPS) 개발 사례에 적용하여 'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측 방법에 대한 연구'에서 제시한 평가기준에 따라 Use Case 다이어그램 분석을 통한 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 비교 분석한다. 그리고 비교 분석한 결과에 따른 두 가지 방법의 장단점을 보완한 규모 예측 방법의 개선점을 도출한다.

논문의 구성은 다음과 같다. 먼저 관련 연구에서는 'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측 방법에 대한 연구'에 대한 설명과 산출한 예상 결과 값을 기술한 뒤 IMPS 대해 간단히 소개한다. 그리고 규모예측 방법을 비교 분석할 평가기준을 기술한 뒤, 제시한 평가기준에 따라 두 가지 소프트웨어 규모 예측 방법을 IMPS 개발 사례를 통해 비교 분석한다. 그리고 비교 분석한 결과에 따른 규모 예측 방법의 개선점을 도출하고 결론 및 향후 연구를 제시한다.

2. 관련연구

2.1 Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측방법에 대한 연구

Use Case 모델은 시스템에 의해서 처리되는 모든 기능적인 요구사항을 표현하기

위한 것으로, 사용자와 개발자, 분석가 등 넓은 범위의 이해 관계자가 시스템에 대한 기능성을 표현하는 방법이다[8]. 프로젝트 관리에서, Use Case와 시나리오는 소프트웨어 개발 프로세스의 반복에 관한 내용을 정의하는데 사용된다. 기능 점수 분석과 같은 기법에 의해 개발 노력 또는 비용에 대한 예측이 Use Case의 서술로부터 도출 가능하다[13].

'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측에 대한 연구'에서는 객체지향 기술에 의한 객체지향 시스템 개발을 위해 OMG(Object Management Group)에서 제안한 객체지향 시스템 분석 설계 언어 UML(Unified Modeling Language)의 9가지 다이어그램 중 시스템 분석 단계에서 주로 작성되는 Use Case 다이어그램 분석을 통한 소프트웨어 규모 예측 방법을 비교 분석하기 위해 먼저 평가기준을 제시하고 Use Case에 의한 소프트웨어 규모 예측 방법으로 제안된 Karner 방법과 Marchesi 방법이 앞에서 제시한 평가기준의 지원정도를 이론적으로 비교 분석했다.

비교 분석의 대상이 된 Use Case 의한 규모 예측 방법을 살펴보면, Karner는 UCP(Use Case Points)라는 Use Case 기반의 프로젝트를 완료할 동안 인력(Man-Hour)을 예측하는 방법을 제안했다. 이 방법은 개발 초기에 사용하며 기능 점수 또는 COCOMO와 같은 그 밖의 예측 방법과 함께 프로젝트 동안의 인력(Man-Month) 수를 구하는데 사용된다. UCP는 시스템에 대한 액터와 Use Case의 타입에 따른 가중 인자를 부여한 UUCP(Unadjusted Use Case Point)를 프로젝트에 관한 테크니컬한 복잡도 TCF(Technical Complexity Factor)와 프로젝트에 참여한 사람들의 경험 수준 EF(Environment Factor)의 가중 인자를 사용하여 조정된다. 산출된 UCP당 20 인원-시간 인자를 사용하여 프로젝트를 예측을 한다.

UUCP(Unadjusted Use Case Points)
= Total Actor Weight + Total Use Case Weight
TCF(Technical Complexity Factor)
= 0.6 + (0.01 • TFactor)
EF(Environmental Factor)
= 1.4 + (-0.03 • EFactor)
UCP(Use Case Points)
= UUCP • TCF • EF

Marchesi 는 객체지향 분석 단계 동안 Use Case 다이어그램과 클래스 다이어그램에 관련한 소프트웨어 메트릭스를 제안했다. 제안된 소프트웨어 메트릭스는 개발 노력, 구현 시간과 개발 중 비용에 대한 초기 예측을 허용하며, 분석 단계이후 지속적인 객체지향성과 품질 측정을 목적으로 한다. 그 중 여기서 비교 분석할 대상은 요구사항 유도 단계에서 Use Case 다이어그램 상의 시스템 복잡도에 대한 초기예측을 목적으로 하는 Use Case 기반 소프트웨어 메트릭스로 UC1, UC2, UC3, UC4 네 가지로 나뉜다. UC1 은 Use Case 다이어그램 상의 전체 Use Case 의 개수, UC2 는 Use Case 와 액터 간의 의사소통의 개수, UC3 는 Use Case 와 Use Case 간에 ≪extends≫, ≪include≫ 관계와 같은 중복을 가지지 않는

의사소통의 수로 시스템의 복잡도를 나타낸다. UC4 는 총 Use Case 의 개수 UC1 과 ≪extends≫, ≪include≫ 관계의 의사소통 정도를 고려함으로써 UC3 를 수정한다. 만약 해당 시스템에 ≪extends≫, ≪include≫ 관계가 없는 시스템(UC2=UC3)은 동일한 UC3 값을 가지며 ≪extends≫, ≪include≫ 관계를 가지는 시스템보다 훨씬 적은 복잡도를 가진다. 즉, 독립적인 의사소통에 대한 복잡도를 UC3 로, 포괄적인 복잡도를 UC4 로 예측한다.

$$UC1 = N_U$$

$$UC2 = \sum_{i=1}^{N_U}\sum_{k=1}^{N_A} C_{ik}$$

$$UC3 = \sum_{i=1}^{N_U}\left(\sum_{k=1}^{N_A} e_{ik}\right)^{KU}$$

$$UC4 = K_1 UC1^2 + K_2[smm([c]) - smm([E])]$$

2.2 인터넷 의료 처방전 시스템

IMPS란 병원, 약국, 감독기관, 환자를 네트워크화 하여 인터넷으로 병원과 약국간의 실시간 연결, 신뢰성 있는 처방전의 발행으로 약품 조제 및 오류의 방지, 약국의 상태 확인 등, 신속한 처방전 발행에 따른 환자의 처방약 조제 대기시간을 줄여주는 다양한 서비스를 제공하는 시스템을 말한다. 또한 이 시스템은 병원과 약국의 일반 업무를 자동화하여 병원과 약국간의 모든 업무를 연계하여 자동적으로 수행 가능하다.

객체지향 소프트웨어 시스템 개발 초기 단계에서, IMPS의 기능을 시스템 운영 관련 기능과 처방전 관련 기능, 두 가지로 나누고 각각의 기능에 관련한 Use Case 즉 기능성을 10가지로 기술한다.

객체지향 소프트웨어 시스템 개발 정교화 단계에서, 초기화 단계에서 기술한 IMPS의 10가지 Use Case에 대한 하부 Use Case 를 기술한다.

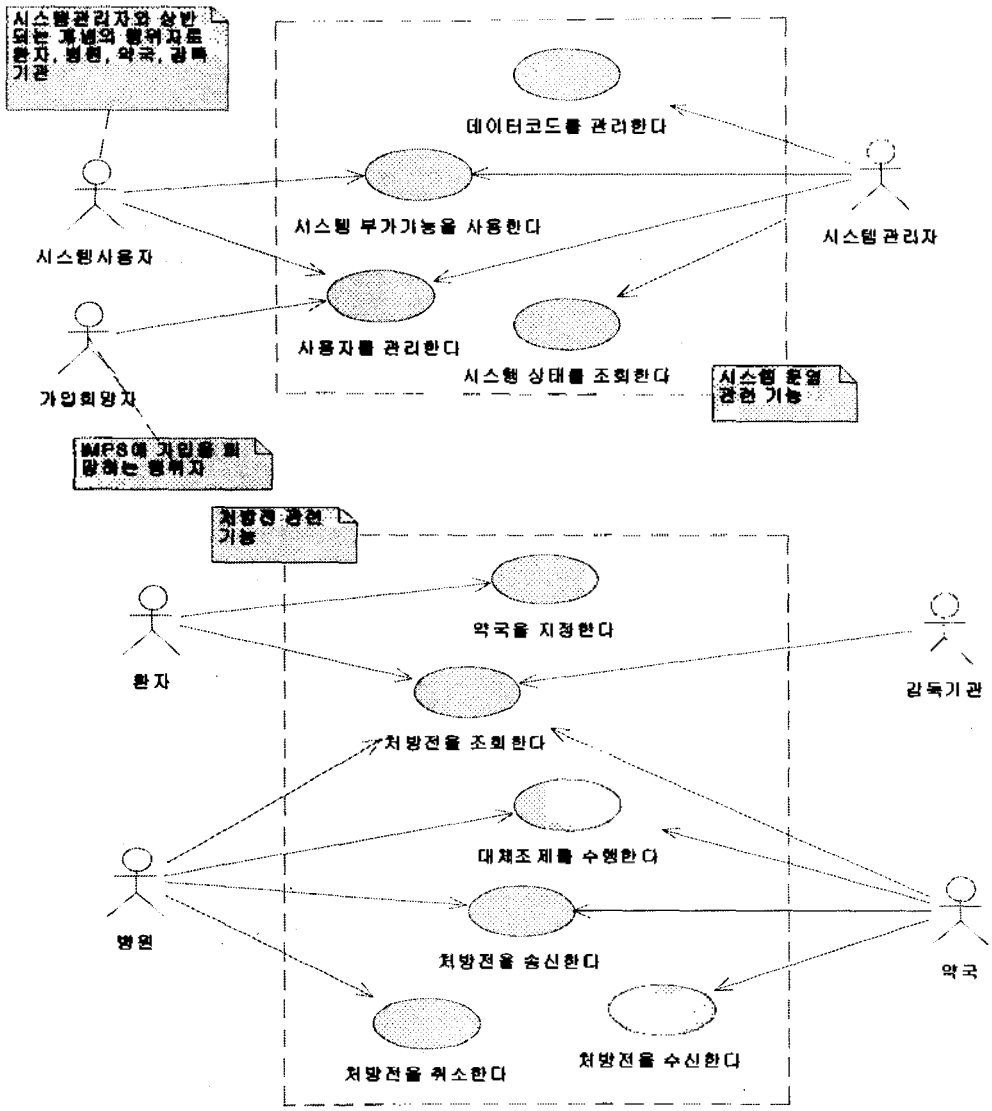
2.3 IMPS 개발 사례에 적용시의 결과 예측

'Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측에 대한 연구'에서 평가기준에 따른 두 가지 규모 예측 방법의 이론적인 비교 분석 결과를 IMPS 개발 사례에 적용하면 다음과 같은 결과가 기대된다.

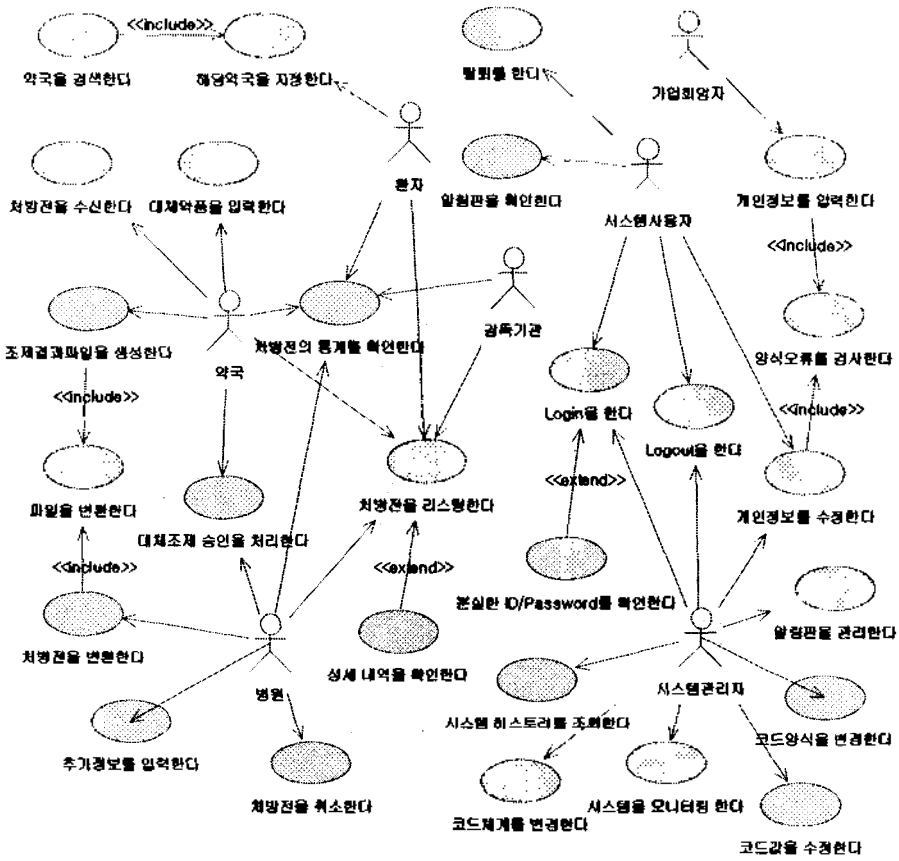
Karner와 Marchesi 두 가지 방법 모두 시스템이 진화함에 따른 시스템에 대한 요구사항의 변화량을 잘 반영하고 있다. 즉, <그림 1>, <그림 2>에 나타난 바와 같이, 시스템 개발 초기 단계에서의 Use Case 10개가 정교화 단계에서의 Use Case 27개로 변화함에 따라서 Karner와 Marchesi 방법 모두 Use Case의 수의 변화에 대해 산출되는 값도 변화하게 된다.

Karner 방법은 Use Case 다이어그램 외에도 동일한 도메인에 있어서 TCF, EF 와 같은 가변적인 시스템 개발 환경 다시 말하면 개발 방법론에 따라 산출되는 결과값이 달라질 수 있는 여지가 있다. 반면에 Marchesi 방법은 동일한 소프트웨어 도메인에 대해 서로 다른 독립된 소프트웨어 전문업체는 동일한 소프트웨어 메트릭스 결과값을 산출가능하나 Use Case 다이어그램에

서 산출한 복잡도는 직접적으로 시간이나 노력 및 비용을 예측할 수 없다.



<그림 1> 시스템 개발 초기 단계에서의 Use Case 다이어그램



<그림 2> 시스템 개발 정교화 단계에서의 Use Case 다이어그램

3. 평가기준

본 논문에서는 Use Case 다이어그램 분석을 통한 소프트웨어 규모 예측 방법을 비교 분석할 평가 기준을 객체지향 소프트웨어 개발 프로세스의 반복적이고 점진적인 접근 방법에 따른 객체지향 소프트웨어 개발(OOSD; Object-Oriented Software Development) 측면, 규모 예측 방법(SIZE; SIZe Estimation) 측면과 보편적인 소프트웨어 메트릭스가 가지는 소프트웨어 메트릭스(SMET; Software METrics) 측면으로 나누어 제시한다.

3.1 객체지향 소프트웨어 개발(OOSD) 측면

객체지향 개발에 대해 가장 보편적인 접근 방법이며 이른바 반복적이고 점진적인

접근 방법이라 불리는 객체지향 시스템에 대한 분석, 설계 코딩 활동들은 명확하게 분리되어 있지 않다[6,9]. 따라서 특정 활동에 관련된 소프트웨어 메트릭스는 이전 활동의 소프트웨어 메트릭스와 일관성을 유지하면서 그것에 근거를 두어야 한다.

- ✧ 일관성 및 연속성: 소프트웨어 메트릭스는 시스템이 진화할 때에, 일관성과 연속성을 유지하면서 재산출 가능해야 한다(시스템에 있어 "작은" 변화는 시스템에 대한 소프트웨어 메트릭스에 있어 "작은" 변화를 이끌어야 한다).

3.2 규모 예측 방법(SIZE) 측면

소프트웨어 규모(Size)는 길이(Length), 기능성(Functionality), 복잡도(Complexity) 세가지 관점을 가진다[12]. 소프트웨어 규모는 세가지 관점에 따라 측정 방법은 서로 다르지만 결과 값은 노력, 시간, 비용으로 산출된다.

- ✧ 노력, 시간, 비용: 소프트웨어 규모 예측을 하는 소프트웨어 메트릭스가 측정하는 속성으로, 해당 소프트웨어 메트릭스가 이 세 가지 중 하나 이상을 지원 가능해야 한다.

3.3 소프트웨어 메트릭스(SMET) 측면

- ✧ 정형화: 소프트웨어 메트릭스의 수학적인 계산은 이상한 단위(unit) 조합을 만들지 않는 정형화된 메트릭스를 사용하여 측정 가능해야 한다.
- ✧ 단순성 및 계산성: 소프트웨어 메트릭스 유도 방법이 비교적 배우기 쉽고 소프트웨어 메트릭스의 계산이 과도한 노력이나 시간이 필요하지 않아야 한다[10].
- ✧ 일관성 및 객관성: 소프트웨어 메트릭스는 항상 애매하지 않은 명백한 결과를 산출해야 한다. 독립된 소프트웨어 전문업체(Third Party)는 소프트웨어에 대한 동일한 정보를 사용하여 동일한 소프트웨어 메트릭스 값을 유도해낼 수 있어야 한다[10].
- ✧ 프로그래밍 언어 독립성: 소프트웨어 메트릭스는 분석모형, 설계 모형 또는 프로그램 구조에 기초를 두어야 한다. 관련성이 적은 프로그래밍 언어 구문이나 의미에 의존해서는 의존해서는 안 된다[10].
- ✧ 품질 피드백을 위한 효과적인 메커니즘: 소프트웨어 메트릭스는 소프트웨어 엔지니어에게 한층 높은 품질의 최종 제품을 만들어 낼 수 있도록 정보를 제공해야 한다[10].

3.4 평가 결과 표기

앞에서 제시한 평가기준에 따라 Use Case 다이어그램 분석을 통한 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 비교 분석 결과를 <표 1>로 나타낸다.

<표 1> 규모 예측 방법 총괄 요약 분석표

측면	평가요소	Karner	Marchesi
OOSD	일관성 및 연속성		
SMET	노력, 시간, 비용		
SMET	정형화		
	단순성 및 계산성		
	일관성 및 객관성		
	프로그래밍 언어 독립성		
	품질 피드백을 위한 효과적인 매커니즘		

또한 각 방법이 평가 요소의 지원 정도를 네 가지로 나누어 다음과 같이 표기한다.

◉ : 잘 지원함
○ : 어느 정도 지원 함
Δ : 약간의 관계가 있음
× : 지원 안함

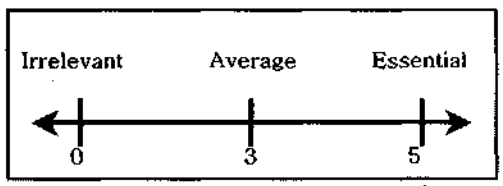
4. 규모 예측 방법 분석

본 장에서는 Use Case 다이어그램에서 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법 두 가지를 앞에서 제시한 평가기준에 따라 IMPS Use Case 다이어그램을 적용하여 비교 분석한다.

4.1 Karner 방법에 의한 분석

Karner 방법을 IMPS 개발에 적용할 때에 사용되는 TCF는 IMPS가 제공하는 각 기능의 우선순위(Precedence and Priority) [4, 5], Use Case에 공통적으로 나타나는 시스템의 기능성, 사용자 측면에서 요구되는 사용성, 시스템의 고장 및 수리시간 기준으로 요구되는 신뢰성, 서비스 처리 시간 위주로 요구되는 시스템의 성능, 시스템 개발 후 지속적으로 추가되어야 하는 지원가능성, 시스템 개발 시에 지켜야 할 설계 제약사항[2, 3] 등의 여섯 가지 측면을 가진다.

TCF 인자 항목의 가중치는 0부터 5까지 범위를 가지는데, 먼저 IMPS가 제공하는 각 기능의 우선순위(Precedence and Priority) 측면의 인자 항목 T1, T2, T3, T4, T5, T6는 IMPS의 각 기능에 대한 우선순위 별로 인자 항목의 가중치를 부여한다. 또한 Use Case에 공통적으로 나타나는 시스템의 기능성 측면의 인자 항목 T7, T8, T9, T10, 사용자 측면에서 요구되는 사용성 측면의 인자 항목 T11, T12, 시스템의 고장 및 수리시간 기준으로 요구되는 신뢰성 측면의 인자 항목 T13, T14, 서비스 처리 시간 위주로 요구되는 시스템의 성능 측면의 인자 항목 T15, T16, T17, T18, T19, T20, T21, T22, 시스템 개발 후 지속적으로 추가되어야 하는 지원가능성 측면의 인자 항목 T23, 시스템 개발 시에 지켜야 할 설계 제약사항 측면의 인자 항목 T24, T25는 IMPS의 서비스를 위해 요구되는 관련성의 정도에 따라 가중치를 부여한다.

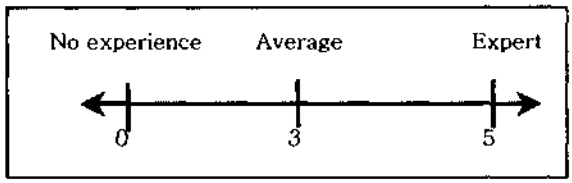


<그림 3> TCF의 가중치 범위

<표 2> 시스템에 관한 테크니컬한 가중인자

인자 순번	인자 항목	가중치	초기 단계	정교화 단계
T1	처방전 송신	2	5	5
T2	처방전 수신	2	5	5
T3	처방전 조회	1	1	1
T4	처방전 취소	1	1	1
T5	대체조제	1	1	1
T6	약국지정	1	1	1
T7	시스템부가기능 사용	0.5	0.5	0
T8	데이터코드 관리	0.5	0.5	0.5
T9	시스템 상태 조회	0.5	0.5	0.5
T10	사용자 관리	0.5	0.5	0.5
T11	Remote Access	1	5	5
T12	Security Insurance	1	3	3
T13	Ease of use	0.5	1	0
T14	Windows Compliance	0.5	0.5	0.5
T15	Availability	1	3	3
T16	Mean Time Between Failure	1	3	3
T17	Mean Time To Repair	1	3	3
T18	Response Time	1	3	3
T19	Throughput	1	3	3
T20	Capacity	1	3	3
T21	Degradation Modes	0.5	3	3
T22	Resource Utilization	0.5	0.5	0
T23	Version Upgrade	0.5	1	0
T24	Java Compatibility	0.5	3	3
T25	Tool Requirements	0.5	0.5	0.5

또한 EF의 인자 항목 역시 0에서 5 까지 범위에서 가중치를 평가한다.



<그림 4> EF의 가중치 범위

<표 3> 환경적인 가중인자

인자 순번	인자 항목	가중치	초기 단계	정교화 단계
F1	Familiar with Rational Unified Process	1.5	3	3
F2	Good Domain Knowledge	2	5	3
F3	Stable Requirements	2	5	3
F4	Application Experience	0.5	0.5	0.5
F5	Object-oriented Experience	0.5	0.5	0.5
F6	Component-based Experience	0.5	3	5
F7	Lead Analyst Capability	-1	3	3
F8	Part-time Workers	-1	0	1
F9	Difficult Programming Language	1	0.5	0.5
F10	Development Tool Used Experience	0.5	0.5	0.5

Karner 방법을 IMPS 개발 초기 단계와 정교화 단계에서의 Use Case 다이어그램에 적용하여 산출한 결과값은 다음과 같다.

<초기 단계>

Total Actor Weight = 7 Complex * 3 = 21

Total Use Case Weight = 7 Simple * 5 + 3 Average * 10 = 65

UUCP = 21 + 65 = 86
TCF = 0.6 + (0.01 * 55.75) = 1.1575
EF = 1.4 + (-0.03 * 24.25) = 0.6725
UCP = 86 * 1.1575 * 0.6725 = 66.9440125
Project Estimate = 67 * 20 = 1340

<정교화 단계>
Total Actor Weight = 7 Complex * 3 = 21
Total Use Case Weight = 24 Simple * 5 + 2 Average * 10 + 1 Complex * 15 = 155
UUCP = 21 + 155 = 176
TCF = 0.6 + (0.01 * 54.25) = 1.1425
EF = 1.4 + (-0.03 * 16.25) = 0.9125
UCP = 176 * 1.1425 * 0.9125 = 183.4855
Project Estimate = 184 * 20 = 3680

4.1.1 객체지향 소프트웨어 개발 측면

이 방법에 따라 산출한 결과값은 시스템이 진화할 때의 다시 말해서 시스템 개발 초기 단계와 정교화 단계에서의 Use Case 다이어그램뿐 아니라 TCF와 EF를 통해서 시스템 요구사항의 변경에 따른 소프트웨어 메트릭스 결과 값의 변경을 잘 반영해주고 있다.

4.1.2 규모 예측 방법 측면

이 방법은 앞에서도 언급한 바와 같이 UCP라는 Use Case 기반의 프로젝트를 완료할 동안 인력(Man-Hour)을 예측하는 방법으로, 개발 초기에 사용하며 기능 점수 또는 COCOMO와 같은 그 밖의 예측 방법과 함께 프로젝트 동안의 인력(Man-Month) 수를 구하는데 사용된다.

4.1.3 소프트웨어 메트릭스 측면

이는 Karner의 경험에 의한 연구를 바탕으로 한 방법으로, 액터, Use Case, TCF, EF 각각의 가중인자에 정의와 그에 따른 타입 값을 부여하여 정형화된 소프트웨어 메트릭스이다. 가중인자 정의와 그에 따른 타입 값의 부여 방식이나 방법에 따른 결과값 산출 방식은 비교적 간단하고 계산하기 용이하다. 그러나 동일한 도메인에 있어서 TCF와 EF는 시스템을 개발하는 프로젝트의 개발 방법론, 개발 업체에 따라 산출되는 결과값은 달라질 수 있다. 또한 <표 2>와 <표 3>에서와 같이 TCF와 EF에 있어서 프로그래밍 언어에 의존적인 가중인자가 포함가능하다. 이 방법은 프로젝트 완료할 동안의 인력을 예측하는 방법으로 프로젝트에 대한 작업의 양을 가늠하는 시작점으로서 프로젝트 참여자에게 도움이 된다.

4.2 Marchesi 방법에 의한 분석

Marchesi 방법을 IMPS 개발 초기 단계와 정교화 단계에서의 Use Case 다이어그램에 적용하여 산출한 결과값은 다음과 같다.

<초기 단계>
UC1 = 10
UC2 = 18
UC3 = 18
UC4 = 28

<정교화 단계>
UC1 = 27
UC2 = 39

UC3 = 32
UC4 = 38.1

4.2.1 객체지향 소프트웨어 개발 측면

이 방법에 따라 산출한 결과값은 시스템이 진화할 때, 다시 말해서 시스템 개발 초기 단계와 정교화 단계에서의 Use Case 다이어그램을 통해서 시스템 요구사항의 변경에 따른 소프트웨어 메트릭스 결과 값의 변경을 반영해주고 있다. 또한 Use Case간의 ≪extends≫, ≪include≫ 관계의 수가 커짐에 따라 시스템의 복잡도가 커지는 것을 양적으로 보여준다.

4.2.2 규모 예측 방법 측면

이는 소프트웨어 규모의 복잡도 관점으로 측정하는 방법으로, 산출한 결과 값 다시 말해서 시스템의 복잡도는 Marchesi가 Use Case 메트릭스 외에 제안한 클래스 메트릭스나 패키지 메트릭스를 측정하는데 사용되며 Use Case 메트릭스에서 직접적으로 시간이나 노력 및 비용을 예측할 수 없다.

4.2.3 소프트웨어 메트릭스 측면

이 방법은 Use Case 다이어그램 상에서 정량적인 측정을 통해 결과값을 산출가능한 소프트웨어 메트릭스로, 결과 값을 산출하는 공식과 파라미터와 상수에 대한 명확한 정의가 나와있으며 그에 따른 결과값 산출 방식은 비교적 간단하고 계산하기 용이하다 있다. 앞의 Karner 방법의 TCF와 EF와 같이 소프트웨어 개발 업체마다 다를 수 있는 인자가 없으므로, 동일한 소프트웨어 정보를 가진 개발 업체들은 동일한 소프트웨어 메트릭스 결과 값을 유도해낼 수 있다. 또한 프로그래밍 언어에 종속적인 인자가 없으며 Use Case 메트릭스에서 산출된 시스템의 복잡도는 개발 노력, 구현 시간과 개발 중 비용에 대한 초기 예측을 하는 Marchesi가 제안한 클래스 메트릭스와 패키지 메트릭스를 측정하는데 이용된다.

5. 비교 분석 결과

Use Case 다이어그램 상에서 소프트웨어 규모 예측을 위해 제안된 두 가지 방법을 IMPS 개발 사례를 적용하여 앞에서 제시한 평가기준의 지원정도를 비교 분석하면 <표 4>와 같다. 또한 비교 분석한 결과는 앞에서 제시한 IMPS 개발 사례에 적용시의 결과 예측과 동일하다.

<표 4> 규모 예측 방법 총괄 요약 분석표

측면	평가요소	Karner	Marchesi
OOSD	일관성 및 연속성	◉	◉
SMET	노력, 시간, 비용	◉	Δ
SMET	정형화	◉	◉
	단순성 및 계산성	◉	◉
	일관성 및 객관성	×	◉
	프로그래밍 언어 독립성	×	◉
	품질 피드백을 위한 효과적인 매커니즘	◉	Δ

5.1 객체지향 소프트웨어 개발 측면

먼저 Karner 방법은 시스템 개발 초기 단계에서의 Use Case 10개가 정교화 단계

에서의 Use Case 27개로 변화함에 따라서 그에 따른 Use Case Weight도 변화게 된다.

Marchesi 방법은 시스템 개발 초기 단계에서의 Use Case 10개가 정교화 단계에서의 Use Case 27개로 변화함에 따라서 그에 따른 UC1, UC2의 값이 변화게 되고 특히 Use Case와 Use Case 간에 ≪extends≫, ≪include≫ 관계의 의사소통의 수가 변화함에 따라 독립적인 의사소통에 대한 복잡도 UC3와 포괄적인 복잡도를 UC4가 변화게 된다.

5.2 규모 예측 방법 측면

Karner 방법은 프로젝트를 완료할 동안 인력(Man-Hour)을 예측하는 방법이다. 이에 반해 Marchesi 방법에서 산출한 복잡도는 직접적으로 시간이나 노력 및 비용을 예측할 수 없다.

5.3 소프트웨어 메트릭스 측면

Karner 방법은 Use Case 외에도 시스템 개발 초기 단계와 정교화 단계에서 TCF, EF와 같은 가변적인 시스템 개발 환경 다시 말하면 개발 방법론에 따라 산출되는 결과값이 변화함에 따라 UCP의 값이 변화게 된다. 따라서 동일한 소프트웨어 도메인에 대해 서로 다른 독립된 소프트웨어 전문업체는 개발 방법론 및 자신의 조직의 프로세스 성숙도에 따라 산출되는 소프트웨어 메트릭스 결과 값은 각기 다를 수 있다.

이에 반해 Marchesi 방법은 Karner 방법과 달리 가변적인 시스템 개발 환경을 가중인자로 포함하지 않고 Use Case 만으로 결과값을 산출함으로써 동일한 소프트웨어 도메인에 대해 서로 다른 독립된 소프트웨어 전문업체는 동일한 소프트웨어 메트릭스 결과 값을 산출 가능하다.

6. 결과에 따른 개선점 도출

앞의 비교 분석 결과에서도 언급한 바와 같이 Karner 방법은 TCF, EF와 같은 가변적인 시스템 개발환경 다시 말해서 개발 방법론에 따라서 산출되는 결과 값도 달라질 수 있다. 또한 산출된 UCP 당 20 인력(Man-Hour) 인자를 사용하여 프로젝트를 예측하도록 제시하고 있는데, 이는 Karner의 개인적인 경험에 따른 것이며 또한 국내 소프트웨어 개발 성숙도를 고려하지 않고 있다. 따라서 개발 프로젝트를 수행하는 팀 또는 개발 업체의 업무 수행 성숙도에 따라 가중인자를 달리해야 한다.

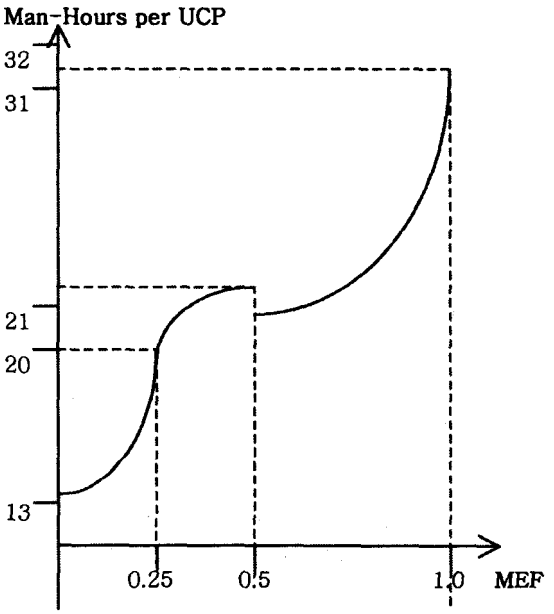
EF는 프로젝트 참여자의 경험 수준과 프로젝트의 안정성(Stability)에 관련한다. 따라서 EF 인자 항목의 가중치(F_i)에 따른 개발 프로젝트를 수행하는 팀 또는 개발 업체의 업무 수행 성숙도(MEF; Maturity of EF)를 다음과 같이 제시한다.

About F_i (only, $i = 1, \ldots, n$ and $X = 0, Y = 0$):

$$\begin{cases} \text{if } F_i \geq 0 \ \wedge \ \text{Assigned value of } F_i < 3, \\ \quad \text{then } X = X + 1 \\ \text{if } F_i < 0 \ \wedge \ \text{Assigned value of } F_i \geq 3, \\ \quad \text{then } Y = Y + 1 \end{cases}$$

$$\text{MEF(Maturity of EF)} = (X + Y) / n$$

또한 산출한 MEF 값에 따라 프로젝트 예측을 위한 UCP 당 인력(Man-Hour) 인자의 값도 달리한다. MEF 비율에 따른 프로젝트 예측을 위한 UCP 당 인력(Man-Hour) 인자의 비율을 다음과 같이 제시한다.



<그림 5> MEF에 따른 UCP당 인력 비율

또한 시스템 개발을 위해 프로젝트 참여자를 대상으로 하는 교육 및 훈련을 위한 시간은 별도로 추가하도록 한다.

이와 같이 개발 프로젝트를 수행하는 팀 또는 개발 업체의 업무 수행 성숙도에 따른 UCP 당 인력을 프로젝트를 예측에 적용한 결과 서로 다른 독립된 소프트웨어 전문업체는 자신의 프로젝트 팀의 업무 수행 성숙도에 따라 좀더 자세히 소프트웨어 규모 예측할 수 있게 해준다.

Marchesi 방법은 객체지향 분석 단계에서 기술되는 Use Case 다이어그램, 클래스 다이어그램 상에서 적용 가능한 Use Case 메트릭스, 클래스 메트릭스, 그리고 패키지 메트릭스를 제시하고 있다. 따라서 이 세가지 메트릭스를 모두 적용함으로써 유스케이스의 개수 및 유스케이스 간의 의사소통 개수에 따른 시스템의 복잡도는 물론 하나의 유스케이스에서 유도될 수 있는 클래스의 양을 가늠하도록 한다. 또한 시스템의 기능성에 대한 모델인 Use Case 뷰뿐만 아니라 애플리케이션 도메인에 대한 정적 뷰를 표현하는 클래스 다이어그램과 관련지어 개발 노력, 구현시간, 개발 비용과 객체지향성과 품질을 예측 가능하도록 한다.

또한 사용자와 개발자, 분석가 등 넓은 범위의 이해관계자는 Use Case 다이어그램을 통해서 서로 시스템에 대한 기능성을 서로 표현가능하며 소프트웨어의 규모를 예측 가능하게 된다. 프로젝트 관리자에게 있어서, 노력과 시간 및 비용을 개발 초기에 예측을 가능하게 함으로서 프로젝트 계획단계에서 작업 네트워크(Task Network), 타임라인 차트(Timeline Chart, Gantt Chart), 직원 구조(Staff Organization) 및 팀 구조(Team Structure) 등의 프로젝트 일정을 결정하는 중요한 정보를 제공한다[14]. 소프트웨어 엔지니어에게는 명확하게 정의된 규칙들의 집합에 기초해서 품질을 평가하는 체계적인 방법을 제공해주며 제품을 구축하기 전에 품질을 평가할 수 있게 해준다.

7. 결론 및 향후 연구

지금까지 시스템 개발초기 단계에서 주로 작성되는 Use Case 다이어그램에서 소

프트웨어의 규모를 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 IMPS 개발사례를 적용하여 제시한 평가기준에 따라 비교 분석하고 비교 분석 결과에 따른 예측 방법에 대한 개선점을 도출하였다.

제안된 소프트웨어 규모 예측 방법은 절대적인 것이 아니기 때문에 논쟁의 여지는 항상 남아 있다. 객체지향 시스템 개발 초기 단계에서의 소프트웨어 규모 예측 방법은 이 단계에서 도출된 산출물과 도메인 지식을 바탕으로 예측이 가능해야 하며, 사용자와 개발자, 분석가 등 넓은 범위의 이해관계자가 쉽게 이용할 수 있고 유용한 정보를 제공해 주는 방법의 선택이 중요하다.

본 논문에서는 Use Case 다이어그램에서의 규모 예측을 위해 제안된 두 가지 방법으로 범위를 한정하여 비교 분석하였다. 좀 더 범위를 넓혀 객체지향 소프트웨어 시스템의 규모 예측을 위해 제안된 여러 방법들을 수집하고 비교 분석하여 여기서 제시된 사항들을 보완하는 일이 향후에 계속되어야 할 연구 과제이다.

참고문헌

[1] 서예영, 이남용, "Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템 규모 예측 방법에 대한 연구," *제28회 정보과학회 임시총회 및 춘계학술발표회*, 논문집(A), pp.580~582, 2001년 4월

[2] 신정은, 김태형, "IMPS-Supplementary Specification Version 0.9(Draft)," 로코즌㈜, 2001년 3월

[3] 신정은, 김태형, 이남용, "IMPS-Supplementary Specification Version 0.9.2(Draft)," 로코즌㈜, 2001년 6월

[4] 신현정, 윤병권, "IMPS-Vision Document Version 0.9(Draft)," 로코즌㈜, 2001년 3월

[5] 신현정, 윤병권, 이남용, "IMPS-Vision Document Version 0.9.2(Draft)," 로코즌㈜, 2001년 6월

[6] G.. Booch, *Object-Oriented Analysis and Design with Applications*, Benjamin-Cummings, 1994

[7] G.. Schneider and J. P. Winters, *Applying Use Cases: A Practical Guide*, Addison-Wesley, 1998

[8] I. Jacobson, M. Christerson, P. Jonsson, and G.. Övergaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992

[9] J. D. McGregor, "Managing Metrics in an Iterative Incremental Development Environment," *Object Magazine*, October 1995, pp. 65-71

[10] L. Ejiogu, *Software Engineering with Formal Metrics*, QED Publis ing, 1991

[11] M. Marchesi, "OOA Metrics for the Unified Modeling Language," *2nd Euromicro Conference on Software Maintenance and Reengineering*, March 1998

[12] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd Edition, International Thomson Computer Press, 1997

[13] P. Kruchten, *The Rational Unified Process: An Introduction*, Addison-Wesley, 1999
[14] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 4th Edition, Mcgraw-hill, 1997

저자소개

서예영(e-mail : yysuh@selab.soongsil.ac.kr)

2000년 숭실대학교 컴퓨터학부 공학사
2000년 ~ 현재 숭실대학교 대학원 컴퓨터학과 석사과정

이남용(e-mail: nylee@computing.soongsil.ac.kr)

1979년 숭실대학교 전자계산학 학사
1983년 고려대학교 경영대학원 경영정보학 석사
1993년 Mississippi State University 경영정보학 박사
1999 ~ 현재 숭실대학교 컴퓨터학과 조교수