Use Case 다이어그램에 의한 객체지향 소프트웨어 시스템의 규모 예측 방법에 대한 연구

서예영[0] 이남용
숭실대학교 컴퓨터학부
yysuh@selab.soongsil.ac.kr, nylee@computing.soongsil.ac.kr

A Study of Theoretical Comparison on Size Estimation Techniques for Object-Oriented Software Systems by Use Case Model

Ye-Young Suh[0] Nam-Yong Lee
Dept. of Computing, Soongsil University

요 약

현재 소프트웨어 개발 주기의 초기 단계에서 소프트웨어의 개발과 유지보수를 위한 비용과 노력을 미리 예측하는 소프트웨어 규모 예측 방법이 요구되고 있다. 이에 따라 소프트웨어 규모 예측 방법을 위한 수백개의 메트릭스가 제안되고 있지만, 난해하고 복잡한 측정 방법으로 인해 소수의 실제 전문가에 의해 사용되고 있다. 이용할 해당 메트릭스의 장점과 단점을 파악하고 적용시켜야 시행착오를 피할 수 있다. 본 논문에서는 객체지향 시스템 분석 단계에서 주로 작성되는 UML Use Case 다이어그램 분석을 통해 소프트웨어 규모 예측을 하는 방법들을 비교 분석한다. 이를 따르면 메트릭스를 적절히 사용하여 보다 효율적인 소프트웨어 프로젝트 관리를 할 수 있을 것이다.

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 있으며 이에 따른 예측 방법이 다양하게 제안되고 있다.

이에 따른 소프트웨어 규모 예측을 위한 수백개의 메트릭스가 제안되고 있지만, 모든 메트릭스가 현재 소프트웨어 엔지니어에게 유용하게 지원되지 않고 있는 실정이다. 어떤 메트릭스는 너무 복잡한 측정을 요구하고 또 어떤 메트릭스는 너무 난해해서 소수의 실제 전문가만이 이해할 수 있고 고품질의 소프트웨어에 대한 기본적인 통찰력을 어지럽히기 때문이다[5]. 사용자가 이용하고자 하는 메트릭스의 장점과 단점을 정확히 파악하고 적용시켜야 시행착오를 피할 수 있다. 따라서 다양한 소프트웨어 규모 예측 방법에 대한 비교 분석이 필요하다. 이를 위해 본 논문에서는 객체지향기술(*Object-Oriented* Technology)에 의한 객체지향 시스템 개발을 위해 OMG에서 제안한 객체지향 시스템 분석·설계 언어 UML(Unified Modeling Language)의 9가지 다이어그램 중 시스템 분석 단계에서 주로 작성되는 Use Case 다이어그램 분석을 통한 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 비교 분석하기 위한 평가기준을 제시하고 그에 따라 비교 분석한다.

논문의 구성은 2장에서는 규모예측 방법을 비교 분석할 평가기준을 제시하고, 3장에서는 2장에서 제시한 평가기준에 따라 2가지 소프트웨어 규모 예측 방법을 비교 분석하고, 4장에서 결론을 맺는다.

2. 평가기준

소프트웨어 규모(Size)는 길이(Length), 기능성(Functionality), 복잡도(Complexity) 세가지 관점을 가진다[4]. Use Case 모델은 시스템에 의해서 처리되는 모든 기능적인 요구사항을 표현하기 위한 것으로, 사용자와 개발자, 분석가 등 넓은 범위의 이해 관계자(Stakeholder)가 시스템에 대한 기능성을 표현하는 방법이다[8]. 프로젝트 관리에서, Use Case와 시나리오는 소프트웨어 개발 프로세스의 반복(Iteration)에 관한 내용을 정의하는데 사용된다. 기능 점수 분석(Function Point Analysis)과 같은 기법에 의해 개발 노력 또는 비용에 대한 예측이 Use Case의 서술로부터 도출될 수 있다[1].

본 논문에서는 Use Case 다이어그램 분석을 통한 소프트웨어 규모 예측 방법을 비교 분석할 평가 기준을 객체지향 소프트웨어 개발 프로세스의 반복적이고 점진적인 접근 방법에 따른 측면(OOSD: Object-Oriented Software Development), 규모예측 방법 측면(SIZE: SIZe Estimation)과 보편적인 소프트웨어 메트릭스가 포함해야 하는 속성 측면(SMET: Software METrics)으로 나누어 제시한다.

2.1 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[6,7]. 따라서 특정 활동에 관련된 메트릭스는 이전 활동의 메트릭스와 일관성을 유지하면서 그것에 근거를 두어야 한다.

- Consistent and Continuous: 메트릭스는 시스템이 진화할 때에, 일관성과 연속성을 유지하면서 재산출될 수 있어야 한다(시스템에 있어 "작은" 변화는 시스템에 대한 메트릭스에 있어 "작은" 변화를 이끌어야 한다).

2.2 규모 예측 방법(SIZE) 측면

- Effort, Time, and Cost: 소프트웨어 규모 예측을 하는 메트릭스가 측정하는 속성들로, 해당 메트릭스가 이 세가지 중 하나 이상을 지원가능해야 한다.

2.3 소프트웨어 메트릭스(SMET) 측면

- Formulable: 메트릭스의 수학적인 계산은 이상한 단위(unit) 조합을 만들지 않는 정형화된 메트릭스를 사용하여 측정가능해야 한다.
- Simple and Computable: 메트릭스 유도 방법이 비교적 배우기 쉽고 메트릭스의 계산이 과도한 노력이나 시간이 필요하지 않아야 한다[9].
- Consistent and Objective: 메트릭스는 항상 애매하지 않은 명백한 결과를 산출해야 한다. 독립된 소프트웨어 전문업체(third party)는 소프트웨어에 대한 동일한 정보를 사용하여 동일한 메트릭스 값을 유도해낼 수 있어야 한다[9].
- Programming Language Independent: 메트릭스는 분석모형, 설계 모형 또는 프로그램 구조에 기초를 두어야 한다. 엉뚱한 프로그래밍 언어 구문이나 의미에 의존해서는 안된다[9].
- An Effective Mechanism for Quality Feedback: 메트릭스는 소프트웨어 엔지니어에게 한층 높은 품질의 최종 제품을 만들어 낼 수 있도록 정보를 제공해야 한다[9].

2.4 평가 결과 표기

각각의 규모 예측 방법의 비교 분석 결과를 아래의 표로 요약한다.

(표 1) 규모 예측 방법 총괄 요약 분석표

측면	평가요소	Karner	Marchesi
OOSD	Consistent & Continuous		
SIZE	Effort, Time, & Cost		
SMET	Formulable		
	Simple & Computable		
	Consistent & Objective		
	Programming Language Independent		
	An Effective Mechanism for Quality Feedback		

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

● : 잘 지원함
☺ : 어느 정도 지원 함
△ : 약간의 관계가 있음
Blank : 지원 안함

3. 규모 예측 방법 분석

본 장에서는 Use Case 다이어그램에서 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 앞 장에서 제시한 세가지 측면에서 각각의 방법을 비교 분석한다.

3.1 Karner 방법

Karmer[2]는 UCP(Use Case Points)라는 Use Case 기반의 프로젝트를 완료할 동안 인원-시간(man-hours)를 예측하는 방법을 제안했다. 이 방법은 개발 초기(preliminary)에 사용하며 기능 점수(Function Point) 또는 COCOMO 와 같은 그 밖의 예측 방법과 함께 프로젝트 동안의 인원-달(man-month)의 수를 구하기 위해 사용된다. UCP 는 시스템에 대한 액터(Actor)와 Use Case 의 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) 가중 인자를 사용하여 조정된다. 산출된 UCP 당 20 인원-시간 인자를 사용하여 프로젝트를 예측을 한다.

Karner 방법을 앞에서 제시한 평가 기준에 따라 세가지 측면으로 분석하면 다음과 같다.

1) OOSD 측면: 이 방법은 시스템이 진화할 때, Use Case 가중인자 중 Use Case 트랜잭션의 개수의 범위에 따라 타입이 달라지는데, 근소한 개수 차이가 동일한 타입에 포함될 경우와 그렇지 않은 경우에 적용되는 가중인자가 달라지게 된다. . 여기서 제시하고 있는 인자들 간의 값의 차이가 크므로, 이에 따른 메트릭스의 산출된 값에 차이가 생긴다.
2) SIZE 측면: 앞에서도 언급했듯이 이 방법은 UCP(Use Case Points)라는 인원-시간 및 인원-달을 예측하는 방법이다.
3) SMET 측면: 이 방법은 경험에 의한 연구를 바탕으로 한 정형화된 메트릭스이며 가중 인자들의 정의와 그에 따른 타입의 값이 명확하게 나와있다. 프로젝트 참여한 사람들의 경험 수준을 나타내는 EF 인자는 소프트웨어 개발 업체마다 다를 수 있다. 따라서 동일한 소프트웨어 정보를 가졌다 하더라도 메트릭스의 산출된 값이 서로 다를 수 있다. 프로젝트에 관한 복잡도를 나타내는 TCF 인자 중에 코드 재사용 가능성에 대해 언급하고 있다. UCP는 프로젝트 완료할 동안의 인원-시간을 예측하는 방법으로 프로젝트에 대한 작업의 양을 가늠하는 시작점으로서 프로젝트 참여자에게 도움이 된다.

3.2 Marchesi 방법

Marchesi[3]는 객체지향 분석 단계(Object-Oriented Analysis) 동안 UML Use Case 다이어그램과 클래스 다이어그램에 관련한 메트릭스를 제안했다. 제안된 메트릭스는 개발 노력, 구현 시간과 개발 중 비용에 대한 초기 예측을 허용하며, 분석 단계이후 지속적인 객체지향성(Object-Orientedness)과 품질 측정을 목적으로 한다. 그 중 여기서 비교 분석할 대상은 요구사항 유도 단계(Requirements Elicitation Phase)에서 Use Case 다이어그램 상의 시스템 복잡도에 대한 초기예측을 목적으로 하는 Use Case 메트릭스로 UC1, UC2, UC3, UC4 네 가지로 나뉜다. UC1은 Use Case 다이어그램 상의 전체 Use Case의 개수,

UC2는 Use Case와 액터 간의 의사소통(Communication)의 개수, UC3는 Use Case와 액터 간에 ≪extends≫, ≪include≫ 관계와 같은 중복을 가지지 않는 의사소통의 수로 시스템의 복잡도를 나타낸다. UC4는 총 Use Case의 개수 UC1과 ≪extends≫, ≪include≫ 관계의 의사소통 정도를 고려함으로써 UC3를 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. 다시 말해 독립적인 의사소통에 대한 복잡도를 UC3로, 포괄적인 복잡도를 UC4로 예측한다.

Marchesi 방법을 앞에서 제시한 평가 기준에 따라 세가지 측면으로 분석하면 다음과 같다.

1) OOSD 측면: 이 방법은 시스템이 진화할 때, ≪extends≫, ≪include≫ 관계가 없는 의사소통의 수가 증감할 때와 그런 관계가 있는 의사소통의 수가 증감할 때에 따라 산출되는 메트릭스의 결과가 달라진다.
2) SIZE 측면: 이 방법에서 Use Case 메트릭스에서 산출된 시스템의 복잡도는 클래스 메트릭스와 패키지 메트릭스를 측정하는데 사용되며 Use Case 메트릭스에서 직접적으로 시간이나 노력 및 비용을 예측할 수 없다.
3) SMET 측면: 이 방법은 UML 다이어그램 상에서 정량적인 측정을 통해 산정 가능하도록 제안된 메트릭스로, 메트릭스의 공식과 파라미터와 상수들의 정의가 명확하게 나와있다. 앞의 Karner 방법의 프로젝트 참여한 사람들의 경험 수준을 나타내는 EF 인자와 같이 소프트웨어 개발 업체마다 다를 수 있는 인자가 없으므로, 동일한 소프트웨어 정보를 가진 개발 업체들은 동일한 메트릭스 값을 유도해낼 수 있다. 또한 프로그래밍 언어에 대해 언급하는 인자가 없으며 Use Case 메트릭스에서 산출된 시스템의 복잡도는 개발 노력, 구현 시간과 개발 중 비용에 대한 초기 예측을 하는 클래스 메트릭스와 패키지 메트릭스를 측정하는데 이용된다.

3.3 비교 분석 결과

Use Case 다이어그램 상에서 소프트웨어의 규모 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 앞에서 제시한 평가 요소의 지원 정도를 네 가지로 나누어 비교 분석하였다.

(표 2) 규모 예측 방법 총괄 요약 분석표

측면	평가요소	Karner	Marchesi
OOSD	Consistent & Continuous	△	☺
SIZE	Effort, Time, & Cost	☺	
SMET	Formulable	●	●
	Simple & Computable	●	●
	Consistent & Objective		●
	Programming Language Independent	△	●
	An Effective Mechanism for Quality Feedback	●	△

Karner 방법에서 프로젝트 참여한 사람들의 경험 수준을 나타내는 EF 인자는 소프트웨어 개발 업체마다 다를 수 있기 때문에 프로젝트 참여자의 경험과 개발 프로젝트의 안전성(Stability)을 고려하여 가중 인자를 적용하여야 한다. 또한 개발 환경을 가중인자로 가지는 프로젝트에 관한 테크니컬한 복잡도 TCF는 시간이 지남에 따라 시대성이 떨어지므로, 일정 기간이 지났을 때의 보완 및 개정이 요구된다. Marchesi 방법은 Use Case 다이어그램 만으로 소프트웨어 규모 예측을 하기에 충분한 정보가 제공되지 않는다. 따라서 객체지향 분석 단계에 분석단계에서 클래스 다이어그램과 관련지어 소프트웨어 규모를 예측할 때에는 Marchesi 방법이 적합하다.

4. 결론 및 향후 연구 과제

지금까지 시스템 개발초기 단계에서 주로 작성되는 Use Case 다이어그램에서 소프트웨어의 규모를 예측을 위해 제안된 Karner 방법과 Marchesi 방법을 객체지향 소프트웨어 개발, 규모 예측 방법, 소프트웨어 메트릭스 측면으로 비교 분석하였다.

제안된 소프트웨어 규모 예측 방법은 절대적인 것이 아니기 때문에 논쟁의 여지는 항상 남아 있지만, 소프트웨어 엔지니어에게 명확하게 정의된 규칙들의 집합에 기초해서 품질을 평가하는 체계적인 방법을 제공해주며 제품을 구축하기 전에 품질을 평가할 수 있게 해준다. 일반적으로 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 이해 관계자가 쉽게 이용할 수 있고 유용한 정보를 제공해 주는 방법의 선택이 중요하다. 또한 프로젝트 관리자 입장에서 프로젝트 계획은 물론 의사결정에 유용한 정보를 제공해주어야 한다.

향후 연구에서는 본 논문에서 비교 분석한 두 가지 방법을 객체지향 소프트웨어 시스템 개발 프로젝트에 적용하여 평가 기준과 그에 따라 비교 분석한 결과를 좀 더 명확하고 효과적으로 제시하도록 보완 연구가 필요하다.

참고문헌

[1] P. Kruchten, *The Rational Unified Process: An Introduction*, Addison-Wesley, 1999
[2] G. Schneider and J. P. Winters, *Applying Use Cases: A Practical Guide*, Addison-Wesley, 1998
[3] M. Marchesi, " OOA Metrics for the Unified Modeling Language," *2nd Euromicro Conference on Software Maintenance and Reengineering, March 1998*
[4] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd Edition, International Thomson Computer Press, 1997
[5] R. S. Pressman, *Software Engineering: A Practitioner' s Approach*, 4th Edition, Mcgraw-hill, 1997
[6] J. D. McGregor, " Managing Metrics in an Iterative Incremental Development Environment," *Object Magazine, October 1995, pp. 65-71*
[7] G. Booch, *Object-Oriented Analysis and Design with Applications*, Benjamin-Cummings, 1994
[8] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992
[9] L. Ejiogu, *Software Engineering with Formal Metrics*, QED Publishing, 1991