요구사양서의 품질평가 모델에 관한 연구

김 경 환[†] · 코지마 츠토무[††] · 박 용 범[†††]

요 약

소프트웨어 응용 분야의 확대로 인해, 소프트웨어는 더 크고, 더 복잡해지고 있다. 게다가 개발기간의 단축 및 비용의 절감·품질의 향상 또한 요구되고 있다. 이러한 요구를 충족시키기 위해 여러 효율적인 방법이 제안되었고, 구현되었다. 대표적으로 프로세스 개선과 객체지향 개발, 요구공학, 소프트웨어 메트릭스 등이 있다. 이러한 수많은 방법 중에서 요구공학은 고품질의 소프트웨어 개발의 기반이 된다. 즉, 고품질의 소프트웨어를 개발하기 위해서는 먼저 요구를 획득하고, 기술하여야 하며, 이에 대한 검증과 관리를 통해 요구사양서의 품질을 향상시켜 가야 한다. 본 논문에서는 IEEE Std-830-1998의 좋은 요구사양서가 가져야 할 특성을 중심으로, 요구사양서에 기술된 내용 자체를 정량적으로 품질평가할 수 있는 메트릭스를 제안하였다.

A Study on Quality Evaluation and Improvement of Software Requirement-Specification

Kyong-Hwan Kim[†] · Tsutomu Kozima[††] · Young B. Park[†††]

ABSTRACT

As the area of software application is increased, the software is becoming larger and more complex. In addition, development of the high quality software within the limits of the budget is strongly demanded. Many methodologies, such as software process improvement, object-oriented development, requirement-engineering, and software metrics have been introduced in the software development process in order to attain such objectives. Among those techniques, Requirements-Engineering gives basic guideline to develop high quality software. In other word, in order to develop high quality software, requirements should be elicited, and described. And with proper reviews and management, the quality of requirements can be improved. In this paper, quantitative measurement method that is based on IEEE Std-830-1998 for the requirement-specifications is proposed.

키워드 : 품질평가(Quality Evaluation), 메트릭스(Metrics), 요구사양서(Requirement Specification)

1. 서 론

근래 소프트웨어의 응용 분야의 확대와 함께 소프트웨어가 더욱 커지고 복잡해지고 있으며 또한 개발기간의 단축, 비용의 절감, 고품질의 소프트웨어 생산이 강하게 요구되고 있다. 이러한 요구를 실현하기 위해 소프트웨어 공학에서는 수많은 방법이 제안되어 왔다. 대표적인 것으로 프로세스 평가·개선모델(CMM/CMMI, SPICE, ISO 9000시리즈 등), 컴포넌트를 사용한 소프트웨어 재사용, 정량적으로 품질을 평가할 수 있는 소프트웨어 메트릭스 등을 들 수 있다. 하지만 고품질의 소프트웨어를 개발하는데 있어 가장 중요한 것 중의 하나가 사용자의 요구를 잘 정의 하는 것으로 요구를 잘 정의하기 위해서는 먼저 요구를 획득하고, 기술하고, 검증·관리 등의 방법을 반복 적용하지 않으면 안 된다[2, 8]. 이 같은 반복·적용으로 소프트웨어의 품질은 향상 된다. 일반적으로 품질의 향상을 인식하기 위해서는 정성적인 평가만이 아니고, 어느 정도 축적된 데이터에 근거해 현재의 상황을 객관적이고 정량적으로 평가할 수 있는 메트릭스가 필요하다.

소프트웨어 메트릭스는 소프트웨어 산출물(product)의 여러 특성(복잡도, 신뢰성, 효율)을 판별하는 객관적인 수학적 척도이다[1]. 소프트웨어 공학의 관점에서 품질평가는 분석단계부터 구현단계까지 단계별 메트릭스를 적용한다. 예를 들어, 분석단계에서는 요구사양서에서 소프트웨어의 기능의 크기를 측정하는 기능점(Function Point) 방법이 제안되었고, 설계·구현단계에서는 설계서와 소스 코드로부터 복잡도를 측정하는 McCabe의 순환수(Cyclomatic Number)와

† 종신회원 : 일본 Software Research Associates
†† 비 회 원 : 일본 프로세스 개선 연구원
††† 종신회원 : 단국대학교 전자컴퓨터학과 교수
논문접수 : 2004년 7월 22일, 심사완료 : 2004년 10월 22일

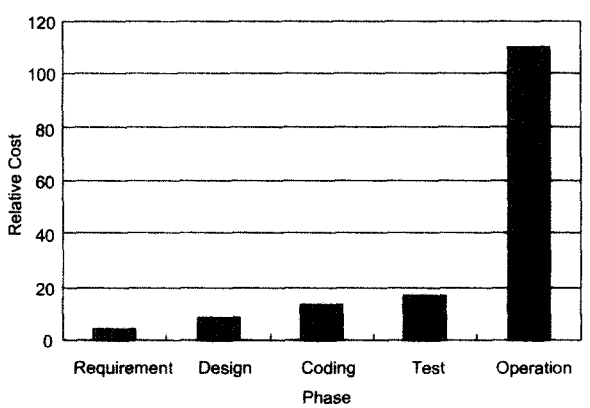
Halstead의 소프트웨어 과학(Software Science)등이 제안되었다. 지금까지 제안된 수많은 메트릭스의 대부분이 소스 코드를 대상으로 한 메트릭스로, 소프트웨어 요구사양서(Software Requirement Specification, 이하 요구사양서) 자체를 대상으로 한 메트릭스는 거의 존재하지 않는다.

본 논문에서는 요구사양서를 측정 대상으로 사용자의 요구가 어느 정도 잘 정의 되어 있는가를 정량적으로 평가할 수 있는 방법을 제안하였다. 이 방법에 의해 요구사양의 현재 및 장래의 잠재적인 문제를 알아낼 수 있고, 또한 이러한 문제에 대해 조기 대책을 강구할 수 있다. 여기에서 품질 척도는 IEEE Std-830-1998[4] 표준에 정의되어 있는 좋은 요구사양서가 가져야 할 특성에 근거하였다.

2. 관련 연구

2.1 요구 사항 분석의 문제점

요구사항이 잘못되어 있을 경우 그 결과는 재작업(rework)으로 나타난다. 재작업을 단순하게 정의하면 처음부터 바르게 작업이 이루어지지 않았기 때문에 산출물을 다시 만드는 것이다. 재작업의 대부분은 산출물의 결함과 잘못된 요구 그리고 사양을 준수하고 있지 않기 때문에 발생한다. 재작업은 전형적인 프로젝트 비용의 30~50%을 소비하고, 잘못된 요구에 의해 발생된 재작업의 비용이 그 중 70% 이상을 차지한다고 한다[5, 8]. (그림 1)에서는 프로젝트가 진행되고 나서 발견된 결함을 수정하기 위해 사용되는 비용이 조기에 수정하는 비용보다 훨씬 소요되고 있음을 보여주고 있고, 요구의 결함을 조기에 발견해야 함을 시사하고 있다.



(그림 1) 요구결함을 수정하기 위한 발견시기에 따른 상대적 비용

2.2 소프트웨어 메트릭스

소프트웨어 메트릭스는 소프트웨어의 다양한 특성을 판별하는 객관적인 수학적 척도로서 소프트웨어의 산출물 특성만이 아니고, 소프트웨어 개발과정을 측정하기 위해서도 사용된다. 소프트웨어 메트릭스는 일반적으로 크게 산출물 메트릭스, 리소스 메트릭스, 프로세스 메트릭스로 나누어진다[3, 9].

① 산출물 메트릭스

산출물 메트릭스는 최종 산출물과 중간 산출물의 특성을 평가하기 위한 척도이다. 평가되는 대표적인 척도로서 규모 및 복잡도등을 들 수 있다.

• 규모 메트릭스

규모 메트릭스는 프로스램 라인수(LOC : Lines Of Code)와 소프트웨어의 기능수를 측정하여 규모를 나타내는 기능점(Function Point) 방법[11]이 있다.

LOC는 대부분 개발자에게 있어서 가장 친숙하고 쉽게 산출할수 있는 메트릭스이다. 기능점 방법은 IBM의 A. J. Albrecht가 1970년 후반에 제안한 것으로 비지니스용 어플리케이션 소프트웨어에 입력수, 출력수, 조회수, 데이터 화일수, 인터페이스수의 가중치 합에 의해 기능적 규모를 나타낸다.

• 복잡도 메트릭스

소프트웨어의 작성과 이해의 어려움을 나타내는 특성이다. 복잡도 메트릭스는 대부분 프로그램 코드를 대상으로 하고 있다. 예를들어 McCabe의 순환수와 Halstead의 소프트웨어 과학 등을 들 수 있다.

② 리소스 메트릭스

리소스 메트릭스는 개발작업에 있어서의 자원의 소비량과 소비될 자원의 특성을 평가하기 위한 척도이다. 주요 평가대상은 시간(작업시간)과 개발자이다. 작업시간은 보통 달력시간을 단위로 평가한다.

③ 프로세스 메트릭스

프로세스 메트릭스는 각각의 개발작업과 개발작업 전체의 특성을 평가하기 위한 척도이다. 평가될 특성으로서 진행상황, 작업효율, 생산성 등을 들 수 있다. 각 변환작업의 프로세스 메트릭스는 그 작업의 입출력 산출물과 자원 특성간의 관계로서 정의되는 경우가 많다. 이것은 개발작업 자체의 정량적 평가가 어렵기 때문에, 작업으로 만들어 낸 산출물의 품질과 소비된 자원의 양에 바탕을 두고 간접적으로 평가한다.

2.3 IEEE Std-830-1998

IEEE Std-830-1998규격서[4]는 고품질의 소프트웨어 요구를 논의하는 적절한 출발점이다. 여기에서는 요구사양서

에 기술해야하는 내용과 기술하지 말아야할 내용에 대해 다룬다. 즉 요구사양서가 취급해야 할 사항들을 서술한다. IEEE Std-830-1998규격서는 5절(개요, 참고문헌, 용어의 정의, 좋은 요구사양서를 생산하기 위한 고려, 요구사양서에 포함되는 부품의 개요)과 부록 2절(요구사양서의 템플릿, IEEE/EIA 12207.1-1997라고 불리는 다른 표준 규격과의 관련)로 구성된다. 이 중에서 본 논문의 기준이 되는 「4.3 좋은 요구사양서가 가져야 할 특성」은 다음과 같은 내용을 담고 있다.

2.3.1 좋은 요구사양서가 가져야 할 특성

좋은 요구사양서가 가져야 할 8가지 특성으로는 다음과 같다[4, 6].

- 타당하다(Correct) : 요구사양서에 기술되어 있는 모든 요구항목을 구축대상의 시스템에 반영하는 것이다. 이것은 사용자의 요구에 일치하고 있는가 아닌가를 나타내는 성질로 이것을 자동적으로 확인하는 방법은 없다.
- 모호하지 않다(Unambiguous) : 요구사양 중에 기술되어 있는 모든 요구가 하나의 의미로 해석할 수 있는 경우 요구사양은 애매모호하지 않다고 말할 수 있다. 만약 어떤 요구가 몇 가지 의미로 해석될 수 있는 경우 그 사양은 애매모호하게 된다. 예를 들어 「크다, 작다」등의 표현이 아니고, 구체적인 수치에 의해 이 성질은 충족된다.
- 완전하다(Complete) : 기능, 성능, 외부 인터페이스, 속성, 설계제약에 관련하는 요구, 모든 상태에서의 입력데이터에 대해 소프트웨어가 어떻게 응답할지의 정의 등 이 같은 요건이 기술되어 있는 경우를 완전하다고 말한다.
- 모순이 없다(Consistent) : 요구사양서에 기술되어 있는 각각의 요구항목이 서로 모순되지 않을 때 모순이 없다고 한다.

중요성과 안정성에 순위를 붙인다(Ranked for importance and/or stability) : 할당된 스케줄과 예산의 범위 내에서 사용자는 어느 요구항목을 대단히 중요하게 생각하고 있는지를 분명히 하는 경우이다. 예를 들어 기술내용은 절대 충족해야 하는지, 조건부로 좋은지, 그렇지 않으면 옵션인지를 분명히 하고 있는 경우 그 요구사양서는 중요도와 안정성에 순위를 붙일수 있다.

- 검증가능하다(Verifiable) : 개발될 소프트웨어가 요구사양서의 기술 내용을 충족시키고 있는가를 체크하기 위한 방법이고, 체크 작업은 타당한 비용 내에서 타당한 시간에 끝날 경우 그 요구사양서는 검증가능하다. 예를 들어 「좋다, 크다, 작다」등의 애매모호한 기술을 포함할 경우 검증 가능하다고는 말할 수 없고 수치 등으로 지정된 기술은 검증하기 쉽다.
- 변경가능하다(Modifiable) : 요구사양의 구조, 스타일을 유지하면서 요구를 용이하게 그리고 완전하게 모순 없이 변경할 수 있는 경우를 의미한다.
- 추적가능하다(Traceable) : 앞 단계에서의 요구사항이 어떻게 실현되고 있는가를 확인하는 것이 가능한 것을 의미한다.

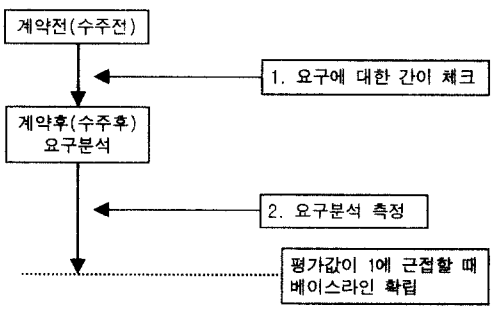
3. 평가방법의 제안

2.3절에서 요구사양서가 가져야 할 특성 8가지를 기술하였다. 그러나 여기에서 추적가능성, 완전성, 타당성은 요구사양서 자체만으로는 측정하기 어렵다. 따라서 본 연구에서는 이들 3항목을 제외하고 나머지 5항목에 대해서 평가방법을 제안한다.

3.1 개 요

요구에 관한 리스크를 식별하고, 조기에 대책을 강구할 수 있도록 다음과 같은 2가지의 활동을 제안한다(그림 2). 첫 번째 활동은 계약 전(수주 전) 사용자(고객)로부터 의뢰받은 요구에 대해, 프로젝트 관계자는 사용자의 요구가 요구사양서에 어느 정도 잘 정의되어 있는가를 간략히 평가하는 것이다. 이것은 계약 후(수주 후)에 소프트웨어 개발 계획서를 작성할 때 인력계획과 작업계획(투입노력의 산정과 스케줄)에 반영시킬 수 있다. 두 번째 활동은 수주 후 요구분석 단계에서 평가하는 활동이다. 소프트웨어 개발 하위공정에서는 프로젝트의 상태를 정량적으로 파악할 수 있지만, 상위공정에서는 요구의 상태를 제 3자가 파악하는 것은 어렵다.

따라서 상위공정의 가시성을 향상시키고 제 3자도 일정한 공통의 인식을 가질 수 있게 하여 상위공정의 문제점을 조기에 파악, 각각의 입장에서 대책을 강구할 수 있도록 한다.



(그림 2) 평가활동

3.1.1 요구 간이체크 단계

사용자의 요구가 가장 잘 정의되어 있지 않은 시점이기도 한 프로젝트의 초기 단계에서, 프로젝트 관계자가 요구사양에 대해 간단한 체크를 실시하여 서로 공통의 인식을 갖는다. 이 활동으로 요구와 관련하는 리스크를 식별하고 리스크 감시 체제를 갖춘다.

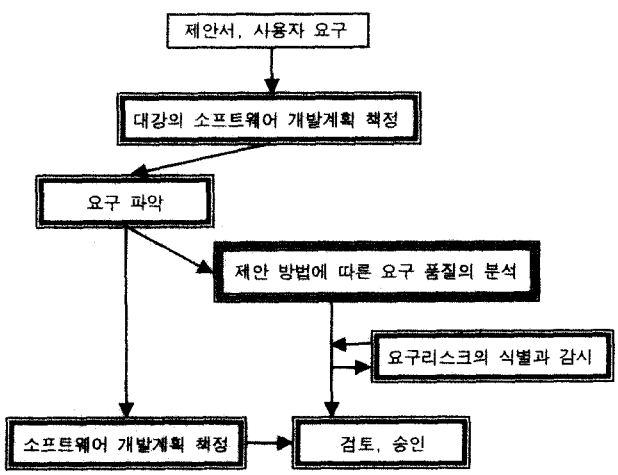
〈표 1〉 간이 체크리스트

질 문 사 항	비 율
요구안에 명확하게 미확정으로 되어 있는 것과 표현이 애매모호하고 하나의 의미로 해석할 수 없는 것은 전체 몇% 차지하고 있는가?	
명확하게 모순하고 있다고 생각되는 것은 몇% 인가?	
현시점의 요구 중에서 그 요구가 만족된다는 것을 검증하는 수단이 없는 것은 어느 정도의 비율(%)인가?	
중복하고 있거나, 요구간에 상호의존도 높은 것이 차지하는 비율(%)은 어느 정도 인가?	

<표 1>의 간이 체크리스트(checklist)는 거래 시 요구가 어느 정도 잘 정의되어 있는가를 진단하기 위한 것으로 고객의 요구나 RFP(Request For Proposal)를 대충 훑어본 담당자가 제안서를 작성할 때의 단계에서 이용한다. 이 단계에서는 직관적인 평가도 무방하다.

3.1.2 요구분석 측정 단계

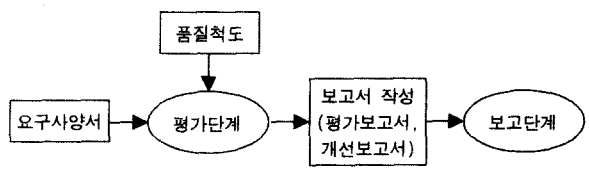
개발자가 사용자의 요구를 정량적으로 파악하고, 제 3자도 요구에 관한 리스크를 파악할 수 있도록 하기 위한 단계이다. 요구분석 측정 단계는(그림 3)과 같은 방법으로 실시된다. ①프로젝트의 요구 정의 작업에 관련하는 전원이 분석하며 3.4절에 기초를 두어 측정한다. ②각자의 결과를 가지고 서로 토의를 한다. ③관계자 전원이 통일된 견해를 체크결과에 반영하고 보고서를 작성한다.



(그림 3) 요구 정의 단계

3.2 평가 프로세스

요구사양서 평가프로세스는 다음과 같이 구성되어 있다(그림 4). 먼저 요구사양서중에 포함되어 있는 요구항목에 대해 5개의 품질척도인 비모호성, 무모순성, 중요성과 안정성, 검증가능성, 변경가능성을 측정하여 평가보고서와 개선보고서를 작성한다.



(그림 4) 평가프로세스

소프트웨어의 성패는 요구사양서와 깊은 연관이 있기 때문에 요구사양서의 품질은 소프트웨어의 품질에 영향을 준다. 따라서 요구사양서에서 발견되는 결함에 의해 품질이 결정된다. 이에 본 논문에서는 앞에서 기술한 품질척도에 충족되지 않을 경우 결함으로 간주한다. 요구사양서를 평가한 후 평가보고서<표 2>를 작성하고, 각 결함에 대해 개선보고서를 작성한다<표 3>. 이 보고서에 의해 사용자의 요구가 어느 정도 좋은지 그리고 어느 부분에 결함이 있는가를 파악할 수가 있다.

〈표 2〉 평가 보고서

평 가 보 고 서			
목적			
참가자		장소	
레뷰 담당자		공수	
프로젝트명		제출일	
품질특성		**평가치**	
비모호성			
무모순성			
중요도와 안정성			
검증가능성			
변경가능성			

〈표 3〉 개선 보고서

개 선 보 고 서			
작성자		참가자	
프로젝트명		날짜	
품질척도	**번호**	**문제점**	**개선지침**
비모호성			
무모순성			
중요도와 안정성			
검증가능성			
변경가능성			

3.3 평가방법

3.3.1 비모호성

요구사양서안에 포함되어 있는 다음과 같은 모호한 항목들을 카운트 한다. ①요구를 하나의 의미로 해석할 수 없는 것, ②정의가 없는 용어를 포함한 요구, ③수치에 따른 지정으로 표현해야 할 것을 「크다, 작다, 빠르다」등의 표현을 사용하고 있는 요구항목들이다. 비모호성의 메트릭스는 다음과 같다.

측정결과 = 1 - (모호한 요구항목의 수 / 총 요구항목수)

측정결과는 positive 메트릭스이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다. 다시 말해서 요구사양서가 명확하게 기술되어 있다는 것을 의미한다. 측정결과는 다음과 같은 범위를 가진다.

$$0 \leqq \text{측정결과} \leqq 1$$

3.3.2 무모순성

요구사양서에 기술되어 있는 개개의 요구항목이 서로에게 모순하고 있는 항목수를 카운트 한다. 비모순성의 메트릭스는 다음과 같다.

측정결과 = 1 - (모순되는 요구항목수 / 총 요구항목수)

측정결과는 positive 메트릭스 이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다. 다시 말해서 요구사양서가 일관되게 기술되어 있다는 것을 의미한다. 측정결과는 다음과 같은 범위를 가진다.

$$0 \leqq \text{측정결과} \leqq 1$$

3.3.3 중요도와 안정성 순위

요구사양서에 기술되어 있는 개개의 요구항목에 대해서 필수/옵션(Essential/Optional, 이하 EO), 혹은 우선순위가 지정되어 있는지를 조사하고, EO 또는 우선순위가 지정되어 있지 않은 항목의 수를 카운트 한다. 이것에 의해 요구항목의 중요도와 안정성을 용이하게 파악할 수가 있다. 메트릭스는 다음과 같다.

측정결과 = 1 - (요구에 EO 혹은 우선순위가 지정되지 않은 항목수 / 총 요구항목수)

측정결과는positive 메트릭스 이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다. 측정결과는 다음과 같다.

$$0 \leqq \text{측정결과} \leqq 1$$

3.3.4 검증가능성

요구사양서안에 포함되어 있는 다음과 같은 검증 불가능한 항목들을 카운트 한다. ① 테스트가 불가능한 항목, ② 타당성을 확인할 수 없는 항목, ③ 「크다, 작다, 자주」 등 정성적인 표현을 사용하고 있는 항목이다. 메트릭스는 다음과 같다.

측정결과 = 1 - (검증 불가능한 항목수 / 총 요구항목수)

측정결과는 positive 메트릭스 이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다. 다시 말해서 요구사양서에 기술되어 있는 요구항목이 검증 가능하다는 의미이다. 측정결과는 다음과 같은 범위를 가진다.

$$0 \leqq \text{측정결과} \leqq 1$$

3.3.5 변경가능성

요구사양서중에 의미 없이 장황하게 기술되어 있는 항목, 서로 의미적으로 의존하고 있는 항목을 카운트 한다. 메트릭스는 다음과 같다.

측정결과 = 1 - (장황한 요구, 의미가 의존하고 있는 항목 / 총 요구항목수)

측정결과는 positive 메트릭스 이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다. 다시 말해서 요구항목을 변경할 수 있음을 의미이다. 측정결과는 다음과 같은 범위를 가진다.

$$0 \leqq \text{측정결과} \leqq 1$$

3.3.6 요구사양서의 품질

소프트웨어의 성패는 요구사양서와 깊은 연관이 있기 때문에 요구사양서에서 발견되는 결함 즉 앞에서 기술한 5가지의 품질척도에서 발견된 전체 결함의 수를 요구사양서의 품질로 정의한다.

요구사양서의 품질 = 1 - (D / 총 요구항목수)
D : 5개의 품질척도의 총 결함수

측정결과는 positive 메트릭스 이다. 이것은 측정결과가 1에 가까울수록 좋은 품질을 나타낸다.

$$0 \leqq \text{측정결과} \leqq 1$$

4. 실험 및 분석

제안 방법의 유효성을 검증하기 위해 모 기업으로부터 의

뢰 받은 총 5개의 프로젝트를 준비하였다. 각 프로젝트는 P1, P2, P3, P4, P5로 표시 하였다. 먼저 3개의 프로젝트는 제안 된 방법을 사용하여 실행한 프로젝트이고, 나머지 2개의 프로젝트는 요구사양서 평가 없이 수행한 프로젝트이다 <표 4>. P1, P2, P3는 네트워크 프로덕션 소프트웨어 개발이었으며 이들 프로젝트는 영상처리와 네트워크 관련 소프트웨어 이다. P4프로젝트는 영상처리 관련 소프트웨어 개발이며 P5는 여러 개의 객체가 메시지를 상호 송/수신하며 처리하는 네트워크 관련 소프트웨어 개발 이었다. P4 프로젝트는 P1, P2, P3 프로젝트의 영상처리 부문과 유사성으로 인해 선택되었고, P5는 네트워크 관련 부분의 유사성으로 인해 선택 되었다.

〈표 4〉 평가 프로젝트

프로젝트	제안방법의 사용 유무
P1	사용함
P2	
P3	
P4	사용안함
P5	

4.1 측정결과

처음에 프로젝트 P1, P2, P3의 요구사양서를 측정한 결과 P1의 비모호성이 0.61, 무모순성 0.72, 중요도과 안정성 1, 검증가능성 0.67, 변경가능성 0.75였고, P2는 비모호성 0.74, 무모순성 0.84, 중요도과 안정성 1, 검증가능성 0.69, 변경가능성 0.76, P3는 비모호성이 0.78, 무모순성 0.81, 중요도과 안정성 1, 검증가능성 0.62, 변경가능성 0.72의 값이 산출 되었다<표 5>.

〈표 5〉 초기 요구사양서 평가결과

프로젝트	비모호성	무모순성	중요도와 안정성	검증가능성	변경가능성	품질
P1	0.61	0.72	1	0.67	0.75	
P2	0.74	0.84	1	0.69	0.76	
P3	0.78	0.81	1	0.62	0.72	
P4	NA	NA	NA	NA	NA	
P5	NA	NA	NA	NA	NA	

이들의 각 평가결과 및 문제점등을 평가 보고서와 개선 보고서에 작성하고 이후 그 보고서를 근거로 요구사양서의 문제점을 수정하였다. 그리고 다시 요구사양서에 대해 평가를 실시는 것을 총 3회 실시 한 결과 P1, P2, P3의 각 품질 특성의 요소값이 크게 향상되었다<표 6>.

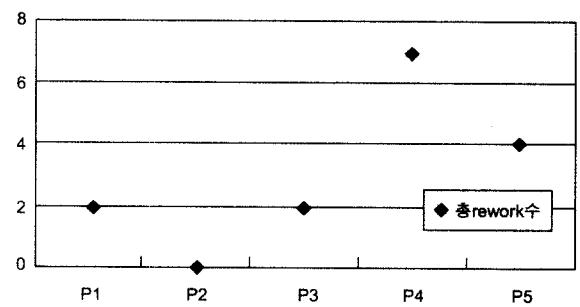
〈표 6〉 요구사양서 평가결과

프로젝트	비모호성	무모순성	중요도와 안정성	검증가능성	변경가능성	품질
P1	1	1	1	1	1	1
P2	1	1	1	1	1	1
P3	0.96	0.96	1	0.96	0.96	0.96
P4	NA	NA	NA	NA	NA	
P5	NA	NA	NA	NA	NA	

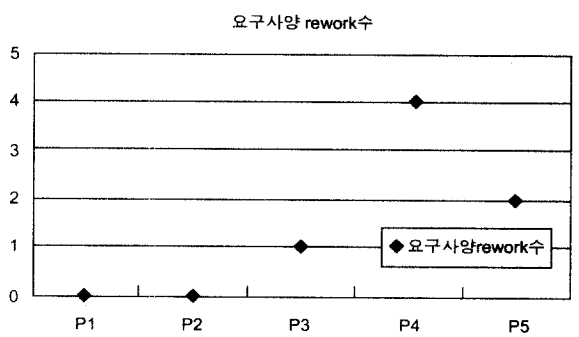
위 결과에서 프로젝트 P3의 중요도와 안정성을 제외한 각 품질특성이 0.96의 값이 산출된 이유는 분석과정 중 12개의 요구항목에 대해서 정확한 의미를 파악할 수가 없었고, 한편 이들 12개의 항목에 관해서 고객에게 문의해 보았지만 고객으로부터의 즉각적인 대답(대답을 회피하며 지연)이 없었다. 그래서 이들 12개의 요구항목을 결함으로 간주하여 평가하였다. 더불어 이 12개의 항목을 리스크로 지정한 후 개발을 진행하였다.

4.2 프로젝트 결과 분석

프로젝트 개발에 따른 재작업(rework)을 분석한 결과 요구사양서를 평가 실시한 후 수행한 프로젝트(P1, P2, P3)가 요구사양서를 평가하지 않고 수행한 프로젝트에 비해 재작업의 수가 적음을 (그림 5), (그림 6)의 그래프를 통하여 알 수 있다.



(그림 5) 총 재 작업의 수



(그림 6) 요구사양에 대한 재작업의 수

즉 P1, P2, P3 프로젝트는 총 재 작업이 평균 약 1.3(2+0+2)회가 발생한 반면 P4, P5 프로젝트는 평균5.5(7+4)회가 발생하였다. 또한 요구사양서에 대한 결함으로 인해 발생한 재작업은 P1이 0, P2가 0, P3이 1회 발생한 반면 P4, P5는 각 4, 2회가 일어났다.

재작업은 성과물의 결함으로 인해 발생할 수도 있지만 고객의 요구를 반영하고 있지 않거나 잘못된 요구의 이해로 인해 발생할 수도 있다. 만약 재작업이 자주 발생한다면 이것은 생산성의 저하 및 비용, 납기(delivery)의 지연 문제로 이어지고 결국 이러한 프로젝트는 실패할 가능성이 높아지게 된다. (그림 6)의 결과에서 보듯이 프로젝트 P4, P5는 요구사양의 결함으로 인해 재작업이 자주 발생 하였지만, P1, P2, P3의 프로젝트에 있어서는 요구사양에 대한 결함의 수는 거의 없었다. 이러한 결과로 미루어 본 방법이 어느 정도의 유효성이 있음을 알 수 있다. 실제 P4, P5 프로젝트는 잦은 재작업 및 변경으로 인해 계획이 마이나스[1]된 실패한 프로젝트로 분류 되었다.

5. 결론 및 연구과제

지금까지 제안된 수많은 메트릭스의 대부분이 소스 코드를 대상으로 한 메트릭스로, 소프트웨어 요구사양서 자체를 대상으로 한 메트릭스는 거의 존재하고 있지 않다. 따라서 본 논문에서는 요구사양서 자체에 대해 품질을 평가할 수 있는 방법을 제안하였다. 그리고 유효성을 검증하기 위해 제안 방법을 적용한 프로젝트 3개와 제안 방법을 사용하지 않은 프로젝트 2개에 대해 비교 분석을 통해 제안된 방법이 실제로 어느 정도 효과가 있음을 확인 하였다.

본 논문에 제안된 방법으로 다음과 같은 사항들을 기대할 수 있다. 첫째, 소프트웨어 개발 하위공정에서는 프로젝트의 상태를 정량적으로 파악할 수 있지만, 상위공정에서는 요구의 상태를 제 3자가 파악하는 것은 어렵다. 따라서 상위공정의 가시성을 향상시키고 제 3자도 일정한 공통의 인식을 가질 수 있도록 하여 상위공정의 문제점을 조기에 파악, 각각의 입장에서 대책을 강구할 수 있도록 한다. 둘째, 관리층에게 요구사양에 관해 보다 객관적이고 정량적인 정보를 제공할 수 있다. 셋째, 요구사양의 결함으로 인한 재작업의 수를 감소시켜줌으로서 생산성의 향상 및 불필요한 공수의 감소에 도움을 준다.

향후 연구 과제로서 제안방법의 각 결함 항목을 좀더 세부적으로 조사하여 추가하고, 보다 많은 소프트웨어 개발 프로젝트에 대해서 검증을 실시해야 할 것이다.

1) 계획 마이나스 프로젝트란 적자는 아니지만 예측비용을 초과한 프로젝트를 말한다

참 고 문 헌

[1] P. Oman and S. L. Pfleeger, "Applying Software Metrics," IEEE Computer Society Press, 1997.

[2] 海谷治彦, 佐伯元司, 大西淳, "ウィンターワークショップ・金澤報告要求工學", 情報處理學會研究報告, pp.87-89, 2001.

[3] 井上克郎, 松本健一, 飯田元, "ソフトウェアプロセス", 共立出版, 2000.

[4] IEEE Recommended Practics for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society, 25, June, 1998, ISBN0-7381-0332-2, "http://www.stanford.edu/class/cs194/handouts/ieee830-1998.pdf."

[5] Karl E. Wiegers, 渡部洋子(譯), "ソフトウェア要求-顧客が望むシステムとは", 日経BPソフトプレス, 2003.

[6] Dean Leffingwell, Dong Widrig, 日本ラショナルソフトウェア株式會社(譯), "ソフトウェア要求管理-新世代の統一アプローチ", ピアソン・エデュケーション, 2002.

[7] 일본 요구공학 사이트, http://www.selab.is.ritsumei.ac.jp/~ohnishi/RE/rewg.html.

[8] Barry Boehm, Victor R. Basili, "Software Defect Reduction Top 10 List," http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.78.pdf.

[9] K. Hatano, Y. Nomura, H. Taniguchi and K. Ushijima, "A Method TO Support Refactoring Using C&K Metrics," Proceedings of PYIWIT, pp.112-118, Mar., 2002.

[10] Capers Jones, 富野壽(譯), "ソフトウェア見積りのすべて(規模, 品質, 工數, 工期の予測法)," 共立出版, 2001.

[11] A. J. Albrecht, "Measuring application development productivity," Proc of Joint SHARE/GUIDE/IBM Application Development Symposium, pp.83-92, 1979.

김 경 환

e-mail : k6082@yahoo.co.kr

1999년 단국대학교 전자계산학과

2002년 일본 오사카대학 대학원 정보공학과 (공학석사)

2002년~현재 일본 Software Research Associates

2003년~현재 일본 프로세스 개선 회원

관심분야 : 소프트웨어공학(프로세스 개선, 소프트웨어 메트릭스, 리스크 관리)

코지마 츠토무

e-mail : t-kozima@sra.co.jp

1986년 일본 Software Research Associates 입사

1993년~1997년 역 공학(Reverse Engineering)개발 연구

1997년~현재 Software Process Improvement 연구, 컨설트활동

2000년~현재 일본 프로세스 개선 연구원

관심분야 : 소프트웨어 프로세스 개선, 리스크 관리

박 용 범

e-mail : ybpark@cs.dankook.ac.kr

1985년 서강대학교 전자계산학과 졸업 (학사)

1987년 N.Y. Polytechnic Univ. 대학원 전자계산학과(석사)

1991년 N.Y. Polytechnic Univ. 대학원 전자계산학과(박사)

1993년~현재 단국대학교 전자계산학과 부교수

관심분야 : Information Architecture, 패턴인식, 분산에이전트