소프트웨어 개발 과정에서 제품의 품질 척도를 적용하는 방법

(Establishment of the Software Quality Metrics for a Software Development Process)

이 선 아 † 최 병 주 ††
(Seon-ah Lee) (Byoungju Choi)

요 약 고품질의 소프트웨어를 개발하기 위하여, 소프트웨어의 품질을 측정하는 메트릭스가 활발히 개발되었고 근래에는 객체지향 메트릭스도 제시되고 있다. 그러나 개발 과정에서 품질을 관리하기 위하여 메트릭스를 이용하는 방법에 대한 연구는 제대로 이루어지고 있지 않다. 이는 개발 과정에서 적용되는 메트릭스의 의미를 전체적인 품질 관점에서 이해하기가 어렵기 때문이다.

본 논문에서는 제품 특성을 기반으로 정의된 소프트웨어 품질 모형(H-SQM)을 이용하여 개발 과정에 소프트웨어 품질 메트릭스를 적용하는 방법을 제안한다. 이 방법에서는 H-SQM을 표현한 원인-결과 다이어그램을 공정분석도로 바꾼다. 공정분석도에 따라 개발 단계별로 소프트웨어 품질 메트릭스를 적용한다. 이러한 방법으로 개발 과정에서 소프트웨어의 품질을 효율적으로 개선해 나갈 수 있도록 메트릭스를 적용할 수 있게 된다.

Abstract In order to develop high quality software, software metrics have been made to assess the quality of software, and recently, many object-oriented metrics have been suggested for this purpose as well. However, research on the utilization of metrics to control software quality in a development process has been inadequate. This is due to the difficulty in assessing the significance of metrics in a software development process from the perspective of overall software quality.

In this paper, we propose a method of applying metrics to a development process using the Hierarchical Software Quality Model(H-SQM) which is defined in terms of the products' special features. The method represents the H-SQM as the cause-and-effect diagram and changes the diagram to the process-analysis diagram. And it applies software quality metrics to each development stage by the process-analysis diagram. In this way, we could utilize the software quality metrics efficiently in order to improve the quality of software in the software development process.

1. 서 론

고품질의 소프트웨어를 개발하기 위하여 여러 가지의 품질 관리 방법이 제시되어 왔다. 이 방법은 크게 프로세스의 품질을 관리하는 방법과 제품의 품질을 관리하는 방법으로 나눌 수 있다. CMM, SPICE는 개발 프로세스의 품질을 측정한다. 이 방법들은 '개발 기간동안 직접적으로 측정할 수 있는 프로세스의 품질을 향상시킴으로써 제품의 품질 향상을 가져올 수 있다'는 가정에 기반을 둔다. 프로세스의 품질 평가 및 개선은 산출물의 일관성을 보증하나 제품 자체와는 독립적으로 이루어진다. 따라서 요구명세가 잘 기술되어 있을 경우 개선된 프로세스를 통해 제품의 품질을 향상시킬 수 있으나, 개발 프로세스에서 제품 품질을 직접적으로 감사(監査)할 수 없다. ISO/IEC 9126[3], ISO/IEC 14598[4] 및 생명주기 단계별 품질 측정표는 소프트웨어 제품의 품질을 측정한다. 이 방법들은 소프트웨어 제품에 대한 품질을

† 비 회 원 : 이화여자대학교 컴퓨터학과
duri@cs.ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bjchoi@mm.ewha.ac.kr
논문접수 : 1999년 6월 10일
심사완료 : 2000년 1월 5일

정의하고, 품질을 품질 속성, 하위 품질 속성으로 세분화하여 메트릭스를 정의한다. 이러한 방법에서 정의하는 품질 속성(quality characteristics)[3]은 제품의 외부적인 행위를 평가하는 관점에서 정의되기 때문에 개발 프로세스에서 직접적으로 품질 속성을 평가할 수 없다. 또한 여러 품질 속성에 동일한 메트릭스가 중복되어 정의되므로 메트릭스의 의미를 명확하게 분석하지 못한다. 따라서 이러한 방법들은 '해당 품질 점검 활동으로 품질 개선이 실질적으로 이루어지는가?', '해당 품질 점검 활동이 품질 개선을 위해 효율적인가?' 등의 의문에 답을 제공하지 못한다.

본 논문에서는 개발자 관점에서의 소프트웨어 품질 모형(H-SQM: Hierarchical Software Quality Model)[6]을 이용하여 개발 과정에 소프트웨어 품질 메트릭스를 적용하는 방법을 제시한다. 이 방법은 첫째, 품질 속성간의 선후 조건 관계를 설정하여 품질 속성간 중복되는 제품 특성을 한 품질 속성에 속하도록 하는 기반을 마련한다. 둘째, 품질 속성간의 관계를 기반으로 소프트웨어의 품질에 영향을 미치는 제품 특성을 규명한다. 셋째, 제품 특성을 개발 단계별 구성 요소(Component)의 특성으로 세분화하고 이를 바탕으로 단계별 메트릭스를 설정한다. 이렇게 함으로써 품질에 영향을 미치는 요인을 측정하는 메트릭스가 무엇인지와 메트릭스를 측정한 결과가 품질의 관점에서 어떠한 의미를 가지는지에 대한 분석이 용이해 진다. 본 논문은 개발 과정에서 소프트웨어 품질의 가시성(Visibility)를 확보하고 제품의 품질과 관련되는 단계별 메트릭스를 설정함으로써, 개발 프로세스에서 제품의 품질 개선에 효율적인 메트릭스를 적용하는 방법을 제안하였다는데 의의가 있다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구에 관하여 기술한다. 3절에서는 개발 프로세스에 소프트웨어 품질 메트릭스를 적용하는 방법을 제시한다. 4절에서는 객체지향 개발 프로세스인 마르미-II[12]에 단계별로 메트릭스를 설정한 사례를 보여준다. 5절에서는 본 논문에서 제시한 메트릭스 적용 방법을 분석한다. 6절에서는 결론과 향후 연구과제를 제시한다.

2. 관련 연구

소프트웨어의 품질을 정의하는 소프트웨어 품질 모형에 대한 연구가 1976년과 1977년 Boehm과 McCall에 의해 각각 시작되었고, 그 이후에도 다양한 품질 모형들이 제시되었다. 이러한 기존의 소프트웨어 품질 모형들은 '제품 특성을 측정하고 제어함으로써 제품의 품질을 향상시킬 수 있다'고 가정한다. 그러나 기존의 모형에서 이러한 가정은 이론적인 근거가 뒷받침되어 있지 않으므로 소프트웨어의 품질과 메트릭스와의 관계는 모호하다. 따라서 소프트웨어의 품질 속성이 만족되기 위해서 어떤 메트릭스 집합이 만족되어야 하는지, 반대로 메트릭스가 만족되었을 경우 품질의 어떤 측면이 개선되는지 등을 알 수 없다. 기존의 소프트웨어 품질 모형[1]은 소프트웨어의 품질 관리에 실제로 사용하기에는 부족한 점이 있다. Dromey[5]는 개발 단계별로 소프트웨어의 구성 요소를 식별하고, 이러한 구성 요소의 특성으로 단계별 품질 모형을 구성하였다. 이 방법은 제품 특성에 기반을 두고 소프트웨어의 품질을 제어할 수 있는 가능성을 보여주지만, 이 방법으로 각각의 품질 속성이 향상되는지, 향상된다면 어떻게 향상되는지를 보여주지 못한다. ISO/IEC 9126의 소프트웨어 품질 모형은 국제 표준으로서 소프트웨어의 품질 정의에 대한 보편적인 모형이다. 이 모형은 품질 속성을 세분화된 품질 속성으로 나누어 명시하므로 제품이 완성된 후의 품질 측정에 적합하다. 그러나 개발 프로세스에서는 직접적인 품질 속성, 즉 제품의 외부적인 행위를 평가할 수 없으므로 프로세스에서의 품질을 검증하는 기준 모형으로 사용되기는 불충분하다. 따라서 이제까지의 소프트웨어의 품질 모형으로는 소프트웨어의 품질 관점에서 단계별 메트릭스를 결정하기가 어렵다.

다른 한편에서는 소프트웨어의 품질을 측정하는 객관적 척도로서 메트릭스가 연구되어 왔다. 또한 객체지향 개발 프로세스에 대한 연구가 활발히 진행됨에 따라 객체지향 소프트웨어의 품질을 평가하기 위한 객체지향 메트릭스와 객체지향 메트릭스를 기반으로 한 품질 평가 방법도 제시되고 있다. 개별적으로 이루어지는 메트릭스의 연구는 제품 특성을 정량적으로 측정하는 방법을 발전시키고 있으나, 이렇게 측정되는 메트릭스가 소프트웨어의 품질에 있어서 가지는 의미를 제대로 보여주지 못하고 있다.

논문 [6]은 위의 두 가지 연구의 딜레마가 소프트웨어의 품질 속성과 소프트웨어의 메트릭스가 다 대 다 관계로 연관되기 때문이라고 지적하고, 이 문제를 해결하기 위해 소프트웨어의 품질 속성의 정의와 범위를 고려하여 품질 속성간의 선후 조건 관계를 정의하였다. 이를 기반으로 H-SQM(Hierarchical software Quality Model)을 구성하여 소프트웨어의 특성이 품질에 끼치는 영향을 잘 보여주는 소프트웨어 품질 모형을 제시하였다. 따라서 H-SQM은 소프트웨어 품질 메트릭스를 측정할 때의 의미를 소프트웨어의 품질 관점에서 해석할 수 있게 만든다.

본 논문은 이러한 H-SQM이 개발 프로세스에서 소프트웨어의 품질을 제어하기 위한 기본 모형으로 적합하다고 보고, 이 모형을 이용하여 개발 과정에서 실질적인 품질 제어를 위한 방법을 제시한다. 또한 객체지향 개발 프로세스인 마르미-II[12]에 이 방법을 이용하여 단계별로 메트릭스를 설정하기 위하여, H-SQM에 객체지향 소프트웨어의 품질과 관련한 메트릭스를 추가하고 해당 메트릭스 집합을 개발 프로세스에 적용한다. 그럼으로 객체지향 소프트웨어의 품질 메트릭스가 실질적인 개발 프로세스의 품질 측정에 이용될 수 있도록 한다.

3. 개발 프로세스에서의 메트릭스 적용 방법

본 논문에서는 개발 프로세스에서 메트릭스를 적용하는 방법을 그림 1과 같이 제안한다. 그림 1은 품질과 제품, 프로세스를 축으로 하여, 소프트웨어 품질 메트릭스를 적용하는 과정을 품질에서 제품으로, 제품에서 프로세스로의 흐름으로 표현한다. 단계 1은 품질 속성을 제품 특성으로 재해석하는 단계이다. 단계 2와 단계 3은 소프트웨어 품질 모형을 개발 과정에 적용할 수 있도록 하기 위해 제품 특성을 구성 요소(component)의 특성으로 세분화하여 메트릭스를 설정하였다. 단계 3'에서는 일반적인 소프트웨어 특성을 객체지향 특성으로 변경하고 기존 메트릭스를 더 적절한 객체지향 메트릭스로 대체한다. 이 후, 단계 4와 단계 5에서 개발 과정에서 제품의 구성 요소가 산출되는 시점에 소프트웨어 품질 메트릭스를 적용하게 된다.

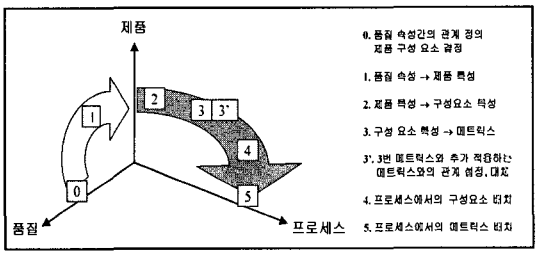

그림 1 개발 프로세스에서 메트릭스를 적용하는 방법

본 논문에서는 H-SQM의 메트릭스를 제품 구성 요소의 산출 시점에서 적용할 것을 제안한다. 이렇게 개발 프로세스 흐름에 따라 산출되는 제품 구성 요소의 품질 관련 특성을 검증함으로써 어느 개발 프로세스에나 적용 가능하게 된다. 본 논문에서는 이를 구체적으로 보여주기 위하여 요즘 활발히 연구중인 객체지향 개발 프로세스에 품질 향상을 위한 메트릭스를 적용해 보기로 한다.

4. 객체지향 개발 프로세스에서의 소프트웨어 품질 메트릭스 적용

본 논문에서는 적용할 객체지향 프로세스로 마르미II를 채택하였다. 이 프로세스는 계획, 정립, 점진적 개발, 인도의 4단계로 이루어져 있다.

4.1 사전 작업 (단계 0)

단계 0에서는 품질을 제품의 관점에서 해석하기 위하여 품질 속성간의 관계를 정의하고, 제품 특성을 구성요소의 특성으로 해석하기 위하여 제품의 구성 요소를 결정한다.

4.1.1 품질 속성간의 관계 정의

제품 특성이 품질에 끼치는 영향에 대한 검증이 용이하도록 품질 속성간의 선후 조건 관계를 제시한다. 이 관계는 이후의 단계에서 품질과 관련되는 제품 특성을 정리하는 기준으로 사용된다. 이로써 제품 특성을 기반으로 하면서도 하위 품질 요소가 중복되지 않는 계층적인 소프트웨어 품질 모형을 만들 수 있게 된다. 논문 [6]에서는 품질 속성을 7가지로 정의하고 품질 속성간의 선후 조건 관계를 그림 2와 같이 정의하였다.

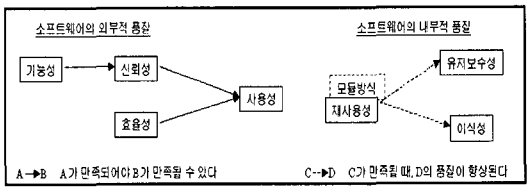

그림 2 품질 속성간의 선후조건 관계

4.1.2 소프트웨어 제품의 구성요소 결정

제품 특성을 개발 단계마다 검증할 수 있는 구성요소(Component)의 특성으로 정의하기 위해 제품의 구성요소가 무엇인지를 정의한다. 이 때 제품 특성은 '제품 자체에서 직접, 간접으로 측정할 수 있으며 시스템의 행위에 영향을 미치는 요인'으로 정의될 수 있으며 이미 완성된 제품의 속성을 말한다. 구성 요소의 특성은 '제품의 구현을 위해 단계별로 필요한 구성 요소들의 특성'으로 정의될 수 있다.

소프트웨어의 제품은 '사용자에게 인도할 것으로 지정된 컴퓨터 프로그램, 절차와 관련문서 그리고 데이타의 전체 집합'[3]이라고 정의된다. 제품의 정의에 따라 제품의 구현을 위해 필요한 제품의 구성 요소를 프로그램, 데이타, 문서 세 가지로 분류한다. 이 분류에 따른 소프트웨어 제품의 구성 요소의 예를 표 1에서 보여주고 있다. 이 구성요소는 개발 프로세스에서의 메트릭스의 측

정 대상이 된다. 하지만 이러한 직접적으로 제품과 연관된 산출물만이 제품의 구성요소는 아니다. 단계별로 필요한 구성 요소는 모두 포함할 수 있다. 예를 들어 요구분석 단계에서는 요구사항, 제약 조건 등이 될 수 있으며, 설계 단계에서는 모듈, 변수, 사전·사후 조건 등이 될 수 있다.

표 1 소프트웨어 주요 구성 요소

프로그램	데이타	문서
시스템, UI, 모듈(클래스), 코드 등	데이타 타입, 데이타 포맷, 파일 등	외부문서, 설계서, 명세서 등

4.2 품질 속성을 제품 특성으로 정의하는 단계 (단계 1, 2, 3)

단계 1, 2, 3에서 개발 관점의 소프트웨어 품질 모형 H-SQM를 구축한다. H-SQM은 품질 속성간의 선후 조건 관계를 고려하여 품질 속성별 제품 특성이 중복되지 않도록 만들어진 소프트웨어 품질 모형이다. H-SQM은 품질 - 품질 속성- 제품 특성 - 메트릭스의 계층으로 이루어져 있다. H-SQM을 그림 3에서 원인-효과 다이어그램(cause-and-effect)으로 나타내었다. 이는 H-SQM이 품질에 영향을 끼치는 요인에 따른 기준으로 메트릭스를 설정하였음을 보여준다. 소프트웨어 품질 모형을 원인-효과 다이어그램으로 나타냄으로써 품

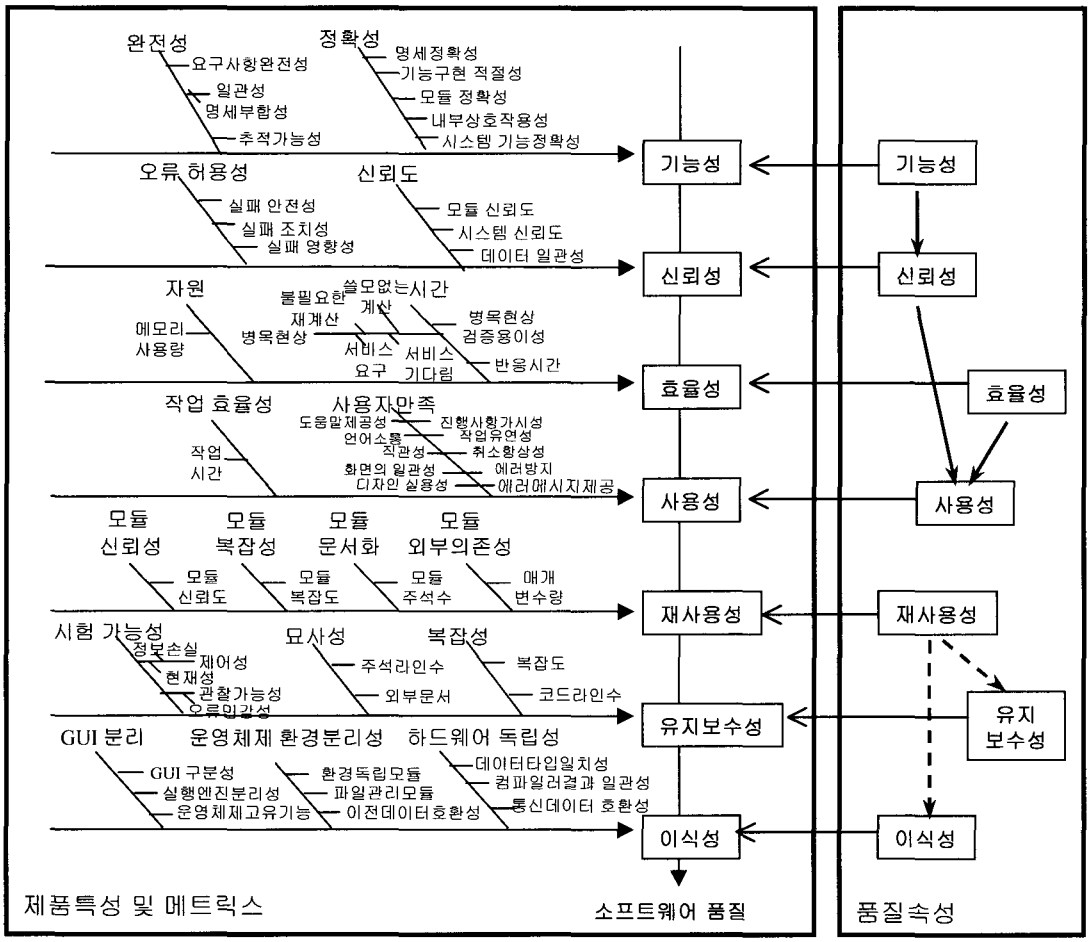

그림 3 H-SQM(소프트웨어 품질의 원인-효과 다이어그램)

질 속성과 제품 특성간의 관계를 명확히 보여주어 품질 관리에서의 소프트웨어 품질 모형의 역할을 뚜렷이 드러낼 수 있다.

4.3 일반 소프트웨어 메트릭스를 객체지향 메트릭스로 대체하는 단계 (단계 3')

단계 3'에서는 H-SQM의 메트릭스를 객체지향 메트릭스로 대체한다. 객체 지향 개발에서는 기존의 모듈 단위가 메소드, 클래스 및 컴포넌트 단위로 세분화된다. 따라서 모듈 관련 메트릭스를 객체지향 특성을 측정하는 메트릭스로 수정할 필요가 있다.

Berard[8]는 객체지향 특성으로 지역화(localization), 은닉화(encapsulation), 정보 은폐(information hiding), 상속성(inheritance), 추상화(abstraction)를 지적하였다. 이와 같은 객체 지향 특성을 기반으로 현재 제시되어 있는 객체지향 메트릭스를 정리하면 다음과 같다. Chidamber와 Kemerer[9,10]가 제안한 6가지 메트릭스 중 CBO와 RFC는 클래스간의 상호작용에 관한 메트릭스이므로 지역화 특성과 관계가 있다. LCOM은 한 클래스의 응집도를 평가하는 메트릭스로 LCOM의 값이 높을수록 은닉화 정도가 높다. DIT와 NOC는 클래스 상속 계층의 깊이와 너비를 평가하는 메트릭스이다. WMC는 클래스의 크기 및 복잡도 정도를 측정한다. Lorenz와 Kidd[8]의 메트릭스 중 NOO와 NOA는 하위 클래스에 포함된 메소드의 특성을 나타낸다. SI는 NOO 메트릭스에 레벨과 메소드 전체 수를 고려해 각 하위클래스의 특수화 정도를 나타낸다. NPavg는 메소드 매개변수의 수를 측정하며, 메소드 매개변수의 수가 많을수록 객체간의 연관성은 더욱 복잡해진다. Binder[11]가 제시한 메트릭스 중 PAP와 PAD는 public 속성과 관련된 메트릭스로 정보 은닉과 관련이 있다. 지역화, 은닉화, 정보 은폐는 모듈화를 도와주고, 상속성은 재사용성을 높여준다. 추상화는 묘사성을 높여주고 복잡성을 낮추어 유지보수를 용이하게 한다.

이러한 객체지향 메트릭스로 H-SQM의 메트릭스를 그림 4와 같이 대체한다. 모듈 복잡성에 대한 메트릭스로 H-SQM에서는 McCabe의 복잡성 메트릭스를 사용하고 있으나, 객체지향에서는 WMC, RFC, LCOM으로 한 클래스 내 복잡성을 평가할 수 있다. 이 때, WMC가 낮게, RFC도 낮게 그리고 응집성이 높게 평가될 때, 모듈 복잡성은 낮게 측정된다. 현재 적은 입출력 매개변수일 경우 모듈 의존성이 높게 평가되도록 되어 있지만, 클래스가 단위가 되면, CBO(다른 클래스와의 결합도)가 외부의존성으로 사용된다. PAP와 PAD는 낮게 유지될수록 시험가능성은 높아진다. SI는 상위 클래스에 의해 암시된 의미가 어느 정도 적절히 이용되고 있는지를 측정하므로, 묘사성과 관련된다. NPavg는 객체 내 메소드 간의 협력의 복잡도로 NPavg가 낮을수록 유지보수성은 높아진다. DIT와 NOC는 클래스에 속하는 메소드가 현재의 상태에서 몇 개의 다른 클래스에서 재사용이 되었는지를 평가하는 것이지, 재사용 될 수 있는 가능성을 평가하는 척도는 아니다. 따라서 H-SQM의 목적에 부합하지 않으므로 제외한다.

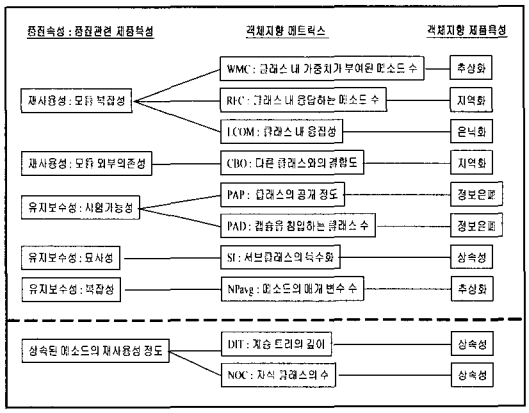

그림 4 품질 관련 제품 특성과 객체지향 제품 특성의 관계

4.4 개발 프로세스에 제품 특성 관련 메트릭스를 적용하는 단계 (단계 4, 5)

단계 4, 5는 개발 프로세스에서 소프트웨어 품질 모형을 적용하는 방법으로 품질 관리 기법에서의 공정 분석도를 이용한다. 3.1절의 H-SQM은 개발 프로세스의 흐름에 따라 재구성하여 공정 분석도로 나타낼 수 있다. 이를 위해 프로세스에서 구성요소를 만들어 내는 시점을 찾아내어 구성요소를 배치하고 그에 따른 메트릭스를 배치하면 그림 5와 같은 공정분석도가 된다.

마르미-II 개발프로세스에서는 요구사항 분석 활동과 점진적 개발 단계의 아키텍처 정제 구현이라는 설계 활동 및 클래스 구현, 컴포넌트 구현, 시스템 구현, 인도 단계의 사용자 테스트 활동에서 메트릭스가 적용된다.

그림 5에서 개발 흐름에 따라 품질 속성별로 제시된 구성 요소의 특성을 개발 단계별로 묶어 각 단계의 메트릭스로 적용한다(표2 참조). 품질 속성마다 측정하려는 측면이 다르기 때문에 각 단계에서 행하는 작업에 따라 강조되는 품질 속성도 달라지게 된다. 분석과 설계 단계에서는 명세에 따른 정확성, 완전성이 중요하기 때문에 기능성이 주로 측정된다. 물론 설계서를 바탕으로

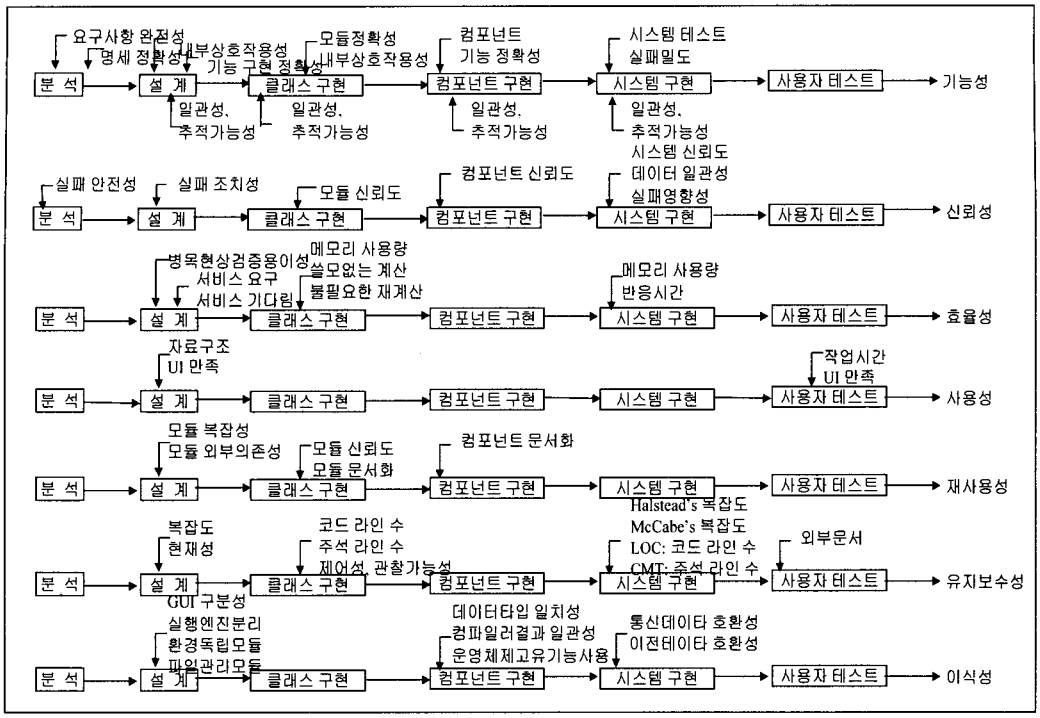

그림 5 H-SQM의 공정분석도 (품질 속성별 프로세스 분석)

각 품질 속성별로 검토가 이루어질 수 있다. 구현에서는 실제 구현된 모듈과 코드가 산출되어 재사용성, 유지보수성, 이식성을 측정한다. 테스트 단계에서는 외부 사용자 관점인 기능성, 신뢰성, 효율성, 사용성을 주로 측정한다.

객체지향 개발 프로세스에 메트릭스를 적용한 결과인 표 2[1]를 보면, 소프트웨어 구성 요소에 클래스가 추가되고, 클래스가 다시 단위 메소드, 상호 메소드, 상호 클래스로 세분화되었다. 객체지향 개발 프로세스에 대한 메트릭스 적용은 재사용성, 유지보수성 측면에서 주로 이루어졌다. 이는 클래스의 특성이 재사용성, 유지보수성의 품질 속성에 영향을 미치기 때문이다(그림 4 참조)

5. 분 석

본 논문에서 제시하는 메트릭스 적용 방법을 서론에서 제시한 두 가지 관점에서 분석하였다. 첫째, Ejiougu[2]가 제시한 여섯 가지 메트릭스의 요건을 응용하여 메트릭스 적용 방법에 대한 비교 기준을 제시하고 ISO/IEC 14598[4]과 비교 분석하였다. 이 분석은 품질을 점검하는 활동을 효율적으로 할 수 있는지에 대한 관점에서 이루어졌다. 둘째, 본 논문에서 제시한 방법이 개발 과정에서 소프트웨어의 품질에 대한 가시성을 확보하였는지를 점검하였다. 이 분석은 해당 방법으로 품질 개선이 실질적으로 이루어지는가에 대한 관점에서 이루어졌다.

5.1 메트릭스 요건 분석

본 논문에서 제시하는 메트릭스 적용 방법을 ISO/IEC 14598[4]과 비교 분석하였다. 이를 위하여 Ejiogu가 정의한 효과적인 소프트웨어 메트릭스가 포함해야 하는 6가지 요건[2]인 품질과의 연관성, 품질 향상 제공, 효율적인 활동, 의미 일관성, 객관성으로 비교 기준을 삼는다.

메트릭스의 적용 방법에 대한 기준으로 메트릭스의 요건을 사용하는 이유는 메트릭스의 적용 방법은 메트릭스를 이용하여 품질을 측정하는 전체적인 관점이기 때문이다. 따라서 메트릭스가 갖추어야 하는 요건을 전체적인 관점으로 확장시키면 바로 메트릭스의 적용 방법에 대한 요건이 된다. 본 논문에서는 다음의 6가지 메

1) 메트릭스의 정의는 논문[6] 참조.

표 2 객체지향 개발 프로세스에 적용된 메트릭스

프로세스	레벨	해당 메트릭스	품질 속성
분석검증	분석서	요구사항 완전성	기능성
		명세 정확성	
		실패 안정성	신뢰성
설계검증	설계서	기능 구현 범위	기능성
		기능 구현 정확성	
		추적가능성, 일관성, 기능명세부합성	
		실패조치성	신뢰성
		병목현상검증용이성	효율성
		서비스 요구	
		서비스 기다림	
		사용만족 검토	사용성
		모듈복잡성	재사용성
		모듈외부의존성	
		복잡도	유지보수성
		현재성	
		GUI 구분	이식성
		실행엔진 분리	
		환경독립모듈	
		파일관리모듈	
클래스 테스트	단위메소드	단위 테스트에서의 오류수	기능성
		쓸모없는 계산	효율성
		불필요한 재계산	
		데이타 타입의 시스템 독립성	이식성
	상호메소드	상호메소드 테스트에서의 오류수	기능성
		WMC : 한 클래스에 속한 메소드 수. RFC : 클래스 내 응답되는 메소드 수 LCOM : 메소드 사이의 응집도	재사용성
		NIPavg : 메소드 사이의 복잡성	유지보수성
	상호클래스	상호클래스 테스트에서의 오류수	기능성
		CBO: 객체 클래스들 사이의 커플링	재사용성
		PAP : 클래스의 공개 정도 PAD : 클래스 은닉화의 침해정도 SI : 상위 클래스의 의미 적절성	유지보수성

프로세스	레벨		해당 메트릭스	품질 속성
통합테스트	컴포넌트		컴포넌트 테스트에서의 오류수	기능성
			컴포넌트의 신뢰성	신뢰성
			컴포넌트의 문서화	재사용성
			컴파일러 결과 일관성	이식성
			컴포넌트의 환경독립성	
			운영체제 고유 기능 사용	
	미니프로젝트		미니프로젝트 테스트에서의 오류수	기능성
시스템테스트	시스템	코드	Halstead's 복잡도	유지보수성
			McCabe's 복잡도	
			LOC : 코드 라인 수	
			CMT : 주석 라인 수	
		실행	시스템 테스트 실패 밀도	기능성
			시스템의 MTTF	신뢰성
			데이타 일관성	
			실패 영향성	
			반응시간	효율성
			메모리 사용량	
			통신 데이타 호환성	이식성
			이전 데이타 호환성	
사용자테스트	시스템	실행	진행사항 가시성	사용성
			작업 유연성	
			취소 항상성	
			에러 방지	
			메시지 제공성	
			대답 제공성	
			사람-컴퓨터 언어소통	
			직관성	
			화면구성의 일관성	
			디자인의 실용성	
			작업효율: 유효 작업 시간	
			통신 데이타 호환성	이식성
			이전 데이타 호환성	
기타평가	외부문서		외부문서분량	유지보수성

트릭스 속성을 사용하여 두 방법에서 사용하는 소프트웨어 품질 모형 및 메트릭스를 비교하였다.

① 품질과의 연관성(empirically and intuitively persuasive) : '품질과의 연관성'은 메트릭스와 품질의 관계가 명확해야 한다는 원칙이다. 본 논문과 ISO/IEC 14598은 메트릭스와 품질과의 관계를 나타내기 위해 각각의 소프트웨어 품질 모형을 이용한다. ISO/IEC 14598은 ISO/IEC 9126의 품질 모형을 이용한다. ISO/IEC 9126의 품질 모형은 사용자 관점에서 정의된 품질 속성의 세분화로 이루어진 계층적 모형이다. 메트릭스와 품질의 관계가 명확해야 한다는 원칙에는 벗어나지 않으나, 메트릭스가 품질 속성을 평가하는 하위 품질 속성의 의미를 띈다는 단점이 있다. 개발 프로세스에서는 직접적인 품질 속성이 되는 제품의 외부적 행위를 평가할 수 없으므로, ISO/IEC 9126의 품질 모형은 프로세스에서 품질 검증 기준으로 사용하기 어렵다. 본 논문의 메트릭스 적

용 방법은 H-SQM(2절 참조)을 이용한다. 이 품질 모형은 품질 속성간의 선후 조건 관계를 정의하고 이를 바탕으로 여러 품질 속성에 영향을 미치는 제품 특성을 하나의 품질 속성으로 귀속시킨 계층적 모형이다. 따라서 메트릭스와 품질과의 관계가 명확해야 한다는 원칙을 만족시킨다. 또한 메트릭스가 개발 프로세스에서 제어할 수 있는 제품 특성을 평가 대상으로 하기 때문에, 프로세스에서 품질 검증 기준으로 사용할 수 있다. H-SQM은 제품 특성을 기반으로 하면서도 '품질과의 연관성'을 만족시켜, 개발 프로세스에 적용된 메트릭스의 측정 결과를 품질의 관점에서 쉽게 분석하도록 한다.

② 품질 향상 제공(an effective mechanism for quality feedback) : '품질 향상 제공'은 개발 프로세스에 적용된 메트릭스가 그 단계에서 필요한 품질 향상에 대한 정보를 제공하는 것을 말한다. ISO/IEC 14598-3는 ISO/IEC 9126-3을 개발자를 위한 메트릭스로 사용한다. ISO/IEC 9126-3의 내부 메트릭스는 설계 명세서를 보고 완성된 시스템의 행위를 검토할 때 사용되는 메트릭스와 순수하게 내부적인 메트릭스(pure internal metrics)로 분류되어 제시된다. ISO/IEC 9126-3은 분석단계, 구현단계에서 검토되어야 할 메트릭스를 제시하지 못하고 있다. 순수하게 내부적인 메트릭스는 소프트웨어 품질 모형과는 포함되지 않고 별개로 제시되어 품질과의 연관성이 미약하다. 따라서 '품질 향상 제공'의 기준을 만족시키지 못한다. 본 논문의 메트릭스 적용 방법은 H-SQM의 메트릭스를 사용한다. H-SQM의 메트릭스는 제품 특성을 기반으로 제시된다. H-SQM은 품질 속성을 제품 특성으로 정의하고 제품 특성을 다시 소프트웨어의 구성 요소의 특성으로 세분화한다. 이 구성 요소를 각 개발 단계마다 배치함으로써 구성 요소를 측정하는 메트릭스를 각 계발 단계마다 적용하게 된다. 각 메트릭스는 관련 품질 속성과 개발 단계가 결정되어 있으므로, 개발 단계에서 필요한 품질 향상에 대한 정보를 제공하게 된다.

③ 효율적인 활동(simple and computable) : '효율적인 활동'은 품질을 검증하는 활동이 품질을 개선하기 위한 효율적이어야 한다는 의미이다. 메트릭스의 적용 방법에 있어서 개발 단계별로 품질과 관련하여 가장 적절한 메트릭스 집합을 설정함으로써, 메트릭스를 측정하는 활동에 소요되는 노력과 시간을 줄일 수 있다. ISO/IEC 14598-3은 모든 품질 속성을 모든 단계에 적용하는 방식으로 이루어지고 있다. 반면 본 논문에서 제시하는 방법에서는 품질 속성을 만족시키기 위한 제품 특성을 명시하고, 이 제품 특성을 향상시키기 위해 각 단계에서 검증해야할 요소를 찾음으로써, 각 품질 속성이 만들어지는 주요 단계가 구분된다. 따라서 본 논문에서 제시하는 방법은 한 단계에 집중적으로 품질이 결정되는 품질 속성의 경우, 품질의 검증 및 개선 활동에 그러한 품질 속성의 특징이 반영된다.

④ 의미 일관성(consistent in its use of units and dimensions) : '의미 일관성'은 메트릭스의 계산이 그 메트릭스가 만들어진 의미를 직관적으로 설명할 수 있어야 한다는 의미이다. 본 논문과 ISO/IEC 14598은 모두 간단하게 단일한 의미를 나타내는 메트릭스 혹은 이미 검증된 메트릭스만을 대상으로 하였으므로 '의미 일관성'을 만족한다.

⑤ 일반성(programming language independent) : 메트릭스는 분석 모형, 설계 모형, 프로그램의 구조에 기반이 되어야 하며 설계 도구나 프로그램 언어에 좌우되어서는 안된다는 의미이다. 이 일반성을 평가하기 위해서는 해당 메트릭스 적용 방법에 포함된 메트릭스 하나 하나를 검증해야 한다. 본 논문과 ISO/IEC 14598에서 사용하는 메트릭스에는 설계 도구나 프로그램 언어에 따라 다르게 계산되는 메트릭(metric)이 없으므로 '일반성'을 만족한다.

⑥ 객관성(consistent and objective) : 본 논문과 ISO/IEC 14598의 메트릭스는 메트릭스를 계산식으로 표현하였다. 해당 계산식에 의해 계산된 결과는 항상 같으므로, '객관성'을 만족한다.

표 3 메트릭스 속성 비교표

적용방법 / 평가기준	ISO/IEC 14598	본 메트릭스 적용방법
품질과의 연관성	△	O
품질 향상 제공	△	O
효율적인 활동	×	O
의미 일관성	O	O
구조 기반	O	O
객관성	O	O

(○:만족함, △:약간 만족함, ×:만족하지 않음)

본 논문의 메트릭스 적용 방법과 ISO/IEC 14598을 분석 기준에 따라 비교한 결과는 표 3과 같다. 표 3의 분석 결과는 3가지로 요약할 수 있다. 첫째 본 논문의 메트릭스 적용 방법은 ISO/IEC 14598보다 개발 프로세스에 메트릭스를 효율적으로 적용할 수 있다. 둘째 적용

된 메트릭스를 품질의 관점에서 쉽게 분석한다. 셋째 메트릭스의 측정 결과가 미흡한 경우 어떤 측면에서 품질 향상이 이루어져야 하는지를 보여주기가 용이하다. 따라서 기존의 방법보다 좀 더 효율적으로 품질을 점검할 수 있다.

5.2 품질의 개발 단계별 모델링

본 논문에서 제시한 방법으로 적용된 개발 단계마다의 메트릭스가 어느 품질 속성과 관련되는지에 대한 정보를 이용하여, 품질 속성과 개발 단계의 관련성을 그림으로 표현하였다. 그림 6, 7을 통해 각 단계의 메트릭스가 만족될 때 어떤 품질 속성이 향상되는지를 볼 수 있다.

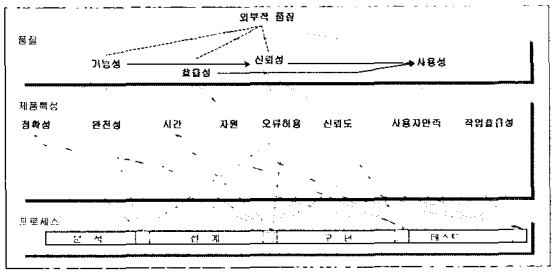

그림 6 외부적 품질과 개발 프로세스의 관계

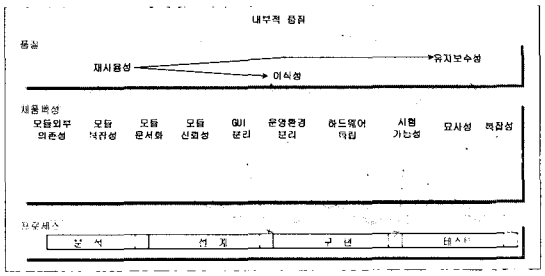

그림 7 내부적 품질과 개발 프로세스의 관계

그림 6, 7을 보면, 각 단계에서 각 품질 속성의 어떠한 측면이 중요하게 고려되어야 하는지를 알 수 있다. 외부적 품질 속성의 경우, 기능성과 신뢰성은 분석 부분에서, 효율성은 설계/구현 부분에서, 사용성은 설계/테스트 부분에서 중요하게 고려된다. 테스트 부분은 모든 품질 속성에 대해 이루어져야 한다. 내부적 품질 속성인 경우 분석 부분에서는 고려되지 않는다. 설계부분에서는 재사용성의 모듈화 측면과 이식성의 환경과의 분리성, 유지보수성의 복잡성이 고려된다. 구현 부분에서는 구현된 모듈의 신뢰성 및 복잡성, 환경과의 분리성에서 물리적인 고려 측면, 코드와 관련되는 특성이 검증 대상이 된다. 테스트에서는 전체 시스템의 속성인 이식성의 컴파일러 결과 일관성, 유지보수성의 전체 시스템의 복잡도 등이 검증 대상이 된다. 이러한 개발단계와 품질의 관계에 대한 모형을 통해 품질 속성 별로 관련되는 제품 특성 및 제품 특성이 구현되는 주요 개발 단계를 개괄적으로 볼 수 있다. 이로써 제품 품질 향상을 위해 실질적인 단계별 메트릭스 적용이 어떻게 이루어져야 하는지를 확인할 수 있다.

본 논문에서 제시한 방법이 개발 과정에서 소프트웨어의 품질에 대한 가시성을 확보하였는지를 점검하였다. 5.2절에서 보여주는 개발 과정에서 소프트웨어 품질의 가시성은 개발 프로세스에서 소프트웨어의 품질을 관리하는데 있어 필수적인 요소이다.

6. 결 론

본 논문에서는 소프트웨어 품질 모형을 기반으로 개발 프로세스 내 메트릭스 적용 방법을 제안하였다. 이로써 서론에서 제시한 두 가지 물음에 답변을 할 수 있다. 첫째 질문은 '해당 품질 점검 활동으로 품질 개선이 실질적으로 이루어지고 있는가?'이다. 본 논문은 H-SQM을 개발 프로세스에 맞추어 공정 분석도로 재구성하였다. 이를 바탕으로 개발 프로세스에 메트릭스를 적용하였다. 해당 제품 특성을 단계별 구성요소의 특성으로 세분화하여 메트릭스를 설정함으로써, 메트릭스의 측정 결과에 따라 품질의 어느 측면이 향상되어야 할지를 검증할 수 있다. 둘째 질문은 '해당 품질 점검 활동이 품질 개선을 위해 효율적인가?'이다. 본 논문에서 사용하는 H-SQM은 소프트웨어 품질 모형의 하위 레벨에서의 중복성을 해결하여 품질 속성마다 반복되어 측정되는 메트릭스를 단일화 하였다. 그리하여 소프트웨어 개발 프로세스에 품질 개선을 위한 효율적인 메트릭스를 적용할 수 있게 되었다.

이를 계속 보완해 나가기 위해서 다음과 같은 작업이 필요하다. 첫째, 메트릭스 분석 작업이 다. 현재의 품질 모형은 소프트웨어 메트릭스 적용에 국한되어 있다. 메트릭스의 평가 결과를 모으고 분석하는 방법을 제시하여 검증 시점의 제품 평가를 할 수 있도록 해야 한다. 그러기 위하여 품질 요소에 중요 가중치의 부여 및 품질 평가 방법의 고안이 필요하다. 둘째, 각 시스템과 통합될 개발분에 대해 품질을 측정하여 전체 시스템의 품질을 예측하는 기법이 있어야 한다. 점진적 개발 프로세스의 장점은 각 개발분이 완성될 때마다 실행 가능한 시스템이 나오는 점이다. 이미 Cleanroom 프로세스[13]

에서는 개발분이 통합될 때마다 신뢰성을 측정하여 전체 시스템의 신뢰성을 예측한다. 이를 확장하여 전체 시스템의 품질을 예측하는 방법을 고안한다. 기타 본 논문에서 제안하는 메트릭스 적용 방법을 사용해서 메트릭스를 기반으로 한 중간 산출물 품질 분석 및 전체 시스템의 품질 예측에 대한 연구를 진행함으로써, 좀 더 실질적인 메트릭스 적용 방법으로 만들어 나갈 수 있을 것이다.

참 고 문 헌

[1] Kitchenham, Barbara and Shari Lawrence Pfleeger. "Software Quality: The Elusive Target". IEEE Software, pp. 12-21, Vol. 13, No. 5. Jan. 1996.

[2] Ejiou, L., Software Engineering with Formal Metrics, QED Publishing, 1991.

[3] ISO/IEC 9126: Information Technology - Software Quality Characteristics and Metrics(Draft), JTC1 SC7 WG6(Evaluation & Metrics) Documents, 1996.

[4] ISO/IEC 14598: Information Technology - Software Product Evaluation, JTC1 SC7 WG6(Evaluation & Metrics) Documents, 1996.

[5] Dromey, R. Geoff. "Cornering the Chimera". IEEE Software, Vol. 13, No. 5, pp. 33-43, Jan. 1996.

[6] 이선아, 최병주. "개발 관점의 계층적 소프트웨어 품질 모형 : H-SQM". 정보과학회논문지(B) 1999. 12.

[7] David G. and Kecheng Liu, "Quality Metrics for Object-Oriented Design". JOOP, January, 1998.

[8] Berard, Edward V.. "Metrics for Object-Oriented Software Engineering". an Internet posting on Comp.software-eng, Jan. 28, 1995.

[9] Chidamber, Shyam R. and Chris F. Kemerer. "Towards A Metrics suite for object oriented design". OOPSLA'91, pp. 197-211, 1991.

[10] Chidamber, Shyam R. and Chris F. Kemerer. "A Metrics suite for object oriented design". IEEE Transactions on Software Engineering, vol. 20, no.6, pp.476-493, Jun, 1994.

[11] Binder, Robert V.. "Testing Object-Oriented Systems: A Status Report". American Programmer, vol. 7, no. 4, April 1994, pp. 22-29.

[12] 정보통신용 시스템 개발방법론, 마르미-II 절차서 Version1.0, 한국전자통신연구원, 1998.

[13] Linger, R., and Trammell, C., "Cleanroom Software Engineering Reference Model 1.0", CMU/SEI-96-TR-022, SEI, Nov., 1996.

이 선 아

1997년 2월 이화여자대학교 전자계산학과 학사. 1999년 2월 이화여자대학교 대학원 컴퓨터학과 석사. 현재 삼성전자 중앙연구소 소프트웨어센터 연구원. 관심분야는 소프트웨어 품질 관리, 소프트웨어 품질 척도, 객체지향 소프트웨어 개발 프로세스

최 병 주

정보과학회논문지 : 소프트웨어및응용 제 27 권 제 2 호 참조