재사용에 기반한 컴포넌트 평가 메트릭의 제안

○
이하용*, 양해술*, 황석형**
*호서대학교 벤처전문대학원
**선문대학교 컴퓨터정보학부
e-mail:insq@unitel.co.kr

Proposal of Component Evaluation Metrics based on Reuse

Ha-Yong Lee*, Hae-Sool Yang*, Suk-Hyung Hwang*
*Graduate School of Venture, Hoseo University
**Sun-Moon Univ. of Information & Computer Science

요약

소프트웨어를 기능 단위인 부품으로 분류하고 소프트웨어 부품을 조합하여 개발해 나가는 컴포넌트 기술은 개발비용의 감소와 소프트웨어 전체의 신뢰성 향상을 목적으로 주목되고 있다. 그러나 재사용의 대상이 되는 각 컴포넌트의 신뢰성이 낮고 오히려 전체의 신뢰성을 손상시키는 결과가 나타나고 있다. 결국, 컴포넌트의 신뢰성 향상이 매우 중요하고 이를 위해 컴포넌트에 대해 품질을 측정할 필요가 있다. 본 연구에서는 소스코드가 공개되지 않은 소프트웨어 컴포넌트에 대해 블랙박스의 개념으로 보고 정적 측면에 대한 품질을 측정하는 기법을 제안한다.

1. 서론

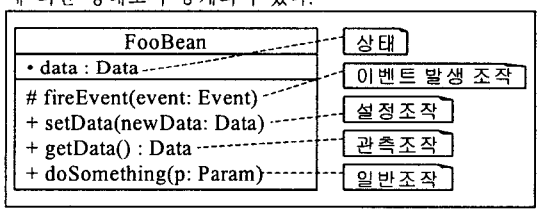
소프트웨어를 기능 단위인 부품으로 분류하고 소프트웨어 부품을 조합하여 개발해 나가는 컴포넌트 기술은 개발비용의 감소와 소프트웨어 전체의 신뢰성 향상을 목적으로 주목되고 있다. 그러나 재사용의 대상이 되는 각 컴포넌트의 신뢰성이 낮고 오히려 전체의 신뢰성을 손상시키는 결과가 나타나고 있다. 결국, 컴포넌트의 신뢰성 향상이 매우 중요하고 이를 위해 컴포넌트에 대해 품질을 측정할 필요가 있다. 소프트웨어에 대한 품질측정의 중요성은 이미 이전부터 지적되어 왔고 객체지향에 관련된 다수의 정적 측정지표(제품 메트릭)가 제안되어 있다[1]. 그럼에도 종래에 제안된 측정 지표를 사용한 품질측정 기법은 대상 제품의 소스코드를 해석해야 하기 때문에 소스코드를 참조할 수 없는 컴포넌트에 대해 적용하기 곤란하다. 본 연구에서는 소스코드가 공개되지 않은 소프트웨어 컴포넌트에 대해 블랙박스의 개념으로 보고 정적 측면에 대한 품질을 측정하는 기법을 제안한다.

2. 컴포넌트 기술

컴포넌트는 수준에 따라 3종으로 분류되는데, 비니니스 로직이 캡슐화된 비지니스 컴포넌트와 최소단위 컴포넌트를 조합하여 어플리케이션 로직을 부가한 복합 컴포넌트, RAD(Rapid Application Development) 툴 등에 부속되는 컴포넌트 라이브러리가 제공하는 컴포넌트이다. 본 연구에서는 컴포넌트 라이브러리가 제공하는 컴포넌트를 품질측정의 대상으로 한다. 이러한 컴포넌트의 기본요소는 다음과 같다.

- 특성 : 컴포넌트의 외관이나 행동을 결정하기 위한 속성
- 관측조작 : 상태를 외부로부터 관측하는 조작
- 설정조작 : 상태를 새롭게 설정하는 조작
- 일반조작 : 컴포넌트가 제공하는 기능을 외부로부터 실행가능한 조작
- 이 벤 트 : 수동적 제어의 기점, 외부에 대한 갱신 통지

컴포넌트 환경을 실현가능한 컴포넌트 시스템으로서 JavaBeans나 ActiveX가 있다. JavaBeans 컴포넌트의 내부에 있는 기본요소를 (그림 1)에 나타내었다. (그림 1)에서는 속성 data가 설정조작 setData 및 관측조작 getData에 의한 상태로서 공개되어 있다.



(그림 1) JavaBeans 컴포넌트의 표준적 내부 기본요소

3. 소프트웨어 메트릭

3.1 종래의 객체지향 프로덕트 메트릭

폭넓게 이용되고 있는 프로덕트 메트릭으로서 CK 메트릭이 있고, 객체지향 프로그램의 설계단계에서 복잡도를 측정하고, 클래스마다 중요도가 부여된 메소드의 수(WMC), 계층의 깊이(DIT), 서브클래스의 수(NOC), 클래스간 결합도(CBO), 클래스에 대한 반응(RFC), 메소드 응집도(LCOM)의 6개 메트릭으로 구성된다. 이중에 WMC 및 DIT는 소스코드를 열어볼 수 없는 컴포넌트에 대해서도 적용할 수 있다.

- 클래스마다 중요도가 부여된 메소드의 수 (WMC: Weighted Methods per Class)

해당 클래스에 있는 메소드의 중요도를 측정한다. 본 연구에서는 중요도를 모두 1로 한다. WMC가 높을수록, 복잡하고 보존성이 낮은 것을 의미한다.

- 계층의 깊이(DIT: Depth of Inheritance Tree)

해당 클래스와 가까운 클래스의 수를 측정한다. DIT가 높을수록, 상속되고 있는 변수나 메소드가 많은 것을 의미한다.

Binder는 테스트 용이성에 관련되어 수많은 메트릭들을 정리하고 있다[2]. 그중에 캡슐화에 관한 공개메소드비율(PAP)과 공개속성비율(PAD)은 소스코드를 볼수 없는 컴포넌트에 대해 적용 가능하다.

- 공개 메소드 비율(PAP: Percent Public And Protected)

클래스 중의 public, protected인 메소드 수의 비율을 측정한다. PAP가 높으면 실행시에 외부로부터 동작에 영향을 받을 가능성이 큰 테스트를 할 경우, 동작을 파악하기 곤란해지고, 반대로 낮다면 테스트가 곤란해진다.

- 공개속성비율(PAD: Public Access to Data members)

클래스 중의 public, protected인 속성수의 비율을 측정한다. PAD가 높으면 테스트시의 거동파악이 용이해 지지만 캡슐화의 파괴를 초래한다.

이와 같은 종래의 메트릭은 본래 클래스군보다 구성된 시스템 전체의 품질향상을 목적으로 하고 시스템내 저결합, 클래스내 고응집을 기본 개념으로 하여 관련된 일련의 클래스군을 측정 대상으로 한다.

3.2 컴포넌트 메트릭의 과제

대상을 컴포넌트에 특화한 측정 툴로서 개발한 컴포넌트가 JavaBeans 컴포넌트의 규약을 만족시키는가를 검증하는 BeanLint[3]가 있다. 또한, 소프트웨어 부품의 조합을 기본으로 한 개발 프로세스에서 프로젝트 메트릭은 다수 제안되어 있다[4]. 프로덕트 메트릭으로서 광의의 컴포넌트로서 클래스라이브러리·패키지를 대상으로 한 것이 제안되어 있지만, 본 연구에서 다루는 컴포넌트에 대해 적용하기 어렵다. 따라서, 컴포넌트의 재사용성을 블랙박스적으로 측정 가능한 품질측정의 구조가 강하게 요구된다.

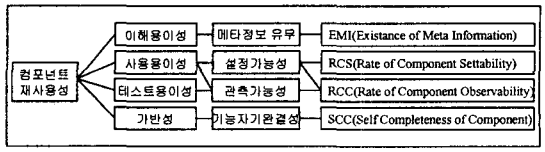
4. 재사용 특성을 고려한 컴포넌트 메트릭의 제안

컴포넌트로부터 자동추출 가능한 기본측정 요소로부터 그 성질을 측정하는 메트릭을 제안한다.

4.1 컴포넌트 특성 모델

본 연구에서 착안한 재사용성에 관한 컴포넌트 특성 모델을, (그림 2)에 나타내었다. 컴포넌트 사용자는 RAD 툴을 사용하여 컴포넌트가 가진 속성의 내부상태로서 공개된 것에 대해 시각적인 상태표에 의해 그 값을 확인하는 것이 가능하다. 속성의 공개·비공개는 컴포넌트 개발자가 컴포넌트의 메타 정보를 명시적으로 제시하는 것이 가능하게 하고 그 적절한 설정이 컴포넌트에 대한 이해용이성으로 귀결되고 있다. 컴포넌트의 사용용이성을 고려할 때, 컴포넌트 개발자는 속성의 내부에서, 본질적으로 그 동작에 크게 관련된 것에 대해서는 적절히 특성으로서 공개하고 필요에 따라 그 값을 외부에서 설정할 수 있도록 할 필요가 있다. 본 연구에서는 이 내부상태의 공개를 관측가능성이라 부르고, 외부로부터 변경가능한 경우를 설정가능성이라 부른다. 또한, 일반적으로 고품질인 컴포넌트는 높은 테스트성을 가지며 테스트용이성은 관측가능성과 관련이 있다.

컴포넌트가 제공하는 기능은 어느 정도 내부해결할 필요가 있는데 이것을 기능자기완결성이라 부른다.



(그림 2) 컴포넌트 재사용성 모델과 메트릭

4.2 기본측정요소

이 장에서는 JavaBeans를 메트릭 측정환경으로 한다. JavaBeans에서는 대상으로 하는 컴포넌트의 상태, 조작, 이벤트에 관한 정보를 외부에 공개하는 메타 정보를 BeansInfo 객체로서 기술할 수 있다.

JavaBeans의 컴포넌트에 대한 기본적인 측정요소이다.

- 컴포넌트 사이즈 : 컴포넌트의 정적인 크기
- 메타정보 : 개발자에 의한 명시적인 메타정보의 유무
- 속성 : 객체지향 클래스로서 갖는 속성의 수
- 상태 : 상태수, 대응하는 관측조작의 유무, 대응하는 설정조작의 유무, 상태의 형정보
- 관측조작 : 명명규칙에서 벗어남, 대응하는 상태와 돌아온 값의 형의 일치
- 설정조작 : 명명규칙에서 벗어남, 대응하는 상태와 인수의 형의 일치
- 일반조작 : 일반조작수, 인수의 형, 돌아온 값의 형

4.3 재사용 특성 컴포넌트 메트릭

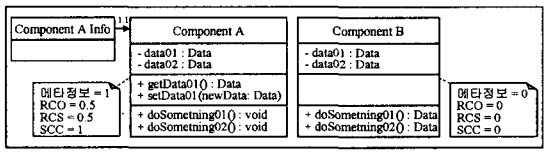
기본 측정요소의 측정결과를 사용하여, 4.1절에서 제안한 컴포넌트의 4개 성질에 대해 메트릭을 설정한다. 기술형식은 문헌[1]에 따른다. 측정한 값이 유효구간 내에 든 컴포넌트의 품질은 높다고 판단할 수 있다.

4.3.1 명시적인 메타정보의 유무 (EMI: Existence of Meta Information)

① 정의 : 대상 컴포넌트에 대한 메타 정보가 명시적으로 보이고 있는지 없는지 명시적인 메타 정보의 유무 (EMI)::=1(메타정보 존재), 0(존재안함)

② 의미 : 이 값이 1인 경우에 메타정보의 제시가 있고, 컴포넌트 개발자가 생각하는 사용방법을 용이하게 파악할 수 있다는 것을 의미한다.

③ 예 : 명시적인 메타정보, 상태의 관측조작, 설정조작 및 일반조작을 갖는 컴포넌트의 예를 (그림 3)에 나타내었다.



(그림 3) 컴포넌트의 메타정보의 유무, 관측가능성, 설정가능성, 기능자기완결성

(그림 3)에서 컴포넌트 A에는 개발자에 의해 메타정보인 ComponentAInfo가 명시적으로 나타나 있다. 이 경우, 컴포넌트A는 컴포넌트B보다 이해용이성이 높다고 말한다.

④ 유효구간(한계값) : 5.3절에서 평가 결과, EMI=1을 얻었다.

⑤ 대책 : 명시적인 메타정보가 존재하지 않는 경우에는 개발자측은 메타정보를 준비한다.

4.3.2 관측가능성(RCO: Rate of Component Observability)

① 정의 : 컴포넌트가 가진 모든 속성의 내부, 외부보다

관측가능한 상태의 비율을 관측가능성이라 정의한다.

$$\text{관측가능성} = \frac{P_r}{A}\left(\begin{matrix}P_r : \text{외부보다관측가능한상태수}\\ A : \text{전속성수}\end{matrix}\right)$$

② 의미 : 이 값이 높을 수록 내부상태의 공개 정보가 많은 것을 의미한다.

③ 예 : (그림 3)에서 컴포넌트 A, B는 공동으로 상태 data01, data02를 갖지만, A에는 data01에 대한 관측조작 getData01이 사용되고 있다. 이 경우, 컴포넌트A는 컴포넌트B보다 관측가능성이 높다.

④ 유효구간(한계값) : 5.3절에서 평가의 결과, RCO = 0.2 ~0.4를 얻는다.

⑤ 대책 : 개발자측의 대책은 RCO가 유효구간보다 높을 때는 순서대로 대응하는 관측조작을 은폐한다. 반대로 RCO가 유효구간보다 낮을 때는 공개해야할 속성으로부터 순서대로 대응하는 관측조작을 추가한다. 사용자측의 대책은 RCO가 유효구간보다 높을 때는 대상 컴포넌트를 내포하는 Adapter 클래스[15]를 작성하고 과잉 관측조작을 은폐한다.

5. 평가

JavaBeans의 정적측면의 품질측정 툴 "BeanMetrix"를 Java 언어로 개발하고 평가했다. 평가 샘플은 JARS.COM [5](이하 JAS라 함)에 수집되어 있는 JavaBeans 컴포넌트군을 사용했다. JARS에서는 컴포넌트를 관련된 범주로 분류하고 그것들에 대해 여러 사람이 장시간에 걸쳐 독자적인 평가를 한다. 평가항목은 다음의 3항목으로 각 점수를 종합하여 별점 4개를 최고점으로 각 컴포넌트에 대해 평가하고 있다.

- 표현성(Presentation):외적인 장점(400점 만점)
- 기능성(*Functionality*):사용용이, 안정성(300점 만점)
- 신규성(Originality):새로운 아이디어, 컨셉(300점 만점)

본 연구에서는 4단계 평가를 구간[0, 1]이 되도록 정규화한 값을 JARS 평가로 정의한다. 평가방법으로서 메타정보의 유무에 관해서는 평가 샘플을 JARS 평가 0이상 0.33미만, 0.33이상 0.66미만, 0.66이상 1이하의 3구간으로 분류하고, 각 구간에서의 분포를 갖는다. 그외의 각 메트릭에 관해서는 각각의 평가분포를 얻고 메트릭 값의 유효구간에서 1/10씩 단락을 짓는다. 각 구간에서 해당 샘플수가 균일하다면, 구간마다 해당 샘플의 JARS 평가평균치를 산출하고 추천 메트릭값으로서 사용하는 것이 고려되지만, 구간마다의 샘플수가 평균치에서 벗어나 보이기 때문에 단순한 평균값으로는 일반적인 경향으로서 보다 상세한 재사용성의 정도를 판단할 수 없다. 그리고 각 구간에서 해당 샘플수를 메트릭값으로 반영하기 위해, 구간마다 해당 샘플의 JARS 평가의 종합을 산출하고, 이것으로부터 종합의 최고값의 구간과 최고값의 85% 이상의 값을 나타내는 구간을 본 연구에서 제안하는 메트릭을 JARS 평가에 관련이 있는 유효구간이라 한다. 이 유효구간 내에 해당하는 컴포넌트의 재사용성은 높다고 판단한다.

<표 1> 카테고리별 샘플수와 JARS 평가 평균치

카테고리	샘플수	JARS 평가평균치
Programming	31	0.47
WWW	2	0.88
Game	2	0.88
Utilities	7	0.61
합계	42	0.53

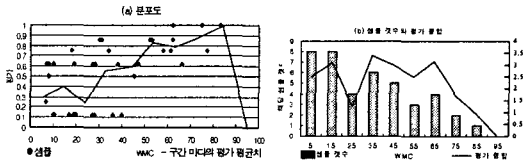
평가 샘플은 메타정보의 유무에 관한 상기 3구간으로부터 14개씩, 합계 42개의 컴포넌트를 무작위로 추출했다. 여전히, 각 샘플의 범주에 속하는 컴포넌트에 대한 JARS 평가의 평균치는 <표 1>의 결과가 되고, 범주마다 JARS 평가의 큰 차이는 보이지 않았다.

5.1 종래의 객체지향 메트릭 측정 결과

종래의 기법은 화이트박스적인 측정이 필요하지만 WMC, DIT, PAP는 클래스 개체를 측정대상으로 하기 때문에, 평가 샘플에 대해 측정할 수 있다.

5.1.1 클래스마다 중요도가 부여된 메소드수(WMC) 측정 결과

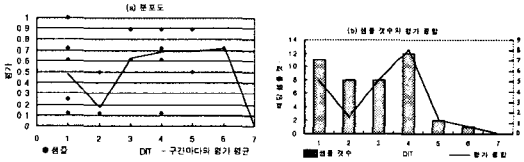
(그림 4)에서 유효구간으로서 10~20, 30~50, 60~70의 3구간이 얻어지고, JARS 평가가 동일한 컴포넌트군에 대해서 WMC값이 평균치에서 벗어나므로 WMC는 컴포넌트에 대한 품질측정지표로서 사용곤란하다.



(그림 4) WMC 분포도와 구간별 샘플과 JARS 평가 종합 상관도

5.1.2 계층의 깊이(DIT) 측정 결과

(그림 5)에서 유효구간으로서 4가 얻어졌다. 객체지향 클래스의 적정 계층 깊이에 가까운 평가결과가 되었다. JavaBeans에 있어서 DIT를 품질측정에 사용 가능하다. 다만, DIT=1에 대해서도 비교적 JARS 평가가 높은 컴포넌트가 존재하기 때문에, DIT만을 사용하여 품질측정을 하는 것은 어렵다고 생각된다.



(그림 5) DIT 분포도와 JARS 평가평균치, 구간별 샘플과 JARS 평가 종합 상관도

5.2 기본 요소 측정 결과

본 기법에서 측정하는 모든 평가 샘플의 사이즈, 상태수, 일반조작수의 평균값은 사이즈 9429.7바이트, 상태수 8.38, 일반 조작수 12.33이었다.

관측·설정조작에서 명명규칙으로부터 벗어난 정도, 대상 상태와 조작인수의 형일치도는 평가 샘플 모두가 기준을 충족하고 있다. 이것은 JARS에 컴포넌트를 투고할 때에 명명규칙 등의 표준규칙을 준수하는 것은 필연사항이기 때문이라고 생각된다.

5.3 컴포넌트 메트릭 측정 결과

5.3.1 명시적인 메타정보의 유무(EMI) 측정 결과

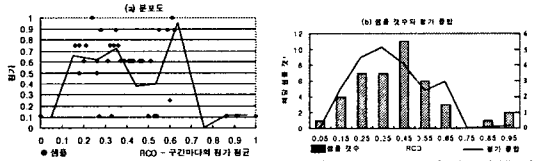
각구간별의 EMI 측정결과를 <표 2>에 나타내었다. 90%의 컴포넌트에 대해서 BeanInfo 객체가 사용되고 있는 결과가 되었다. BeanInfo 객체는 컴포넌트 사용자에 대해서 컴포넌트 개발자가 그 사용법을 명시하는 것이고 개발자는 미리 독자적으로 BeanInfo 객체를 사용한 것은 필수사항이라고 할 수 있다. 따라서, EMI의 유효구간을 1로 한다.

<표 2> 구간 마다의 해당 컴포넌트수

	.33미만	0.33이상0.67미만	0.67이상
1	13	12	13
0	1	2	1

5.3.2 관측가능성(RCO) 측정 결과

RCO의 모든 평가 샘플에 대한 측정 결과를 (그림 6)에 나타내었다. 유효구간으로서 0.2~0.4를 얻은 결과가 되었다. 결국, 모든 속성의 1/3 정도는 상태로서 관측조작을 수반해 외부에 공개해야만 한다는 것을 알 수 있다. 동구간에 해당하는 컴포넌트는 전체의 33%이고, JARS 평가에 관계하지 않고 많은 컴포넌트가 해당하는 것만은 아니기 때문에 동구간은 품질의 판단시에 유효하다고 할 수 있다. 또한, RCO가 높은 컴포넌트의 JARS 평가가 낮은 경향이 보인다. 이것은 상태의 대부분이 외부로부터 관측가능한 컴포넌트는 요망되지 않고, 어느 정도까지 비공개로 해야 됨을 의미한다. 이상에 의해, RCO는 컴포넌트 특성을 고려한 품질측정 지표로서 유용한 것을 알 수 있다.



(그림 6) RCO 분포도와 JARS 평가평균치, 구간별 샘플과 JARS 평가 종합 상관도

5.4 어플리케이션 작성 실험에 의한 유효성 검증

관측가능성, 설정가능성의 영향을 검증하기 위해, 컴포넌트를 사용하여 코딩 및 결합에 의한 2종류의 어플리케이션 작성실험을 했다. 각 실험을 3명이 실시했다. 코딩실험에서는 컴포넌트 CalendarBean[6]을 사용하고, 선택된 일부를 통지하는 어플리케이션을 편집기를 사용하여 코딩하였다. 작성에 즈음해서 CalendarBean의 본래의 판과, 목적으로 하는 어플리케이션 기능실험을 위해 사용하는 상태에 영향 없는 범위에서 관측가능성, 설정가능성을 유효구간 내에 설정한 판, 기능실현을 위해 최소한 필요한 상태뿐만아니라 공개한 판의 3종을 사용하고 각각을 사용하여 어플리케이션의 완성까지 필요한 시간·컴파일 회수·잘못 사용된 상태수를 측정한다. 본 실험에서 사용한 컴포넌트는 소스 코드를 열어볼 수 없기 때문에 각 판의 차이는 피험자에 주어지는 API 도큐멘트에 반영된다.

결합 실험에서는 컴포넌트 FukaGraphBean[7]를 사용하고, 항목명·요소값을 입력하고 그래프 화면을 보존하는 어플리케이션을 RAD 툴을 이용하여 비주얼한 결합 조작에 의해 작성한다. 작성시에 FukaGraphBean의 원래의 판, 관측가능성·설정가능성을 유효구간내에 거두었던 판, 및 최소한 필요한 상태의 공개한 판의 3종을 사용하고 어플리케이션 완성까지 필요한 시간·RAD 툴상에서의 주요조작 회수·잘못된 사용한 상태수를 측정했다.

양 실험에서 사용한 컴포넌트 각 3판씩의 관측가능성·설정가능성을 <표 3>에 나타내었다.

양 실험에서 사용하는 컴포넌트가 원래의 판인 것보다, 관측가능성·설정가능성을 유효구간 내에 얻었던 판을 사용한 방법이 작성시간이 짧았다. 또한, 본래의 판을 사용한 때에는 양 실험에서 잘못된 상태의 사용이 보이지만, 유효구간 내 판, 최소한 상태판에서는 볼 수 없는 관측가능성·설정가능성을 확보한 효과가 보였다. 코딩 실험에서는 유효구간 내에 얻었던 판과 최상한계의 상태판이라고 해서 어플리케이션 작성 시간에 큰 차이가 보이지 않는다. 코딩을 주체로서 컴포넌트를 사용할 때, 관측가능성·설정가능성의 유효구간은 사용용이성이 높은 것을 나타내는 기준으로서 타당한 것을 알 수 있었다. 결합 실험에서는 유효구간 내에 얻었던 판보다 최소한계의 상태 판의 방법이 어플리케이션 작성 시간이 짧았다. 이것은 사용해야 할 상태가 특정되어 있기 때문에 작성이 용이했던 것이 원인이다. 그러나 컴포넌트가 다른 문맥에서 사용된 것을 고려하면, 관측가능성·설정가능성은 어느 정도의 높이를 가지고 있는 방법이 바람직하다. RAD 툴 상에서의 주요조작 회수에 대해서 양판의 결과는 같은 것으로부터, RAD 툴에 의해 컴포넌트를 사용할 때에 관측가능성·설정가능성의 유효구간은 사용용이성이 높은 것을 나타내는 기준으로서 타당한 것을 알 수 있다.

<표 3> 컴포넌트의 각 판에서 관측가능성·설정가능성

컴포넌트명	원래의 판(RCO.RCS)	유효구간내판	최소한계 상태판
CalendarBean	0.78, 0,89	0.33, 0.33	0.22, 0.22
FukaGraphBean	1.00, 1.00	0.38, 0.38	0.23, 0.23

6. 결론

본 연구에서는 컴포넌트 단체에 대해서 블랙박스적으로 그 정적측면의 품질을 측정하기 위해 컴포넌트를 재사용 특성을 측정가능한 메타정보의 유무·관측가능성·설정가능성·기능자기완결성의 4종의 메트릭을 제안하고, JARS 평가와의 상관관계로부터 그 타당성을 확인했다. 또한, 어플리케이션 작성실험을 하여, 제안한 메트릭의 유용성을 확인했다. 본 연구에서 제안한 각 메트릭은 컴포넌트의 특정부분에 착안한 원시적인 메트릭이고, 컴포넌트 전체로서의 품질을 측정하기 위해서는 향후 원시 메트릭과 관련된 항목마다 결합 모델[8]에 기반하여 조합된 것이 필요하다고 생각된다.

참고문헌

[1] M. Lorenz, J.Kidd, "Object-oriented software metrics", PrenticeHall, 1995.

[2] R. Binder, "Design for Testability in Object-Oriented Systems", Communications of the ACM, Vol.37, No.9, 1994.

[3] M. Joknson, "BeanLink:A JavaBeans Troublesh ooting Tool", JavaWorld, December, 1998.

[4] T. Takeshita, "Metrics and Risks of CBSE", Symposium on Assessment of Software Tools and Technologies, 1997.

[5] JARS.COM, http://www.jars.com/

[6] IBM alphaBeans, http://www.alphaworks.ibm.com/alphabeans/

[7] FukaBeans, http://www.fuka.info.waseda.ac.jp/Project/CBSE/fukabeans/

[8] S. Lai, C. Yang, "A Software Metric Combination Model for Software Reuse", Asia-Pacific Software Engineering Conference, 1998.

[9] 양해술, 이하용, "컴포넌트 기반 소프트웨어 시험평가 모듈의 개발", 한국정보처리학회 소프트웨어공학연구회지, 2001. 12.