소프트웨어 컴포넌트의 품질 평가 모델

김지혁*, 김수동*
*숭실대학교 컴퓨터학과

The Quality Evaluation Model of Software Component

Kim, Ji Hyeok° , Kim, Soo Dong°
Department of Computer Science, Soongsil University
E-mail : jhkim@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

요 약

소프트웨어 컴포넌트는 특정 도메인 내에서 패밀리 멤버들 사이의 공통 기능성을 구현한 것이다. 하나의 멤버를 위해 개발된 컴포넌트는 다양한 패밀리 멤버에서 재사용하기 어렵다. 그러므로, 컴포넌트를 개발할 경우에 다양한 멤버에 대해 고려해야 한다. 그러므로 소프트웨어 컴포넌트의 품질 측정은 성공적인 컴포넌트 기반 시스템 개발을 위한 중요한 선행작업이다. 본 논문에서는 소프트웨어 컴포넌트의 품질을 평가하기 위한 품질 평가 모델을 제안한다. 소프트웨어 컴포넌트를 측정하기 위해서 소프트웨어 컴포넌트의 특징을 식별하고, 식별된 특징을 기반으로 하여 소프트웨어 컴포넌트의 품질 평가 모델을 제안한다. 제안된 품질 평가 모델은 특성, 부특성, 메트릭으로 구성된다.

1. 서론

컴포넌트 기반 개발은 큰 단위의 소프트웨어 재사용을 위한 약속으로써 학계와 산업계에서 점점 각광받고 있는 재사용 접근 방법중의 하나이다. 소프트웨어 컴포넌트는 비즈니스 개념과 프로세스의 소프트웨어 구현인 비즈니스 컴포넌트이다[1][2].

그러므로, 여러 패밀리 멤버에서 소프트웨어 컴포넌트를 사용하여 시스템을 개발한다. 그러나 소프트웨어 컴포넌트를 사용하기 위해서는 소프트웨어 컴포넌트의 품질을 평가해야 한다. 소프트웨어 컴포넌트는 여러 패밀리 멤버를 위한 가변성을 제공하기 때문에, 소프트웨어 컴포넌트의 품질 평가는 성공적인 컴포넌트 기반 시스템 개발을 위해 중요한 선행작업이다.

본 논문에서는 먼저 소프트웨어 컴포넌트의 특징을 식별한다. 그 다음 식별된 특징으로부터 소프트웨어 컴포넌트를 위한 품질 평가 모델을 도출한다.

본 논문은 다음과 같이 구성된다. 2장에서는 소프트웨어 품질 모델에 관련된 연구를 조사하고 제한을 논의한다. 3장에서는 소프트웨어 컴포넌트의 특징을 정의하고, 4장에서는 품질 속성인 특성과 부특성을 3장에서 식별된 특징을 기반으로 도출한다. 5장에서는 도출된 품질 속성의 메트릭을 정의하고 6장에서 이러한 메트릭의 적용을 살펴보

고 7장에서 결론을 맺는다.

2. 관련 연구

소프트웨어 품질 모델은 소프트웨어의 품질 속성 및 속성들 간의 관계에 대한 명세서이다. ISO 9126 [3]은 범용 소프트웨어를 위한 대표적인 품질 모델이다. 품질 모델은 특성(Characteristic) 및 부특성(Sub-characteristic), 그리고 각 부특성을 계산하기 위한 메트릭(metric)을 포함한다. 이러한 품질 모델은 소프트웨어 컴포넌트와 같은 재사용 가능한 모듈 보다는 개발이 완료된 소프트웨어 제품들을 평가 대상으로 한다.

또 다른 대표적인 품질 모델로는 McCall [4]의 품질 모델이 있다. ISO 품질 모델의 특성에 해당되는 Factor 와 부특성에 해당되는 Criteria 로 구성된다. McCall 의 품질 모델은 11 개의 품질 요소와 25 개의 품질 항목으로 이루어져 있다. ISO 9126 품질 모델과 마찬가지로 McCall 모델 또한 소프트웨어 컴포넌트에 적용 가능하지 않다. Boehm [5]의 품질 모델은 개발 완료된 소프트웨어 어플리케이션을 위한 품질 프레임워크를 제공한다.

3. S/W 컴포넌트의 특징

품질 모델을 도출하기 위해서 본 연구에서는 우선 소프트웨어 컴포넌트의 특징을 고려한다. 그러므로 본 장에서는 대표적인 컴포넌트 연구를 기반으로 하여 소프트웨어 컴포넌트의 특징을 식별한다.

컴포넌트는 높은 모듈성을 제공한다. 컴포넌트는 일반적으로 여러 객체와 클래스로 구성된다. 그래서 컴포넌트는 객체지향프로그래밍의 객체 보다 큰 단위이다 [6]. 컴포넌트는 소프트웨어 개발의 긴밀히 연결된 패키지이다. 컴포넌트는 독립적으로 개발되고 인도될 수 있기 때문에 provide 인터페이스와 require 인터페이스를 위해 명시적이고 잘 명세된 인터페이스를 가진다. 또한, 컴포넌트는 컴포넌트의 속성 일부분을 커스터마이즈해서 다른 컴포넌트들로 구성할 수 있다.

컴포넌트는 공통 특징을 제공한다. 컴포넌트는 공통 특징을 제공한다. 컴포넌트는 내적 구조의 재사용과 비스니스 분야 안에서 내적 공통성을 증가시키기 위해 패밀리 멤버 사이의 공통 기능성을 획득해야 한다. 이 특징은 객체 지향 재사용과 컴포넌트 기반 재사용을 구별한다. 만약 표준 도메인 모델이 존재하면, 컴포넌트는 표준을 준수해야 한다. 최근에 OMG와 마이크로소프트 같은 조직에서 재정, 텔레커뮤니케이션과 같은 중요한 비즈니스 섹터안에서 표준 도메인 모델을 개발해왔다.

컴포넌트는 가변성을 획득해야 한다. 컴포넌트는 다양한 패밀리 멤버에서 사용되기 때문에 조직들 간의 가변성을 획득해야 한다. 가변성 분석은 공통성 분석 이후에 수행되며, 식별된 가변성은 커스터마이제이션 메커니즘에 영향을 받는다. 패밀리 멤버 사이의 공통 기능성에서는 logic, attribute, workflow와 같은 사소한 가변성이 존재한다.

컴포넌트는 계약으로써 대치될 수 있다. 컴포넌트는 생산자와 소비자사이의 계약을 만족해야 한다. 이러한 계약은 provide 인터페이스와 require 인터페이스로써 표현된다. 만약 인터페이스 사이의 계약이 만족하면 컴포넌트는 다른 컴포넌트와 합성될 수 있으며 표준 인터페이스를 만족하는 다른 컴포넌트로 대치될 수 있다. 합성 메커니즘은 전형적으로 컴포넌트의 참조모델에 의해 정의되고, 그 메커니즘은 컴포넌트 플랫폼에 의해 구현된다. 어플리케이션 안에 통합된 컴포넌트는 어플리케이션의 다른 부분의 변경 없이 효과적으로 컴포넌트를 새로운 버전으로 대치할 수 있을 것이다.

컴포넌트는 커스터마이즈가 가능하다. 공통성과 가변성 분석은 모델로부터 컴포넌트를 추출하는

핵심 활동이다 [7]. 컴포넌트는 이러한 가변성을 구현해야 하고 가변성을 해결하기 위해 커스터마이즈하기 위한 메커니즘을 제공한다.

컴포넌트는 참조 모델을 준수한다. 컴포넌트 구현 환경, 예를 들어 컴포넌트 플랫폼은 특정한 컴포넌트 참조 모델을 구현한다. EJB 플랫폼에서 개발된 컴포넌트는 특정한 해석기나 게이트웨이를 사용하지 않는다면 .NET에서 개발된 컴포넌트와 상호운용하기 어렵다. 그래서 컴포넌트 소비자는 목표 컴포넌트의 컴포넌트 참조 모델을 고려해야 하고, 얼마나 잘 참조모델을 준수하는지를 고려해야 한다.

4. S/W 컴포넌트를 위한 품질 평가 모델

소프트웨어 컴포넌트의 특징에 기반하여, 품질 평가 모델을 정의한다. 이러한 품질 평가 모델은 특성, 부특성, 메트릭으로 구성된다. 품질 평가 모델은 타겟 시스템을 위한 다양한 관점으로 정의된다. 즉, 품질 모델은 타겟 시스템의 유일한 특징으로부터 도출된다.

본 논문에서는, 3장에서 식별된 특징으로부터 소프트웨어 컴포넌트의 품질 평가 모델을 도출한다. 본 논문에서는 그림 1과 같은 품질 평가 모델의 5개의 특성을 도출한다.

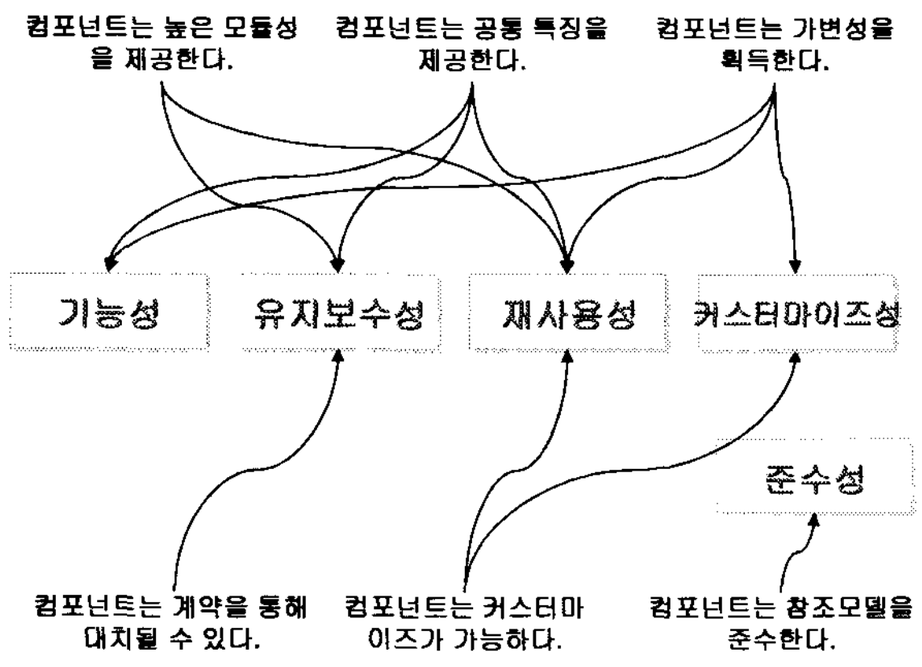

그림 1. 특징과 특성의 연관관계

기능성 (Functionality): 이 특성은 컴포넌트가 얼마나 내포된 요구사항인 기능과 비기능을 일정하게 충족하는지를 측정하는 것이다. 게다가 소프트웨어 컴포넌트는 도메인 특정한 컴포넌트이다. 그러므로, 이러한 특징은 모든 특성 중에서 가장 우선시 된다. 컴포넌트의 경우, 기능과 비기능 요구사항은 패밀리 요구사항 명세서(Family Requirement Specification)에서 패밀리 멤버가 공통으로 가지는 기능의 집합으로써 정의된다.

유지보수성 (Maintainability): 컴포넌트가 높은 모듈성, 공통 특징, 계약을 제공한다는 특징은 유지보수성에 영향을 준다. 소프트웨어 유지보수는 수정, 향상, 새로운 기능의 추가와 예방적인 유지보수를 포함한다. 유지보수성은 얼마나 효과적으로 유지보수 활동이 소프트웨어 컴포넌트에서 수행될 수 있는 지를 측정한다.

재사용성 (Reusability): 소프트웨어 컴포넌트는 특정 도메인을 타겟으로 하기 때문에 동일한 도메인 안에서 높은 재사용을 제공해야 한다. 공통성과 가변성 분석으로부터 가변성을 실현한 컴포넌트의 require 인터페이스는 높은 재사용성을 제공한다. require 인터페이스와 커스터마이제이션을 통해 컴포넌트를 재사용할 수 있기 때문에 조직에서 소프트웨어 컴포넌트가 사용되는 범위가 넓다. 컴포넌트의 높은 모듈성과 재사용 단위가 크다는 특징은 객체 지향 재사용보다 더 높은 재사용성을 생산한다.

커스터마이즈 가능성 (Customizability): 커스터마이제이션은 컴포넌트 소비자의 목적에 맞게 컴포넌트를 수정하는 것이다. 가변성의 획득은 커스터마이제이션을 야기한다. 컴포넌트는 패밀리 멤버를 위한 공통 특징을 제공하고 부가적으로 각 패밀리 멤버를 위한 가변성을 제공한다. 그러므로, 각 패밀리 멤버 안에서 컴포넌트를 사용하기 위해서는 커스터마이제이션은 필수적이어야 한다. 선

택과 플러그-인 메커니즘은 수정을 지원하기 위한 커스터마이제이션 메커니즘이다.

준수성 (Conformance): 컴포넌트는 컴포넌트 참조 모델을 구현할 때에 컴포넌트 플랫폼을 생산한다. 그래서 컴포넌트는 컴포넌트 참조 모델과 컴포넌트 플랫폼에 의존적이다. 준수성은 컴포넌트가 얼마나 잘 선택된 참조 모델을 준수하는지를 측정할 것이다. 참조 모델을 준수하지 않는 컴포넌트는 쉽게 다른 컴포넌트와 상호운영되지 않고 재사용성이 낮을 것이다.

품질 평가 모델의 특성이 정의되면 품질 속성의 하위 수준인 부특성이 정의되어야 한다. 소프트웨어 컴포넌트의 특성을 도출하기 위해서 우선적으로 ISO/IEC 9126으로부터 적용하고 커스터마이즈하였으며 소프트웨어 컴포넌트의 특징을 반영하여 새로운 특성인 맞춤성을 추가하였다. 이와 같이 도출된 특성의 부특성을 위해 그림 2와 같은 소프트웨어 컴포넌트의 품질 속성에 대한 계층도를 제안한다.

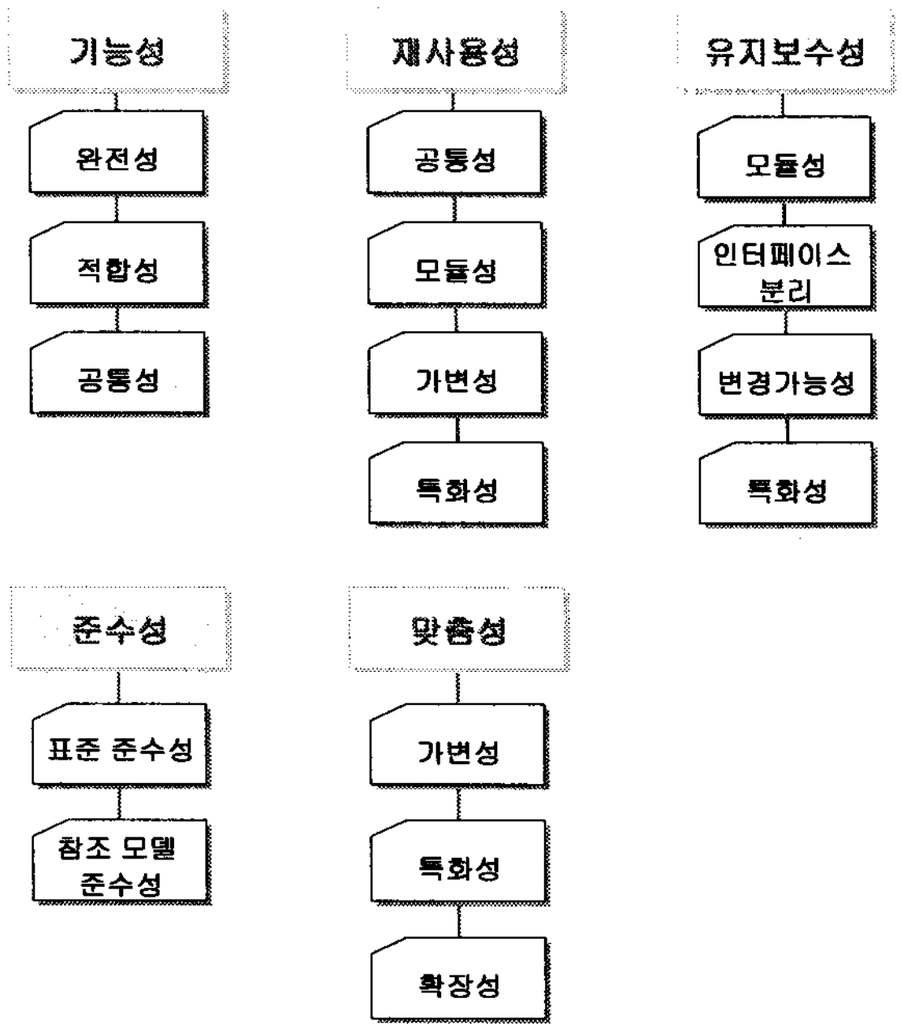

그림 2. 소프트웨어 품질 속성의 계층도

5. 소프트웨어 컴포넌트의 메트릭

본 장에서는 이전 장에서 도출한 부특성을 측정하기 위한 대표적인 메트릭을 제안한다. 각 특성을 대표하는 부특성을 기반하여 메트릭을 제안한다.

5.1. 공통성을 위한 메트릭

컴포넌트의 공통성을 측정할 때는 기능과 비기능적 측면을 함께 고려해야 한다. 기능 공통성을 측정하기 위해 *Functional Coverage (FC)* 메트릭을 비기능 공통성을 측정하기 위해 *Non-functional Coverage (NC)* 메트릭을 정의한다.

FC는 컴포넌트 안에 공통적인 처의 수를 측정하는 것이다. 휘처의 공통성은 패밀리 멤버가 기능 휘처를 공유하는 정도를 계산함으로써 측정할 수 있다.

FC = *(공통적인 기능 휘처들의 수) /*
(전체 기능 휘처들의 수)

이러한 휘처의 수는 패밀리 요구사항 명세서(FRS)로부터 식별된다.

FC의 범위는 0..1이다. FC값이 낮으면 기능 공통성이 낮은 것을 가리키며, 도메인 내의 모든 패밀리 멤버에 적용된 경우는 1을 가리킨다. 메트릭 값이 높으면, 컴포넌트가 적용 가능한 범위가 넓다는 것을 의미한다.

NC 메트릭은 컴포넌트 내부의 비기능적 휘처의 정도를 측정한다. 이것은 FC 메트릭과 비슷하며 다음과 같이 계산된다.

NC = *(공통적인 비기능 휘처들의 수) /*
(전체 비기능 휘처들의 수)

비기능적 휘처는 관련된 도메인과 프로젝트의 제약사항을 고려하여 품질 평가자가 판단한다.

NC의 범위는 0..1이다. NC의 값이 높으면 더 높은 비기능 공통성을 가리키며, NC의 값이 1이면 모든 비기능 휘처가 패밀리 멤버안에 적용된 것을 의미한다.

마지막으로, 이 두 개의 메트릭으로부터 공통성 커버리지인 *Commonality Coverage (CC)*의 값을 계산할 수 있다.

$$CC = W_{FC} \bullet FC + W_{NC} \bullet NC$$

여기에서 W_{FC} 와 W_{NC} 는 각 메트릭을 위한 가중치이며 합은 1이다. 각 가중치의 값은 도메인내에서의 우선순위에 따른 비율을 고려하여 정한다.

CC의 범위는 0..1이다. CC의 값이 1이면 모든 기능과 비기능 휘처가 패밀리 멤버에서 공통적으로 적용된 것을 의미한다.

5.2 모듈성을 위한 메트릭

모듈성을 측정하기 위해 *Component Modularity (CM)*를 정의한다. 이 메트릭은 컴포넌트의 독립의 정도를 측정하며, 이것은 기능의 응집성의 정도이다.

CM = *(다른 컴포넌트의 기능을 발생시키는 것 없이 이 컴포넌트 자체에 의해 제공되는 기능성의 범위)* /
(다른 컴포넌트를 발생하기 위한 필요에 상관없이 컴포넌트에 의해 제공되는 기능의 전체 범위)

CM의 범위는 0..1이고, CM의 값이 1이면 컴포넌트는 자신만을 포함한다는 것을 의미한다. 그러나 대부분의 컴포넌트는 발생과 위임의 형태로 다른 컴포넌트와 어느 정도 상호작용하기 때문에 1의 값을 가지지는 않는다..

5.3. 인터페이스 분리를 위한 메트릭

인터페이스 분리를 측정하기 위해 *Interface Partition (IP)* 메트릭을 정의한다. 이 메트릭은 인터페이스를 통한 컴포넌트의 내부 로직이나 워크플로우를 표현하는 정도를 측정한다. 이것은 다음과 같이 계산된다.

IP = *(인터페이스의 영향 없이 수정된 로직이나 워크플로우의 수)* /
(내부 컴포넌트안에 수정 가능한 로직이나 워크플로우의 전체 수)

여기에서, 유지보수를 위한 수정은 컴포넌트 내부에서 발생한다.

IP의 범위는 0..1이고. IP의 값이 높으면 컴포넌트의 수정이 인터페이스의 영향을 주지 않고 일어난 것을 의미한다.

5.4. 표준 준수성을 위한 메트릭

표준 준수성을 측정하기 위해 *Standard Conformance (SC)* 메트릭을 정의한다. 이 메트릭은 관련 있는 표준에 대한 준수성의 정도를 측정한다. 이것은 다음과 같이 계산된다.

SC = *(컴포넌트의 구현 시에 엄중히 따르는 표준 요소의 수)* /
(연관된 표준의 집합 안에서 명시된 표준 요소의 전체 수)

여기에서, 표준 요소는 정부의 법령, 실제로 존재하는 표준, 도메인 내에서의 표준 정책 등이 될 수 있다.

SC의 범위는 0..1이다. SC의 값이 1이면컴포넌트가 모든 관련된 표준 요소를 준수한다는 것을 의미한다.

5.5. 참조 모델 준수성을 위한 메트릭

참조모델 준수성을 측정하기 위해 *Reference-model Conformance (RC)* 메트릭을 정의한다. 이 메트릭은 적용된 컴포넌트 참조 모델의 준수성의 정도를 측정한다. 이 메트릭은 다음과 같이 계산된다.

RC = *(컴포넌트의 구현 내에서 엄격히 준수하는 구문과 의미론적 요소의 수)* /
(컴포넌트 참조 모델 안에서 명시된 구문과 의미론적 요소의 수)

여기에서 요소는 인터페이스와 클래스의 명, 규칙, 제약사항, 정책, 요구사항 등이 될 수 있다.

RC의 범위는 0..1이고, RC의 값이 1이면 컴포넌트가 컴포넌트 참조 모델의 모든 구문과 의미론

적 요소를 준수한다는 것을 의미한다.

5.6. 가변성을 위한 메트릭

가변성을 측정하기 위해 *Coverage of Variability (CV)* 메트릭을 정의한다. CV 메트릭은 FRS에서 식별된 가변성이 얼마나 많은지를 측정한다. 이것은 다음과 같이 계산된다.

CV = *(FRS안에 포함된 가변점의 수) /*
(명시적이고 잠재적인 가변점의 수)

여기에서 몇몇 식별된 가변점은 경제적인 가치의 관점에서 고려되어야 한다.

CV의 범위는 0..1이고 CV의 값이 높으면 컴포넌트가 가변점의 수를 충분히 고려했다는 것을 의미한다. 이것은 컴포넌트가 다양한 어플리케이션에서 이용 가능하다는 것을 의미한다.

5.7. 특화성을 위한 메트릭

특화성은 얼마나 많은 가변점이 효과적으로 특화될수 있는 지를 측정한다. 여기서 유효하다는 것의 의미는 부작용(Side Effect)이나 결점(Fault) 없이 가변점이 가변치에 의해 바운드되는 것을 의미한다. 특화성을 측정하기 위해 *Variability Tailoring (VT)* 메트릭을 정의한다. 이 메트릭은 얼마나 많은 가변점이 효과적이고 정확하게 해결되는지를 측정한다. 이것은 다음과 같이 계산된다.

VT = *(효과적으로 해결할 수 있는 가변점의 수) /*
(전체 가변점의 수)

여기에서 가변점은 제약사항을 고려하여 각각의 가변점을 해결할 수 있는 메커니즘을 효과적이고 적절하게 정의한 가변점들을 의미한다.

그러나 이러한 가변점을 효과적으로 해결할 수 있는지를 정량적으로 측정하기 어렵기 때문에 체크리스트를 통한 품질 평가자의 판단이 필요하다. 예를 들어, 품질 평가자는 Open 가변성을 위한 Plug-in 메커니즘이 객체 전달을 하기에 적절하게 정의되어 있는지 조사해야 한다 [8].

VT의 범위는 0..1이고, VT의 값이 높으면 컴포넌트가 효과적으로 가변점을 해결했다는 것을 의미한다.

6. 실제 소프트웨어 컴포넌트로의 적용

제안된 품질 평가 모델을 사용하여, 소프트웨어 컴포넌트의 전반적인 품질을 다음과 같이 계산할 수 있다.

C_i는 품질 평가 모델에서 정의된 i번째 특성이고 i의 범위는 $1 \le i \le 5$이다. S_{ij}는 C_i의 j번째 부특성을 지칭한다. 예를 들어, S_{32}는 그림 2에 따라 유지보수성의 인터페이스 분리이다. 소프트웨어 컴포넌트의 목적에 따라 각 특성의 값을 계산하기 위해 다른 가중치를 부특성에 할당할 수 있다.

$$QC_i = \sum_{j=1}^{n} W_{ij} \cdot S_{ij}$$

소프트웨어 컴포넌트의 목적에 따라 각 특성의 값을 계산하기 위해 다른 가중치를 부특성에 할당할 수 있다.

그리고 각 특성에 연관된 가중치는 적용된 컨텍스트(context)에 따라 달라질 수 있다. 소프트웨어 컴포넌트의 품질을 평가한 최종값인 Q는 다음과 같이 계산할 수 있다.

$$Q = \sum_{i=1}^{5} W_i \cdot QC_i$$

Q의 값이 높으면 컴포넌트의 품질이 우수하다는 것을 의미한다.

소프트웨어 컴포넌트를 위한 제안된 품질 평가 모델을 효과적으로 적용하기 위해서는 품질 평가 프로세스, 지침, 템플릿의 집합이 정의되어야 한다.

7. 결론

소프트웨어 컴포넌트는 다양한 패밀리 멤버에서 사용하기 위한 비즈니스 개념과 프로세스를 구현

한 컴포넌트이다. 그래서, 소프트웨어 컴포넌트의 품질을 평가하는 것은 성공적인 컴포넌트 기반 시스템 개발을 위해 중요한 선행작업이다.

본 논문에서는 소프트웨어 컴포넌트의 주요 특징을 식별하였고, 소프트웨어 컴포넌트의 품질을 평가하기 위한 품질 평가 모델을 도출하였다. 본 논문에서는 5개의 특성, 16개의 부특성을 도출하였고 대표적인 메트릭을 정의하였다. 제안된 품질 평가 모델이 소프트웨어 컴포넌트를 사용한 컴포넌트 기반 시스템 개발을 위해 기본적인 프레임워크로 제공될 수 있다고 판단된다.

[참고문헌]

[1] Building Reliable Component-Based Software Systems, Crnkovic, I. and Larsson, M., Artech House, Inc., 2002.

[2] Component–Based Software Engineering, Heineman, G. T. and Councill, W. T., Addison-Wesley, 2001.

[3] Software Engineering—Product Quality—Part 1: Quality Model. ISO/IEC 9126-1, June, 2001.

[4] Software Quality Management, McCall, J. A., A Petrocelli Book, 1979.

[5] Characteristics of Software Quality, Boehm, B. W., Brown, J. R., Lipow, H., MacLeod, G. J. and Merrit, M. J., Elsevier North Holland, 1978.

[6] Objects, Components, and Frameworks with UML, D'Souza, D. F. and Wills, A. C., Addison Wesley Longman, Inc., 1999.

[7] Component-based Product Line Engineering with UML, Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wust, J., and Zettel, J., Pearson Education Ltd, 2002.

[8] A Theoretical Foundation of Variability in Component-Based Development, S. Kim, J. Her, and S. Chang, Information and Software Technology, Vol.47, pp.663-673, 2005.