패싯 분류를 이용한 객체지향 컴포넌트의 계층구조화

한정수*, 김귀정**
*천안대학교 정보통신학부, **건양대학교 컴퓨터학과
e-mail:jshan@cheonan.ac.kr

A Hierarchy of Object-Oriented Component Using Facet Classification

Jung-Soo Han*, Gui-Jug Kim**
*Division of Info. & Comm. Cheonan University
**Dept of Computer Science, KonYang University

요 약

본 연구는 재사용이 가능한 객체지향 컴포넌트의 효율적인 검색을 위해 클래스 개념 범주(CCC)를 정의하고 클래스의 상속 관계를 이용한 CCC 상속을 제안하였다. CCC의 상속은 클래스 간 상속관계를 인지한 후, 하위 클래스는 상위 클래스의 모든 CCC를 자동 상속받게 되며, 클래스 구문분석에 의해 객체지향 코드로부터 자동 인식된다. 본 연구는 모든 용어에 대한 범주를 수동으로 할당해주는 기존 방법의 단점을 극복할 수 있으며, 시소러스를 자동으로 갱신할 수 있다는 장점이 있다.

1. 서론

소프트웨어 재사용성을 높이기 위한 다양한 컴포넌트 기반의 개발 방법론이 제안되고 있으며, 현재 한국컴포넌트컨소시엄등에서 상호호환을 위한 컴포넌트 개발에 관한 연구가 진행되고 있다. 이처럼 많은 컴포넌트 중 사용자가 원하는 컴포넌트를 식별하는 방법은 분석자의 경험에 의존하는 경우가 많으며, 특히 여러 카테고리에 분류된 컴포넌트의 경우에는 대부분 개발자의 경험에 의해 검색이 이루어지고 있다. 이 방법은 극히 주관적이고, 비효율적이며, 시스템의 일관성을 유지하기도 어렵기 때문에 자동화된 컴포넌트의 검색 방법이 무엇보다 필요하다.

이에 본 연구에서는 재사용이 가능한 객체지향 컴포넌트의 효율적인 검색에 목적을 둔다. 이를 위하여 컴포넌트를 구성하는 클래스를 여러 경험적 상황에 기반을 둔 패싯 항목으로 구분하였다. 이 항목을 클래스 개념 범주(CCC)로 정의하여 클래스의 기본적인 기능과 행위 그리고 클래스를 구현하고 실행할 수 있는 실제 상황을 기술할 수 있도록 하였다. 또한 본 연구에서는 클래스의 상속 관계를 이용한 CCC 상속을 제안하였다. 상속을 받은 하위 클래스는 모든 상위 클래스의 CCC를 상속받을 수 있도록 하였다. CCC의 상속은 객체지향 코드로부터 자동 인식되며 클래스 구문분석에 의해 특정 키워드에 의한 클래스 간 상속관계를 인지한 후, 하위 클래스는 상위 클래스의 모든 CCC를 자동 상속받게 된다. 이는 시소러스를 자동으로 갱신할 수 있으며, 사용자 수작업의 부담을 최대한 감소시켜 컴포넌트의 재사용성을 높일 수 있다는 장점이 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 컴포넌트 분류와 검색에 대한 관련연구를 설명하고, 3장에서는 클래스의 개념적인 부류에 대해 설명한다. 4장에서는 검색방법에 대해 설명하고, 끝으로 5장에 실험결과를 살펴보고 결론을 맺는다.

2. 관련 연구

컴포넌트를 분류하기 위해서 널리 사용되는 방법으로는 열거형 분류방법과 패싯 분류방법이 있다[1]. 열거형 분류방법은 예상되는 모든 컴포넌트를 클래스로 분할하면서 계층적 관계로 정의한 후 각 컴포넌트들을 계층 구조상의 클래스에 사용자가 직접 할당하는 방법이다. 이 방법은 컴포넌트를 신속하게

찾을 수는 있으나 새로운 컴포넌트를 추가할 경우 분류계층을 다시 구성해야 하는 단점이 있다. 패싯 분류 방법은 열거형 방법의 단점을 개선하기 위하여 컴포넌트들이 갖는 공통적인 특성을 합성하여 하나의 패싯으로 표현하며 하나의 컴포넌트는 여러 개의 패싯으로 나눌 수 있다. 이 방법은 분류가 간단하고 확장이 용이한 장점이 있지만, 항목이 많아지면 관련성의 명세와 동의어 처리가 어려워진다.

기존의 컴포넌트 검색 연구는 시그니쳐 일치 검색[2], 행위 샘플링에 의한 검색[3], 명세서 일치에 의한 검색[4] 등이 있다. 시그니쳐 일치 검색은 함수의 파라메터 타입이나 인터페이스와 같은 시그니쳐 정보를 이용하여 컴포넌트를 검색하는 방법이며, 행위 샘플링 검색은 인터페이스 명세서와 호환되는 인터페이스를 가진 저장소의 루틴을 실행하여 그 결과를 비교함으로써 검색하는 방법이다. 명세서 일치 방법은 컴포넌트 명세서를 비교하여 다른 컴포넌트와 대체될 수 있는지를 비교하여 결정한다. SARM(Spreading Activation Retrieval Method)[5]은 컴포넌트와 질의어 사이에 질의어 기능을 포함하는 유사한 컴포넌트들을 검색하여 보다 더 정확하고 넓은 범위의 컴포넌트들을 찾을 수 있는 방법이다. SARM은 직접 인덱싱되지 않은 컴포넌트들까지 검색할 수 있는 효율적인 검색 방법이며 정보저장소에 컴포넌트들을 구축할 때 각 항목을 일일이 인덱싱하지 않아도 되기 때문에 많은 비용이 절감된다. 그러나 활성값을 이용한 반복 계산으로 유사도를 측정하였기 때문에 검색시간을 지연시키는 단점이 있다.

3. 클래스의 개념적 분류

3.1 클래스 개념 범주 생성

본 연구에서는 클래스 분류를 위해 클래스 개념 범주(Class Concept Category : CCC)를 생성하였다. 클래스가 사용될 수 있는 여러 경험적 상황을 패싯 항목으로 설정하는 패싯 분류방법을 사용하였다[1]. 패싯은 클래스의 기본적인 기능과 행위 그리고 클래스를 구현하고 실행할 수 있는 실제 상황을 기술할 수 있도록 「Component Type」, 「Class Type」, 「Using Scope」, 그리고 「User Interaction Style」의 4개 패싯을 포함하도록 정의하였다. 표 1은 본 연구에서 정의한 클래스에 대한 패싯 항목(CCC)을 나타낸 것이며, 이는 마이크로소프트 윈도우 애플리케이션 개발을 위한 컴포넌트의 클래스들을 위한 패싯 분류이다. 하나의 컴포넌트는 여러 개의 CCC에 분류될 수 있으며, CCC는 도메인 전문가에 의해 자체적으로 확장되거나 추가될 수 있다. 「Component Type」은 각 클래스의 적용에 대한 컴포넌트 범위로서 클래스가 포함된 컴포넌트가 파일 시스템이나 메모리 관리, 런타임(File System and Interprocess Communication Service)등을 위해서 사용되는지 또는 사용자 인터페이스 구현(User Interface Development)을 위한 부분에 사용되는지를 나타낸다. 「Class Type」은 마이크로소프트 윈도우 애플리케이션 개발을 위한 클래스 라이브러리 분류를 바탕으로 클래스 타입을 5개의 CCC로 분류하였다. “Window Support”는 윈도우와 그래픽 관련 클래스를 포함하고, “Data Structure and File”은 자료 구조와 파일 및 데이터베이스 클래스를 포함하며, “Internet and Network”는 인터넷 관련 클래스를 말한다. 또한 “Exception and Debugging”의 클래스는 에러 처리와 디버깅을 담당하며, “OLE”는 OLE 관련 클래스를 포함한다. 「Using Scope」는 클래스의 역할을 분류한 것으로 “Frame Architecture”는 윈도우 프로그램이 공통적으로 수행하는 기능들을 구현해 놓은 클래스들이 포함되어 있으며, “Application Processing”은 응용 도메인에 종속적인 클래스가 포함된다. 「User Interaction Style」은 사용자와의 상호작용 방법에 따른 분류로써 “Graphic and Drawing Support”와 “Interactive and Dialog”로 구분되는데, 전자에는 그래픽처리 클래스가 포함되고 후자에는 이벤트와 윈도우, 다이얼로그 클래스들이 포함된다.

표 1. 패싯 분류에 의한 CCC

패싯	패싯 항목(CCC)
Component Type	· File System and Interprocess Communication Service · User Interface Development
Class Type	· Window Support · Data Structure and File · Internet and Network · Exception and Debugging · OLE
Using Scope	· Frame Architecture · Application Processing
User Interaction Style	· Graphic and Drawing Support · Interactive and Dialog

위와 같이 초기 설정된 CCC는 고정되어 있지 않으며 필요에 따라 증가될 수 있다. 이렇게 분류된 CCC는 객체지향 시소러스를 구축하기 위해 CCC와 클래스 사이의 관련 값을 계산하게 된다[1]. 이 방법으로 구축된 객체지향 시소러스는 기능적으로 유사한 클래스뿐 아니라 연관된 클래스까지도 질의 확장에 의해 선택되어 질 수 있도록 해준다. 또한 계산

과정을 단순화시키는 역할을 하여 시소러스 구축을 단순화시키는 동시에 업데이트를 용이하게 해준다.

3.2 자동 CCC 상속

본 연구에서는 객체지향 컴포넌트의 특성에 부합하는 클래스 범주를 분류하고 클래스의 상속 관계를 이용한 CCC 상속을 제안하였다. 한 클래스를 상속받은 하위 클래스는 상위 클래스와 같은 범주에 분류될 수도 있고, 다른 범주에 분류될 수도 있다. 그러나 다른 범주에 분류된다 할지라도 하위 클래스는 상위 클래스의 특성을 포함하고 있기 때문에 상위 클래스가 가지고 있는 CCC가 하위 클래스로 상속될 수 있도록 하였다. 또한 다중 상속을 받은 하위 클래스인 경우에는 모든 상위 클래스의 CCC를 상속받는다. 그러나 상속이 거듭될수록 상위 클래스들이 포함된 CCC의 특성은 일정하게 유지되지 못할 것이므로 CCC의 관련정도를 상속레벨에 따라 감소시켜야 할 것이다[6]. 따라서 본 연구에서는 상속레벨 h에 따라 CCC 관련정도를 $(1/2)^h$만큼 감소시켰다. 그림 1은 클래스 상속에 따른 CCC 상속과정을 보여준다. 'Command_Target' 클래스는 'Document', 'Window_Frame', 그리고 'Command_UI' 클래스의 상위 클래스로써, 'Window Support'와 'Frame Architecture' CCC에 포함된다. 'Document' 클래스는 자체적으로 'Data Structure and File' CCC에 포함되며, 'Command_Target' 클래스에서 상속되었기 때문에 'Window Support'와 'Frame Architecture' CCC를 상속받을 수 있다. 이때, CCC 관계값을 계산할 때, CCC별 클래스 발생횟수를 상속레벨 h에 따라 $(1/2)^h$로 계산한다. 즉, 'Document' 클래스에 대한 3개의 CCC별 발생횟수가 'Data Structure and File'은 1, 'Window Support'는 $(1/2)^h$, 그리고 'Frame Architecture'는 $(1/2)^1$이 된다. 같은 방법으로, 'OLE Document' 클래스의 경우에는 'OLE'는 1, 'Data Structure and File'은 $(1/2)^1$, 'Window Support'는 $(1/2)^2$, 그리고 'Frame Architecture'는 $(1/2)^2$의 CCC별 발생횟수를 가지게 된다.

CCC의 상속은 객체지향 코드로부터 자동 인식되는데 클래스 구문분석에 의해 특정 키워드에 의한 클래스 간 상속관계를 인지한 후, 하위 클래스는 상위 클래스의 모든 CCC를 자동 상속받게 된다. 이는 모든 용어에 대한 범주를 수동으로 할당해주고 관계를 설정해야 하는 기존의 시소러스 방법의 단점을 극복할 수 있으며, 기존의 클래스를 상속받은 새로운 클래스를 시소러스에 추가시키고자 할 때, 소스코드만을 사용하여 구문분석에 의해 시소러스를 자동으로 갱신할 수 있다는 장점이 있다.

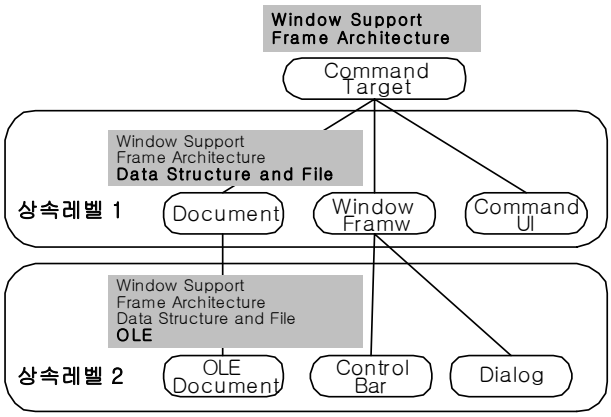

그림 1. CCC 상속

4. 컴포넌트 검색

CCC에 의해 다중 패싯분류된 컴포넌트의 검색을 위해서는 쿼리가 표현하고자하는 CCC가 어떤 것인가를 찾아야 하고, 또한 그 쿼리가 여러 개의 CCC를 만족할 수 있음을 이해해야 한다. 이에 본 연구에서는 쿼리가 클래스 형태로 주워졌을 때, 이를 만족하는 CCC를 찾기 위하여 스프레딩 엑티베이션방법을 이용하였다[5]. 이를 위해 클래스와 CCC들간의 초기 활성값을 설정하였는데, 각 클래스의 활성값은 컴포넌트와 클래스간의 가중치에 의해 결정된다[5]. 각 클래스의 초기 활성값은 한 클래스에 대해서 계산된 가중치의 평균값이고, CCC의 초기 활성값은 클래스와 CCC 간의 연관성를 이용하여 계산된 클래스-CCC관계값이다. CCC를 검색하는 과정이 그림 2에 나타나 있다. 클래스 A의 초기 활성값은 0.8이고, CCC W의 초기 활성값은 0.5이다. 쿼리로 특성 「A」를 입력하면 3개의 컴포넌트가 검색된다. 여기서 클래스 「A」는 CCC 「W」와 「X」에 직접 연결되어 있지만 CCC 「Y」와 「Z」에는 연결되어 있지 않다. 그러나 「A」→「X」→「D」→「Z」를 통하여 연결되고, 「「A」→「X」→「D」→「Y」를 통하여 2개의 CCC(「Z」,「Y」)가 연결됨을 알 수 있다. 하지만 「Y」는 검색과정에서 적게 참조되므로 연결이 제거된다. 이처럼 각 클래스와 CCC는 서로 연결되어 있는 노드를 참조해 가면서 활성값을 계산하게 된다. 순환이 반복될수록 활성값은 안정되며 참조회수가 기준에 미달되는 부분은 자동으로 제거되어 계산과정이 종료된다.

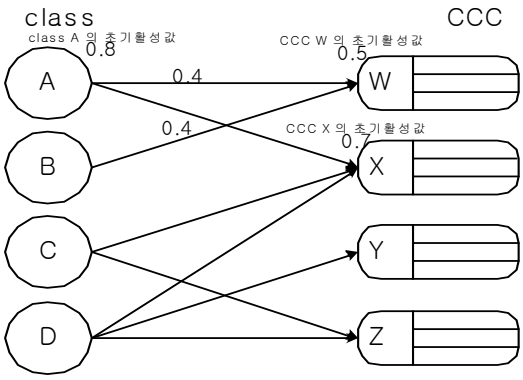

그림 2. 쿼리에 대한 CCC 검색 과정

CCC 검색 결과, 한 쿼리셋에서 공통적으로 나타나는 CCC를 모두 만족하는 후보컴포넌트에 대해서 쿼리셋과 컴포넌트들과의 신뢰도를 계산한다. 최종적인 신뢰도는 동치관계, 포함관계, 유사도를 계산함으로써 얻어진다[1]. 유사도가 계산된 후보 컴포넌트들은 유사도 순으로 우선순위에 따라 최종적으로 검색된다.

5. 실험결과 및 결론

본 연구에서는 컴포넌트의 특성에 부합하는 클래스 범주를 분류하고 클래스의 상속 관계를 이용한 CCC 상속을 제안하였다. 또한 가중치와 클래스-CCC관계값을 이용한 Spreading Activation 방법을 적용하여 CCC를 검색하였고 CCC를 모두 만족하는 후보컴포넌트에 대해서 유사도를 계산하여 우선순위에 따라 컴포넌트를 검색하였다. CCC상속관계의 효율성을 실험하기 위하여 재현율과 정확성을 측정하였다. 그림 3과 그림 4는 클래스를 CCC로 분류하여 계층구조로 표현하고 Spreading Activation 방법을 적용하여 검색했을 때와, 단지 패싯 분류하여 검색했을 때의 방법을 비교하였다. 제안한 방법이 정확도 면에서도 뒤지지 않으면서 재현율이 크게 향상되었음을 알 수 있다.

본 연구에서는 컴포넌트의 특성에 부합하는 클래스를 CCC로 분류하고 클래스의 상속 관계를 이용한 CCC 상속을 제안하였다. 클래스가 사용될 수 있는 여러 경험적 상황을 패싯 항목으로 설정하였으며 본 연구에서는 상속레벨 h에 따라 CCC 관련정도를 $(1/2)^h$만큼 감소시켰다. CCC의 상속은 클래스 간 상속관계를 인지한 후, 하위 클래스는 상위 클래스의 모든 CCC를 자동 상속받게 된다. 이는 모든 용어에 대한 범주를 수동으로 할당해주는 기존 방법의 단점을 극복할 수 있으며, 새로운 클래스를 시소러스에 추가시키고자 할 때, 시소러스를 자동으로 갱신할 수 있다는 장점이 있다.

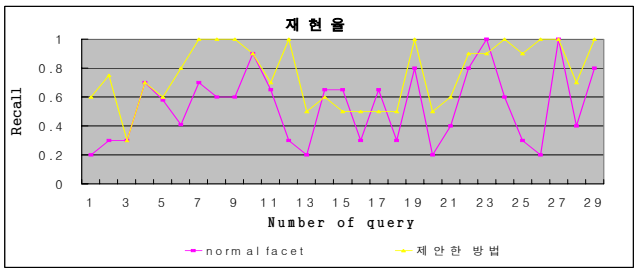

그림 3. 재현율

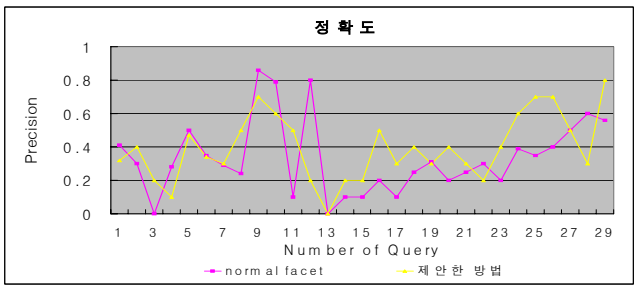

그림 4. 정확도

참고문헌

[1] 김귀정, 한정수, 송영재, "컴포넌트 검색을 지원하는 퍼지 기반 시소러스 구축," 한국정보처리학회 논문지, 제10-D권 제5호, pp. 753-762, 8. 2003.

[2] A. M. Zaremski, J. M. Wing, "Signature Matching: A Tool for Using Software Libraries," ACM Transaction Software Engineering and Methodology, Vol. 4, No. 2, 1995.

[3] A. Podgurski, L. Pierce, "Retrieving Reusable Software by Sampling Behavior," ACM Transaction Software Engineering and Methodology, Vol. 2, No. 3, 1993.

[4] A. M. Zaremski, J. M. Wing, "Specification Matching of Software Components," In Proceedings of the third ACM SIGSOFT symposium on the foundations of software engineering, 1995.

[5] Scott Henninger, "Information Access Tools for Software Reuse," System Software, pp. 231-247, 1995.

[6] E. Damini, M.G.Fugini, C. Bellettini, "A Hierarchy-Aware Approach to Faceted Classification of Object-Oriented Components", The ACM Transaction on Software Engineering and Methodology, Vol.8, No.4, Oct. 1999, 425-472.