XML 명세에 기반한 소프트웨어 컴포넌트 검색

°권대삼*, 이윤수**, 윤경섭***, 왕창종*
*g1981351@inhavision.inha.ac.kr, cjwangsc@inha.ac.kr, 인하대학교 전자계산공학과
**yslee@intra.ansant.ac.kr, 안산공과대학 전산정보과
***ksyoon@true.inhatc.ac.kr, 인하공업전문대학 컴퓨터정보과

Retrieval of Software Component based on XML Specification

°T. S. Kwon*, Y. S. Lee**, K, S, Yoon***,C. J. Wang*
*Dept. of Computer Science & Engineering, Inha University
**Dept of Computer Information, Ansan College of Technology
***Dept. of Computer and Information System, Inha Technical College

요 약

소프트웨어 컴포넌트의 재사용은 새로운 소프트웨어를 개발하기 위해 이미 개발되어진 컴포넌트나 적절하게 수정된 컴포넌트를 사용하는 것이다 따라서 컴포넌트 저장소에 저장되어 있는 컴포넌트를 효율적으로 검색할 수 있어야 하며, 검색된 컴포넌트를 적용하여 새로운 소프트웨어를 개발할 수 있어야 한다.

이 논문에서는 컴포넌트 저장소에 XML 기반으로 명세된 컴포넌트들의 검색 방법과 소프트웨어 아키텍처 재구성을 위한 구조 검색 방법을 제안한다. 제안한 검색 방법에서 시그니쳐 일치 방법은 컴포넌트 검색의 재현율을 향상시키며, 행위 일치 검색은 컴포넌트 검색의 정확성을 향상시킬 수 있다. 또한, 구조 검색 방법은 소프트웨어 아키텍처의 재구성을 위해 컴포넌트의 구조적인 관점에서 컴포넌트를 검색할 수 있다

1. 서론

소프트웨어 재사용(software reuse)은 개발되어 있는 컴포넌트(components)들을 새로운 소프트웨어 개발에 사용함으로써 소프트웨어 개발 기간을 단축시키고, 개발 비용의 절감과 소프트웨어의 생산성을 향상시킬 수 있는 효과를 얻을 수 있다[1] 따라서, 소프트웨어 컴포넌트를 재사용하기 위해서는 재사용될 수 있는 소프트웨어 컴포넌트들을 분류하여 컴포넌트 저장소에 저장하고, 개발자의 요구와 일치하는 재사용 가능한 컴포넌트들을 컴포넌트 저장소로부터 검색해야 한다[2, 3]

최근에 새로운 소프트웨어를 개발하기 위해 재사용 가능한 컴포넌트의 검색 방법에 관한 연구가 활발히 진행되고 있다[1] 대표적인 예가 Zaremski 와 Wing이 제안한 시그니쳐 일치(signature matching) 방법과 소프트웨어의 실행 속성을 기반으로 하여 컴포넌트를 검색하는 행위 샘플링(behavioral samapling) 검색 방법이 있다 시그니쳐 검색 방법은 컴포넌트 내에 존재하는 함수의 타입을 비교하여 재사용 가능한 컴포넌트를 검색한다[4] 이 방법은 다른 검색 방법에 비해 재현율(recall)은 높지만, 정확성(precision)이 떨어지는 단점이 있다[5]

행위 샘플링 검색 방법은 임의로 선택한 입력값을 이용하여 컴포넌트를 실행시켜 실행 결과와 임의의 출력값을 비교하여 컴포넌트를 검색하는 방법이다[6]. 이 방법은 컴포넌트를 직접 실행시켜야 하기 때문에 시간이 오래 걸리며, 오류 처리를 위한 방법이 요구되지만 검색의 정확성은 높다[7]

따라서, 이 연구에서는 XML(eXtensible Markup Language) 기반으로 명세된[8] 컴포넌트들을 시그니쳐 일치 방법과 행위 일치 방법을 이용하여 검색 한다 또한, 컴포넌트의 기능적인 면보다는 구조적인 면을 고려하여 소프트웨어 아키텍처를 구성 하는 구조 검색 방법을 제안한다. 제안한 시그니쳐 일치 방법은 컴포넌트 검색의 재현율을 높이고, 행위 일치 방법을 이용하여 정확성을 향상시킬 수 있다.

2. 관련 연구 고찰

2.1 시그니쳐 일치 검색

시그니쳐 일치 방법은 소프트웨어 컴포넌트로부터 쉽게 얻을 수 있는 시그니쳐 정보를 이용하여 컴포넌트 저장소로부터 재사용 가능한 소프트웨어 컴포넌트를 검색하기 위한 방법이다 시그니쳐 일치 방법의 가장 일반적인 형태는 다음과 같다[4]

SignatureMatch(q, M, C) = {c ∈ C· M(c, q)}

q는 질의(query)이고, M은 일치 술어(match predicate), C는 저장소

내에 컴포넌트들이며, 일치 술어에 따라 질의와 일치하는 컴포넌트들의 집합을 반환한다.

소프트웨어 컴포넌트들은 함수와 모듈들로 구성되기 때문에 시그니처 일치 방법에서는 함수 일치(function matching)와 모듈 일치 방법(module matching)을 제공하고 있다[5]

함수 일치 방법은 시그니처 정보로 함수의 타입에 대해 질의의 타입을 검사하는 방식으로 컴포넌트를 검색한다. 일반적으로 정확하게 일치하는 컴포넌트를 검색할 수 없기 때문에 질의와 유사한 컴포넌트를 검색하는 이완 일치(relaxed match) 방법을 제공한다

개발자가 추상 데이터 타입(abstract data type)에 대한 연산의 집합을 요구할 수 있기 때문에, 시그니처 일치 방법에서 이와 같은 요구사항을 지원하기 위해 모듈 일치 방법을 제공한다 모듈에 대한 시그니처 정보는 인터페이스이고, 함수 일치 방법과 유사하게 컴포넌트를 검색한다

시그니처 일치 방법은 개발자의 요구에 맞는 컴포넌트를 검색할 수 있으며, 검색 공간을 줄이기 위한 필터링 역할을 수행한다. 그러나 실제로 컴포넌트의 메소드 기능에 대한 검색이 아니기 때문에 행위 일치 검색에 비해 정확성이 떨어지는 단점이 있다

2.2 행위 샘플링에 의한 검색

행위 샘플링(behavior sampling) 메소드는 소프트웨어 저장소로부터 자동화된 재사용 컴포넌트를 검색하기 위해서 제안되었다 기본적인 행위 샘플링은 저장소 루틴을 검색자가 제공하는 연산 입력(operational input)의 샘플 상에 실행시키고, 검색자에 의해 제공된 출력과 루틴의 출력을 비교함으로써 연관된 루틴을 확인한다[6]

개발자는 루틴의 파라미터의 수, 타입, 모드(입력/출력)를 규정 한다 이와 같은 애트리뷰트들에 대한 개발자의 명세서를 대상 인터페이스 명세서(target-interface specification)라 하고, 대상 인터페이스 명세서와 호환되는 인터페이스를 가진 저장소 루틴을 후보 루틴(candidate routine)이라 한다

기본 행위 샘플링의 구현은 루틴에 대한 인터페이스 명세서를 대상 인터페이스 명세서와 비교하여 후보 루틴을 결정하고, 후보 루틴의 행위는 입력 샘플 상에 실행시킴으로써 평가된다 후보 루틴의 출력은 검색자가 제공한 출력과 일치하는 지를 검사하기 위해 테스트된다. 일치하는 컴포넌트는 소프트웨어 개발에 적용할 수 있다[7]

행위 샘플링의 정확성은 입력 샘플의 크기에 따라 증가되지만 샘플의 크기에 따라 비용과 시간이 증가되고, 효율성의 떨어지게 된다. 따라서 행위 샘플링의 정확성을 항상시키기 위해서 샘플의 크기를 최소화할 수 있는 방법이 고려되어야 한다 또한 행위 샘플링의 재현율을 향상시키기 위해 기본 행위 샘플링의 확장을 고려해야 한다

3. XML 명세에 기반한 컴포넌트 검색

이 장에서는 컴포넌트 검색을 위한 시그니처 일치 검색과 행위 일치 검색 방법, 구조 검색을 이용한 컴포넌트 검색 방법을 제안한다

3.1 검색 알고리즘

개발자가 새로운 소프트웨어를 개발하기 위해서 필요한 컴포넌트 검색을 위한 시그니처 일치 검색 방법과 행위 일치 방법을 제안하고, 소프트웨어의 기능성보다는 구조적인 컴포넌트를 필요로 할 때, 검색을 위한 구조 검색 방법을 제안한다

```
begin Match
  while(!endofComponentInterfaceRepository())
    compare numQueryparameter to numComponentparameter
    compare numQueryDirection to numComponentDirection
    if(Querytype == Methodtype or Querytype >= Methodtype
      or Querytype <= Methodtype)
      Behavioral match(component, userInputValue)
      if(userOutputValue == ResultValue)
        return Component
      end if
    end if
  end while
end Match
```

그림 1 검색 알고리즘

그림 1 은 이 논문에서 제안한 컴포넌트 검색의 전체적인 알고리즘을 보여주고 있다

3.2 시그니쳐 일치에 의한 검색

XML 기반으로 명세되어 있는 컴포넌트가 컴포넌트 명세서 저장소에 저장되어 있을 때, 시그니쳐 일치 방법을 이용하여 재사용 가능한 컴포넌트를 검색한다

시그니쳐 일치 방법은 컴포넌트 인터페이스에 존재하는 메소드의 파라미터 타입과 파라미터의 입출력 방향을 개발자의 질의와 비교하여 컴포넌트를 검색하는 방법이다 시그니쳐 일치 방법의 일반적인 일치 술어는 다음과 같다

$$M(L, Q) = (d_l, d_{ln}, t_l) \Leftrightarrow (d_q, d_{qn}, t_q)$$

$_l$은 컴포넌트 저장소 내에 컴포넌트 인터페이스의 파라미터 타입들이고, d_l 과 d_q 는 컴포넌트 인터페이스 내에 메소드와 질의의 파라미터 입출력 방향을 나타내고, d_{ln} 과 d_{qn} 은 파라미터 입출력 방향의 수를 나타낸다.

이완 일치 방법은 질의와 정확하게 일치하는 컴포넌트를 검색할 수 없을 때 질의와 유사한 컴포넌트를 검색하기 위한 방법이다

명세된 컴포넌트들을 질의와 비교할 때, 컴포넌트 인터페이스 내에 메소드의 파라미터 입출력 방향의 순서는 고려할 필요가 없다. 하지만, 입출력 방향의 수는 반드시 고려되어야 한다. 또한 입출력 방향의 수가 일치하더라도 입출력 방향의 타입이 질의와 다른 컴포넌트들의 저장소 내에 존재할 수 있기 때문에, 이와 같은 컴포넌트를 검색하여야 한다 따라서 질의나 컴포넌트 인터페이스 메소드의 파라미터 입출력 방향의 수가 일치하였을 때, 파라미터 타입만을 비교하여 컴포넌트를 검색한다

일반화 일치

일반화 일치(generalized match)의 일치 술어는 다음과 같다.

$$match_{gen}(L, Q) = (d_{ln} = d_{qn}) \wedge (L_{dt} \geq Q_{dt})$$

$d_{ln} = d_{qn}$ 는 컴포넌트 인터페이스의 파라미터 입출력 방향의 수와 질의의 입출력 방향의 수가 같아야 한다는 의미이다 $L_{dt} \geq Q_{dt}$ 는 컴포넌트 인터페이스의 파라미터의 타입이 질의의 입출력 방향의 타입 보다 일반적이다

특수화 일치

특수화 일치(specialized match)의 일치 술어는 다음과 같다

$$match_{spe}(L, Q) = (d_{ln} = d_{qn}) \wedge (L_{dt} \leq Q_{dt})$$

$d_{ln} = d_{qn}$ 는 컴포넌트 인터페이스의 파라미터 입출력 방향의 수와 질의의 입출력 방향의 수가 같아야 한다는 의미이다 $L_{dt} \leq Q_{dt}$ 는 질의의 입출력 방향의 타입이 컴포넌트 인터페이스의 파라미터의 타입보다 일반적이다는 의미이다 즉, 일반화 일치 방법의 역이다. 특수화 일치는 개발자가 새로운 소프트웨어를 개발할 때 필요한 컴포넌트 내에 존재하는 함수의 파라미터 타입을 알지 못할 경우에 사용할 수 있다

3.3 행위 일치에 의한 검색

행위 일치 검색을 수행하는 데 있어서 컴포넌트 저장소에 있는 모든 컴포넌트를 대상으로 하는 것은 비효율적이다 따라서 시그니쳐 일치 방법에 의해 검색된 컴포넌트들을 대상으로 하여 새로운 소프트웨어를 개발하기 위해 개발자의 요구에 맞는 컴포넌트를 검색하는 방법이다. 개발자는 컴포넌트를 실행시키기 위한 입력값과 컴포넌트 실행 결과와 비교하기 위한 출력값을 설정한다. 컴포넌트 실행 결과와 출력값의 비교하여 컴포넌트를 검색한다.

그림 는 행위 일치에 의한 검색 방법의 일반적인 검색 구조를 보여주고 있다

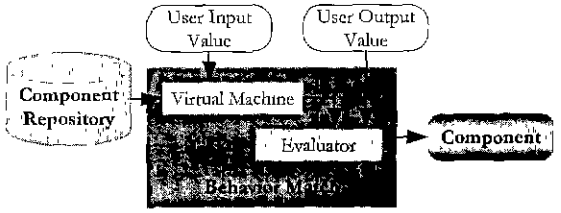

그림 2 행위 일치 검색

컴포넌트 저장소의 컴포넌트는 가상 머신 상에서 개발자의 입력값을 가지고 실행된다. 평가기는 가상 머신 상에서 실행된 컴포넌트의 결과와 개발자의 출력값을 비교하여 개발자의 요구 사항에 맞는 컴포넌트를 검색한다

3.4 구조 검색

구조 검색(architecture retrieval)은 개발자의 질의와 구조적으로 일치하는 패턴을 검색검색한다. 소프트웨어 구조 저장소에 저장된 구조들을 그림 3의 구조 검색 알고리즘을 이용하여 검색한다

```
step 1. Substitute all components for variable
        componet
step 2. Query architecutre-> Q
        Architecture Repository -> R
        Candidate architecture-> C[]
        i = 0;
        j = 0;
        FindFirstComponent(Q);
step 3. if (!EndOfRepository()) {
         ReadArchitecureRepository(R);
         FindFirstComponent(R)
        }
        else {
          go to step 8.
        }
step 4. if (Count(R.component) == Count( Q.component) )
          go to step 5.
        else
          go to step 3.
step 5. if (Q[i] == null)
          go to step 7;
        else {
          c1 = Count(Q.component[i].in);
          c2 = Count(Q.component[i].out);
        }
step 6. if ( c1 == count(R.in ) && c2 == count(R.out)) {
          i++;
          go to step 5;
        }
        else {
          go to step 3;
        }
step 7. C[j] = R;
        j++;
        go to step 3;
step 8. for( k = 0; k < j ; k++ ){
          flag = true;
          for( i = 0 ; i < Count( Q.component) ; i++ ) {
            if(!CompareType(Q.component[i].in,
                            C[k].component[i].in) ||
               !CompareType(Q.component[i].out,
                            C[k].component[i].out)) {
                 flag = fulse;
                 break;
            }
          }
          if ( flag == true) break;
        }
        if (flag == true) Select(C[k]);
```

그림 3 구조 검색 알고리즘

이 알고리즘의 1 단계에서 7 단계는 구조 내에 컴포넌트와 입출력 커넥터의 수를 비교하여 후보 구조를 추출하는 과정이다 마지막 8 단계에서 후보 구조 중 커넥터의 타입이 일치하는 구조를 검색한다

4. 결론 및 향후 연구 과제

본 연구에서는 XML 기반으로 명세된 컴포넌트들을 시그니쳐 일치 검색 방법을 이용하여 컴포넌트 저장소로부터 질의와 일치하는 컴포넌트를 검색하였다 또한 검색의 효율성을 향상시키기 위해 시그니쳐 일치 방법을 이용하여 필터링된 컴포넌트들을 대상으로 행위 일치 방법을 이용하여 소프트웨어 개발에 이용할 수 있는 컴포넌트를 검색하였다 제안한 방법을 이용하여 컴포넌트 검색의 재현율과 정확성을 향상시켰다 또한 컴포넌트의 구조적인 관점에서 설계 패턴을 검색하여 소프트웨어 아키텍쳐 재구성할 수 있도록 하였다

향후 연구로는 다른 컴포넌트 검색 방법과의 통합할 수 있는 방법에 관한 연구와 행위 일치 검색에서 오류 처리를 위한 방법의 연구가 필요하다. 또한, 제안한 구조 검색은 정확한 일치만 지원하기 때문에 연관 된 구조를 검색할 수 있는 방법에 관한 연구가 필요하다.

참고 문헌

[1] Charles W Krueger, "Software Reuse," *ACM Computing Surveys*, Vol. 24, No. 2, June 1992

[2] A Mili, R. Mili and R. Mittermeir, "A Survey of Software Components Storage and Retrieval," *The Institute for Software Research Technical Report*. 17, October, 1997

[3] J Penix, P Bamona and P Alexander, "Classification and retrieval of reusable components using semantic features" *Proceedings of 10th Knowledge-Based Software Engineering*, pages 131-138, 1995

[4] Amy Moormann Zaremski and Jeannette M Wing, "Signature Matching A Tool for Using Software Libraries," *ACM Transaction Software Engineering Methodology* 4, 2, 1995

[5] Amy Moormann Zaremski, "Signature and Specification Matching," *School of Computer Science Carnegie Mellon University Technical Report CS-CMU-96-103*, 1996

[6] A. Podgrudski and L. Pierce, "Behavior sampling A Technique for automated retrieval of reusable components," *Proceedings of 14th International Conference on Software Engineering*, 1992.

[7] R. J Hall, "Generalized behavior-based retrieval," *Proceedings of 15th International Conference on Software Engineering*, 1993

[8] 김원기, 안치돈, 이윤수, 왕칭종, "XML 기반의 컴포넌트 명세 언어," 한국정보과학회 추계 학술발표논문집, 1999