컴포넌트 기반 시스템 개발을 위한 요구사항과 컴포넌트 명세화 방안

박병철* 이재호** 박수용**
서강대학교 컴퓨터학과
**jhlee@selab.sogang.ac.kr, **sypark@ccs.sogang.ac.kr
데이콤ST
*nowforit@dacomst.com

Requirements and Component Specifications at for Component-based System Development

Byoungcheol Park*, Jaeho. Lee**, Sooyong Park**
*Dacom System Technologies co.
**Software Engineering Lab. Dept. of Computer Science & Engineering, Sogang University

요약

컴포넌트 기반의 소프트웨어가 개발되어짐에 따라 소프트웨어공학 측면에서 컴포넌트 기반의 소프트웨어 개발을 위한 컴포넌트 기반 소프트웨어공학(CBSE: Component Based Software Engineering)에 대한 연구가 진행되고 있다[1]. 컴포넌트 기반의 소프트웨어를 개발하는데 있어서 최대한 효율성과 생산성을 증가시키기 위해서는 개발초기, 즉 요구사항 분석단계에서 기존에 개발되어진 컴포넌트들 중에 사용자가 요구하는 역할을 수행하는 적절한 후보 컴포넌트를 추출할 수 있어야 한다. 따라서 본 논문에서는 XML의 특성을 활용한 요구사항과 컴포넌트 명세 언어를 정의하고 이를 통해 정형화된 요구사항으로부터 시스템 개발에 필요한 컴포넌트들을 추출하는 방안을 제시하고자 한다.

1. 서 론

컴포넌트 기반의 소프트웨어가 개발되어짐에 따라 소프트웨어공학 측면에서 컴포넌트 기반의 소프트웨어 개발을 위한 컴포넌트 기반 소프트웨어공학(CBSE: Component Based Software Engineering)에 대한 연구가 진행되고 있다[1].

컴포넌트 기반의 소프트웨어[3]를 개발하는데 있어서 최대한 효율성과 생산성을 증가시키기 위해 개발초기, 즉 요구사항 분석단계에서 기존에 개발되어진 컴포넌트들 중에서 사용자가 요구하는 역할을 수행하는 적절한 컴포넌트를 검색함으로서 전체 시스템의 아키텍쳐를 구성하는데 효율성을 증가시키고, 전체 개발비용을 절감하는 효과를 갖게 될 것이다. 따라서 본 논문에서는 컴포넌트 기반 시스템을 개발하는데 있어서 적합한 요구사항 명세와 컴포넌트 명세를 정의하여 이를 이용해 추출해 낸 후보 컴포넌트들을 시스템 개발 초기부터 적용(adaptation)시킴으로서 개발기간과 비용을 절감시키고 컴포넌트 시스템의 장점인 재사용성을 극대화할 수 있는 방안을 제시하고자한다.

2장에서는 관련연구로서 요구사항 명세화 방안, 컴포넌트 명세화 방안을 살펴보고, 3장에서는 컴포넌트 기반 소프트웨어 개발에 적합한 요구사항 명세화 방안과 컴포넌트 명세화 방안을 제시하고, 4장에서는 3장에서 제시된 각각의 명세화 방안을 토대로 하여 둘 간의 관계를 설정함으로서 요구사항 명세로부터 컴포넌트 후보를 추출할 수 있는 방안을 제시한다. 5장에서는 결론을 맺고자 한다.

2. 관련연구

2.1. 요구사항 명세

요구사항 명세는 분석된 요구사항을 명확하고, 정확하게 기록하여 문서화하는 것이다. 잘 정의된 요구사항명세서는 정확성(correct), 명확성(unambiguous), 완전성(complete), 입증가능성(verifiable), 일관성(consistent), 수정용이성(modifiable), 연계성(traceable) 등의 속성들을 갖추어야 한다[4].

요구사항을 명세하는데 있어서 자연어에 의한 방법과 정형화 기법(formal method)을 사용하는 방법이 있다. 하지만 이러한 요구사항의 정형화 기법들은 요구사항들간의 의미적 관계를 표현하는데 초점을 맞추고 있어서 구조적으로 정형화하기 힘들다는 단점이 있다.

2.2. 컴포넌트 명세화

컴포넌트 명세 언어(Component Specification Language)는 특정 문제에 대한 컴포넌트의 기능을 기술함으로써 컴포넌트 재사용성을 향상시키고 보다 쉽게 접근할 수 있는 방법을 제공한다. 컴포넌트 명세에 있어 기존에 이미 많은 명세 언어가 사용되고 있으며, 이들은 나름대로의 장점을 가지고 있다. 대표적으로 Rapide와 C2등을 예로 들 수 있으나 이들은 대부분 구현을 위해 특정 언어에 의존성이 강하며, 표현 능력에 한계를 가지고 있다. 이러한 한계를 해결하기 위해 문제 요구사항에 대한 설명, 컴포넌트의 기능, 그리고 컴포넌트 구조에 대한 명확한 정의를 제공하는 정형화된 명세 언어(Formal Specification Language)에 대한 연구가 현재 이루어지고 있다[2].

3. 요구사항과 컴포넌트의 명세화 방안

3.1. 요구사항 명세 언어

XML 기반의 요구사항 명세 언어를 정의하기 위해

유즈 케이스 명세(Use Case Specification)와 보완 명세(Supplementary Specification)에 필요한 항목들을 추출하고 구조화하여 요구사항 명세를 위해 필요한 항목들을 이용하여 DTD(Document Type Declaration)로 정의한다. 정의한 DTD에 의해서 XML 기반의 요구사항 명세 언어 구조를 얻을 수 있다.

전체적인 요구사항 명세는 [그림 1]과 같이 하나의 Supplementary Specification과 다수의 Use Case 명세로 구성된다.

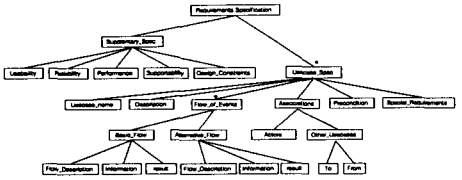

그림 1 요구사항 명세를 위한 항목들의 전체 트리구조

[그림 1]의 구조를 통한 DTD를 기본으로 한 요구사항 명세 언어 기본구조는 [그림 2]과 같은 형태를 가진다. 기존 명세 언어의 복잡성을 피하고 XML의 특징인 구조적인 데이터 표현을 명세 언어 기술에 활용하였다.

```
<?xml version="1.0"?>
<requirement_architecture>
  <supplementary>
    <usability> </usability>
    <reliability> </reliability>
    <performance> </performance>
    <supportability> </supportability>
    <design_constraints> </design_constraints>
  </supplementary>
  <usecases>
    <usecase_name> </usecase_name>
    <description> </description>
    <flow_of_events>
      <basic_flow>
        <flow_description> </flow_description>
        <information> </information>
        <result> </result>
      </basic_flow>
      <alternative_flow>
        <flow_description> </flow_description>
        <information> </information>
        <result> </result>
      </alternative_flow>
    </flow_of_events>
    <association>
      <actors> </actors>
      <other usecase> <to> </to>
        <from> </from>
      </other_usecases>
    </association>
    <precondition> </precondition>
    <special_requirements> </special_requirements>
  </usecases>
</requirement_architecture>
```

그림 2 XML기반 요구사항 명세 언어 구조

3.2. 컴포넌트 명세(Component Specification)

3.2.1. 컴포넌트 명세

컴포넌트 명세는 기존에 개발되어진 컴포넌트를 재사용 하는데 필수적인 자료가 되고, 또한 이 컴포넌트를 개발한 후에 유지보수하고 업그레이드하는데 필수적인 자료이다.

[그림 3]은 본 논문에서 제안하는 Component Specification에 필요한 항목들을 분류하여 트리구조(tree structure)로 표현한 것이다.

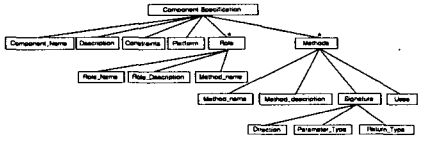

그림 3 컴포넌트 명세를 위한 항목들의 트리구조

3.2.2. 컴포넌트 명세 언어

앞 절에서 정의한 컴포넌트 명세에서 분석한 각 항목들을 이용하여 컴포넌트 명세 언어를 위한 DTD를 정의한다. 정의한 DTD에 의해서 [그림 4]와 같은 XML 기반의 컴포넌트 명세 언어 구조를 얻을 수 있다.

```
<?xml version="1.0"?>
<component_architecture>
  <component_name> </component_name>
  <description> </description>
  <constraints> </constraints>
  <platform> </platform>
  <role>
    <role_name> </role_name>
    <role_description> </role_description>
    <method_name> </method_name>
  </role>
  <methods>
    <method_name> </method_name>
    <method_description> </method_description>
    <signature>
      <direction> </direction>
      <parameter_type> </parameter_type>
      <return_type> </return_type>
    </signature>
    <uses> </uses>
  </methods>
</component_architecture>
```

그림 4 XML기반 컴포넌트 명세 언어 구조

4. 관계설정과 컴포넌트 후보추출

4.1. 컴포넌트 후보 추출을 위한 일치성 검사 규칙

각 항목들에 대한 의미적 일치성을 검사하여 후보 컴포넌트 추출에 필요한 기준을 제공한다.

[그림 5]는 각각의 관계들에 대한 의미적 일치성을 검사하는 규칙들 Rule1~Rule8까지 제안하여 정의하였다.

Rule 1.
Component Name 또는 Component Description이 Use Case Name 또는 Use Case Description과 의미적으로 포함되거나 일치해야 한다.

For component_name, component_description ∈ COMPONENT_SPEC,
usecase_name, usecase_description ∈ USECASE_SPEC.
R1(COMPONENT_SPEC, USECASE_SPEC) :
(component_name OR component_description) ⇒ (usecase_name OR usecase_description)

Rule 2.
Component Constraints가 Supplementary Specification의 Usability, Reliability, Performance, Supportability, Design Constraints, Use Case Specification의 Special Requirements 들과 의미적으로 위배되는 사항이 없어야 한다.

For constraints ∈ COMPONENT_SPEC, usability, reliability, performance, supportability, design_constraints ∈ SUPPLEMENTARY_SPEC,
special_requirements ∈ USECASE_SPEC
R2(COMPONENT_SPEC, SUPPLEMENTARY_SPEC, USECASE_SPEC) :
NOT((constraints) ⇒ (usability AND reliability AND performance AND supportability AND design_constraints AND special_requirements))

Rule 3.
컴포넌트가 수행될 수 있는 운영환경을 기술하는 Component Platform은 요구사항 명세의 Design Constraints에서 명시하는 운영환경을 포괄해야 한다.

For platform ∈ COMPONENT_SPEC,
design_constraints ∈ SUPPLEMENTARY_SPEC,
R3(COMPONENT_SPEC, SUPPLEMENTARY_SPEC) :
(design_constraints) ⇒ (platform)

Rule 4.
컴포넌트의 Role Name과 Role Description은 요구사항 명세의 사건의 흐름 설명하는 Flow Description의 내용에 포함되어야 하며 Rule 5보다 선행되어 검사되어야 한다.

For role_name, role_description ∈ COMPONENT_SPEC,
flow_description ∈ USECASE_SPEC,
R4(COMPONENT_SPEC, USECASE_SPEC) :
(role_name OR role_description) ⇒ (flow_description)

Rule 5.
컴포넌트의 Method Name과 Method Description은 요구사항 명세의 사건의 흐름을 설명하는 Flow Description의 내용에 포함되어야 한다.

For method_name, method_description ∈ COMPONENT_SPEC,
flow_description ∈ USECASE_SPEC,
R5(COMPONENT_SPEC, USECASE_SPEC) :
(method_name OR method_description) ⇒ (flow_description)

Rule 6.
Parameter Type은 요구사항 명세의 Information에서 요구하는 데이터 타입과 일치해야한다.

For parameter_type ∈ COMPONENT_SPEC,
information ∈ USECASE_SPEC,
R6(COMPONENT_SPEC, USECASE_SPEC) :
(parameter_type) ⇒ type(information)

Rule 7.
Return Type은 요구사항 명세의 Result에서 요구하는 데이터 형태와 일치해야 한다.

For return_type ∈ COMPONENT_SPEC,
result ∈ USECASE_SPEC,
R7(COMPONENT_SPEC, USECASE_SPEC) :
(return_type) ⇒ type(result)

Rule 8.
Uses는 컴포넌트의 Method가 사용되는 형태에 대한 설명으로 요구사항 명세의 Flow Description, Information, Result 항목들의 내용과 일치해야 한다.

For uses ∈ COMPONENT_SPEC,
flow_description, information, result ∈ USECASE_SPEC,
R8(COMPONENT_SPEC, USECASE_SPEC) :
(uses) ⇒ (flow_description OR information OR result)

그림 5 컴포넌트 후보 추출을 위한 일치성 검사 규칙

4.2 일치성 검사 규격에 의한 컴포넌트 후보 추출 방안

제안된 일치성 검사 규칙들을 이용하여 요구사항 명세로부터 컴포넌트 저장소에서 구축하고자 하는 시스템에 필요한 후보 컴포넌트를 추출할 수 있는 알고리즘을 제시하도록 하겠다.

다음은 제한하는 방안을 적용하기 위한 전제들을 정리해 놓은 것이다.

$$REQ\_SPEC = \{SUPPLEMENTARY\_SPEC,\ \Sigma_{i=1,n} USECASE\_SPEC_i\} \quad (1)$$

$$COMPONENT\_REPOSITORY = \Sigma_{i=1,n} COMPONENT\_SPEC_i \quad (2)$$

$$WEIGHT_{all} = WEIGHT(R1) + \ldots + WEIGHT(R8) \quad (3)$$

본 논문에서 의미적 일치 정도를 나타내는 *WEIGHT*값은 그 값이 클수록 요구사항에 알맞은 컴포넌트라는 것을 보여주기 위한 것으로 *WEIGHT*를 산출되는 구체적인 방법은 또 다른 연구가 필요할 것이다. [그림 6]은 위의 전제를 바탕으로 일치성 검사 규칙을 이용하여 컴포넌트 저장소로부터 컴포넌트 후보를 추출하는 알고리즘이다.

```
For each USECASE_SPEC{
While(Using of COMPONENT_REPOSITORY)
  if(R1(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R1) }
  if(R3(COMPONENT_SPEC), SUPPLEMENTARY_SPEC)) {
      set WEIGHT(R3) }
  else if {
      return WEIGHTall = 0 and break }
  }
  if(R4(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R4)
    if(R5(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R5) }
    if(R6(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R6) }
    if(R7(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R7) }
    if(R8(COMPONENT_SPEC), USECASE_SPEC)) {
      set WEIGHT(R8) }
  }
  if(R2(COMPONENT_SPEC), SUPPLEMENTARY_SPEC,
      USECASE_SPEC)) {
      set WEIGHT(R2) }
End of While
WEIGHTall = WEIGHT(R1) + ... + WEIGHT(R8)
Return ), WEIGHTall
```

그림 6 컴포넌트 후보 추출 알고리즘

결과 값은 컴포넌트와 일치성 검사 규칙에 의해 구한 전체 가중치를 반환한다.

이 알고리즘을 통하여 전체 시스템 개발의 초기 단계인 요구사항분석을 마친 후 컴포넌트 저장소로부터 시스템에 필요한 후보 컴포넌트를 추출함으로서 이후 시스템 개발 공정에서 이러한 후보 컴포넌트를 적용시킬 수 있다.

4.3. ITS 도메인 적용 사례

4.3.1. 실험 예제

본 논문에서 작성한 요구사항 명세서와 컴포넌트 명세서는 명세서간의 의미적 관계를 검사하기 위한 것으로 간소화시킨 것이다. 또한 첨단대중교통분야 시스템을 구축하기 위해 분석된 Use Case 중에서 "시내버스정보 제공"이라는 Use Case를 중심으로 후보 컴포넌트를 추출하였다. [그림 7]는 분석한 도메인 즉, 대중교통정보 시스템에 대한 요구사항을 본 논문에서 제안한 형식화된 요구사항 명세 언어로 표현한 것이다.

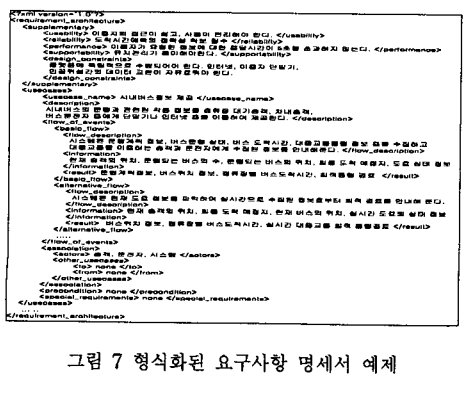

그림 7 형식화된 요구사항 명세서 예제

4.3.2. 컴포넌트 추출 예제

[그림 8]는 추출된 후보 컴포넌트들의 형식화된 명세서를 나타낸다. [그림 9]는 컴포넌트 후보 추출 과정을 적용된 일치성 검사 규칙들의 순서에 따라 표로서 보인 것이다.

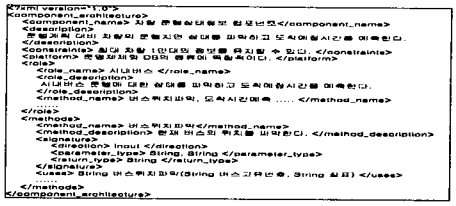

그림 8 형식화된 컴포넌트 명세서 예제1

적용된 일치성 검사 규칙	의미적 관계 설정시 고려된 문구	
	요구사항 명세서	컴포넌트 명세서
Rule1	시내버스정보 시내버스의 운행과 관련된 각종 정보	차량 운행상태 정보 운행계획 도착예정시간
Rule3	플랫폼에 독립적	운영체제와 DB의 종류에 독립적
Rule4	버스 운행 상태 버스 도착시간	시내버스 운행에 대한 상태 도착예정시간, 시내버스
Rule5	버스 운행 상태	버스 위치 파악
Rule6	none	none
Rule7	none	none
Rule8	버스위치정보 버스의 위치 운행되는 버스의 위치	버스의 위치 파악
Rule2	none	none

그림 9 알고리즘에 의한 일치성 검사 규칙 적용

5. 결론

본 논문에서는 컴포넌트 기반의 소프트웨어 시스템을 개발하는데 있어서, 컴포넌트 기반의 소프트웨어를 개발하기 위한 적합한 요구사항 명세화 방안과 컴포넌트 명세화 방안을 제시하였고, 각각의 명세를 XML로 표기하였기 때문에, 각각의 명세화가 구조적이며, 변경과 확장성이 용이하고, 웹 환경에서 적합한 컴포넌트를 검색하기도 용이할 뿐만 아니라, 요구사항 명세와 컴포넌트 명세를 매핑하기도 용이하다. 이 요구사항 명세와 컴포넌트 명세에서 각각의 항목별로 관계를 설정하는 규칙과 요구사항 명세를 가지고 컴포넌트 저장소에서 검색하는 알고리즘을 제안하여, 요구사항에 적합한 컴포넌트를 검색하는데 필요한 가이드 라인을 제시하였을 뿐만 아니라, 추출된 컴포넌트가 가중치를 가지고 있어서 시스템을 개발하는 개발자가 시스템에 적합한 컴포넌트를 추출하는데 도움을 준다. 본 논문에서 제시한 방안으로 컴포넌트를 개발하기 전에 명세화하여 유지한다면, 소프트웨어 개발의 생산성이 한층 높아 질 것이다.

6. 참고 문헌

[1] Alan W. Brown, Kurt C.Wallnau, "The Current State of CBSE", IEEE Software, September/October, 1998

[2] Jun-Jang Jeng and Betty H. C. Cheng, "A Formal Approach to Reuse More General Components", IEEE Proceedings of 9th Knowledge- Based Software Engineering Conference, Monterey, California, pp.90-97, September, 1994

[3] P.Allen and S.Frost, Component Based Development for Enterprise Systems: Applying the Select Approach, Cambridge University Press, New York, 1997

[4] 강기선, 김진태, 박병철, 박수용, "Requirement engineering: Overview", 소프트웨어공학회지, 1998