재사용 라이브러리 시스템에 대한 분류 기준

Classification Criteria for Reuse Library Systems

이 성 구*
Lee, Sung-Koo

요 약

소프트웨어 개발 생산성과 질을 개선하기 위한 재사용 접근 방법들과 이들을 지원하는 라이브러리 시스템들이 개발되었다. 이들 시스템들은 재사용 컴포넌트들을 효과적으로 분류, 저장, 검색, 이해하기 위해 다양한 방법을 적용한다. 그러나, 라이브러리 시스템들의 수가 증가할 때, 시스템들을 분류하고 그들의 속성을 비교/분석하는 것은 어렵다.

본 논문에서는 재사용 라이브러리 시스템들을 분류하기 위한 기준을 제시한다. 제시된 기준들은 컴포넌트의 속성을 코드화 하는 패싯(facet)과 속성(attribute) 기반 분류 방법의 결합에 의해 정의된다. 제안된 분류 기준에 대한 유용성을 보이기 위해, 컴포넌트 분류 방법과 응용 도메인에 기초한 대표적인 라이브러리 시스템들이 선택되고, 제안된 기준에 의해 분류된다.

Abstract

In order to improve software development productivity and quality, reuse approaches and supporting library systems have been proposed. Library systems have applied various methods to classify, store, retrieve, and comprehend reusable components effectively. As the number of library systems grows, it is difficult to categorize, compare and analyze existing reuse libraries.

In this paper, we present classification criteria for reuse library systems. A set of criteria is defined by integrating facet-based and attribute-based classification methods which encode the properties of a reusable component. In order to show the usefulness of the proposed classification criteria, representative library systems based on application domains, as well as component classification methods are selected and reviewed. We then classify these library systems according to the proposed criteria.

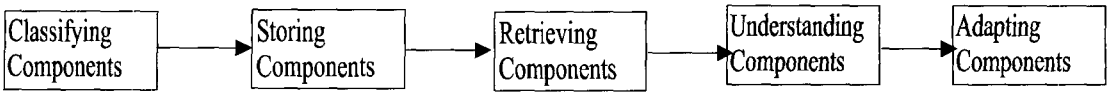
☞ Keyword : classification criteria, library system, facet, component

1. 소 개

소프트웨어 재사용에 대한 접근 방법은 일반적으로 'for reuse' 와 'with reuse' 방법으로 구분될 수 있다 [1,2]. 특히, 재사용 컴포넌트들로 구성된 라이브러리에서 소프트웨어 시스템을 개발하는 'with reuse' 방법은 개발될 소프트웨어에 후보 컴포넌트들을 검색하기 위한 라이브러리 시스템을 필요로 한다. <그림 1>은 'with reuse'에 의한 소프트웨어 개발을 위한 5가지 과정을 보인다.

컴포넌트들의 수가 증가할 때, 개발자는 각각의 컴포넌트를 조사하는 것은 불가능하다. 이에 따라 컴포넌트 분류 문제는 컴포넌트들의 속성 정보에 의해 유사한 컴포넌트들을 그룹하는 문제에 대해 언급하는 것이 필요하다. 또한, 'with reuse' 프로세스는 개발자의 필요에 대한 효과적인 질의 구성과 함께, 유사한 기능을 갖는 후보 컴포넌트들에 대한 검색 과정, 후보 컴포넌트들 사이에서 가장 적절한 컴포넌트를 선택하는 이해 과정, 그리고 선택된 컴포넌트를 수정함으로써 새로운 응용 프로그램에 적용시키는 적응 과정으로 구성된다. 결국, 이러한 과정을 체계적으로 지원하는 라이브러리

* 정회원 : 한신대학교 컴퓨터정보소프트웨어 학부 부교수
sklee@hanshin.ac.kr
[2006/08/10 투고 - 2006/09/06 심사 - 2006/09/26 심사완료]
☆ 이 논문은 2006년도 한신대학교 학술연구비 지원에 의하여 연구되었음



〈그림 1〉 'with reuse' 소프트웨어 개발 프로세스

시스템은 필요로 된다.

'with reuse' 프로세스를 지원하는 다양한 라이브러리 시스템들이 제안되었다. 그러나, 다양한 재사용 방법과 시스템 기능을 적용하는 라이브러리 시스템들의 수가 빠르게 증가할 때, 시스템들을 효과적으로 분류하여 그들의 속성을 비교/분석하는 것은 어렵다. 이를 위해, 본 논문에서는 라이브러리 시스템 분류 기준을 정의한다. 제시된 분류 기준은 전통적인 라이브러리 시스템들과 미래의 시스템들에 대한 분류 프레임워크로 이용되어 그들 사이의 재사용 방법에 의한 차이는 물론 시스템 속성에 의한 비교는 용이할 것이다.

라이브러리 시스템에 대한 분류 기준은 컴포넌트 재사용에 대한 접근 방법에 따라 정의될 수 있다. 예를 들면, 시스템은 다양한 도메인에 적용될 수 있는 컴포넌트들에 대한 재사용 접근 방법을 지원하는가 혹은 단일 도메인에 적용하는가에 의해 분류될 수 이다. 즉, 시스템이 지원하는 도메인에 대한 영역은 시스템 분류 기준의 하나가 된다. 재사용 접근 방법에 대한 분류 구조는 패싯 기반 방법을 사용한다. 또한, 라이브러리 시스템들은 1장에서 언급된 'with reuse' 소프트웨어 개발 과정에 적용되는 다양한 방법에 의해 분류될 수 있다. 개발 과정을 지원하는 방법은 다른 시스템과 구별되는 중요한 속성이 된다. 그러므로, 시스템 속성은 분류 기준으로 정의될 수 있으며, 이를 위해 속성 기반 분류 방법을 이용한다. 본 논문에서 정의된 기준으로 구성되는 분류 구조는 컴포넌트 표현을 위해 사용되는 패싯(facet)과 속성(attribute) 기반 분류 방법을 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 재사용 접근 방법과 시스템 지원 속성에 의해 정의되는 시스템 분류 기준에 대해 논의한다. 3장은 컴포넌트 분류 방법에 기초한 라이브러리 시스템들이 소개된다. 4장은 정의된 분류 기준에 의해 3장에서 언급된 라이브러리 시스템들은 분류되고 비교된다. 5장에서 본 논문의 결론과 향후 과제에 대해 언급한다.

2. 분류 기준

본 논문에서 제안된 분류 기준은 재사용 컴포넌트 표현을 위해 사용되는 패싯(facet) 기반과 속성(attribute) 기반 분류 방법의 결합을 통해 정의된다. Prieto-Diaz에 의해 제안된 패싯 기반 방법은[3] 패싯과 각 패싯에 속하는 값을 결정함으로써 컴포넌트에 대한 속성이 표현된다. 패싯과 패싯 값은 도메인 분석을 통해 미리 정의되며, 패싯은 컴포넌트의 다양한 관점을 나타낸다. 컴포넌트 표현을 위해 이용되는 속성 기반 분류 방법은 해당 도메인에 대한 속성을 미리 정의한다. 그러나 각 속성에 주어지는 값들은 정의되지 않는다. 컴포넌트들은 그들의 속성을 가장 잘 표현하는 속성 값들을 결정함으로써 분류된다. 속성 기반 분류 방법은 속성 값들이 미리 정의되지 않는다는 것을 제외하면 패싯 분류 방법과 유사하다.

2.1 접근 방법에 의한 분류

Prieto-Diaz는 컴포넌트 분류를 위해 제안된 패싯 기반 방법을, 소프트웨어 재사용 접근 방법들을 분류하기 위해 적용하였다[4]. 표1은 6개의 패싯들로 구성된 재사용 접근 방법에 대

〈표 1〉 Frakes 분류기준

패싯	값	패싯	값
개발범위	Internal	도메인범위	Vertical
	External		Horizontal
		관리	Systematic
수정	White Box		Ad Hoc
	Black Box	재사용 엔터티	Code
	Adaptive		Abstract Level
접근방법	Generative		Instance Level
	Compositional		Customization Reuse
	In-the-Small		Generic
	In-the-Large		Source Code
	Indirect		
	Direct		
	Carried Over		
	Leveraged		

〈표 2〉 Karsson 분류기준

패싯	값
재사용범위	General
	Domain
	Product Line
재사용목표	Internal
	External
컴포넌트 크기 (granularity)	Fine
	Coarse

한 분류 기준을 보인다. 또한, <표 2>는 Karlsson에 의해 제안된 재사용 접근 방법 분류를 위한 3가지 분류 기준을 보인다[5].

본 논문에서 제안된 라이브러리 시스템 분류 기준에 대한 프레임 워크를 제공하기 위해, <표 1>과 <표 2>로부터 라이브러리 시스템 분류를 위해 적당한 패싯들이 선택된다.

대부분의 라이브러리 시스템들은 이미 존재하는 컴포넌트 재사용에 의한 소프트웨어 개발(compositional)을 지원한다. 그러므로, <표 1>의 '접근방법' 패싯은 본 논문에서 제안된 분류 기준에 포함되지 않는다. 또한, 본 논문은 재사용에 대한 관리 문제(예, 계약, 법, 재정)는 언급하지 않고, 기술적인 문제들에 대해 언급하므로, '관리' 패싯은 제외된다. '수정' 분류 기준에 대한 값인 'White Box'는 컴포넌트 내부를 새로운 소프트웨어에 맞게 수정(적응)하는 것을 의미하므로 'Adaptive'와 결합될 수 있다. <표 2>의 '컴포넌트 크기' 패싯은 관리되는 컴포넌트의 크기를 의미하므로 <표 1>의 '재사용 엔터티'와 유사하다. 이들을 고려했을 때, 재사용 접근 방법으로부터 시스템 분류를 위한 제안된 기준은 다음과 같다.

- 개발 범위 : 재사용 컴포넌트들이 현재 프로젝트의 내부(혹은 외부)에서 제공되는가를 언급
- 수정 : 컴포넌트들이 수정 없이(혹은 수정) 재사용 되는가를 언급
- 도메인 범위 : 컴포넌트 재사용이 하나의(혹은 둘 이상) 도메인에 적용되는가를 언급
- 컴포넌트 크기 : 재사용 컴포넌트의 크기를 언급

2.2 시스템 속성에 의한 분류

라이브러리 시스템은 재사용 컴포넌트들의

분류, 저장, 검색, 이해 과정을 지원하는 자동화된 도구이다. 시스템은 이러한 과정에 대한 시스템 지원 방법(속성)에 의해 분류될 수 있다. 제안된 분류 기준은 다음과 같다.

- 컴포넌트 타입 : 라이브러리에 존재하는 컴포넌트 타입
- 분류 : 시스템에 이용되는 컴포넌트 분류 방법
- 검색 : 후보 컴포넌트들을 검색하는데 이용되는 메커니즘
- 이해 : 재사용을 위해 후보 컴포넌트를 이해하기 위해 지원되는 방법
- 구현 환경 : 시스템 구현 환경

2.3 시스템 분류 기준

<표 3>은 본 논문에서 제안하는 라이브러리 시스템 분류 기준에 의한 구조를 보인다. 재사용 접근 방법에 의한 분류 기준에서 패싯 값들은 미리 정의된다. 반면에, 시스템 속성에 의한 분류 기준에서 각 속성에 대한 값들은 정의되지 않았다. 'with reuse' 과정에 적용되는 방법들은 다양하다. 또한, 이들에 대한 정의된 값은 시스템 사이의 고유한 특성을 비교하기에 한계를 갖는다. 그러므로, 속성 기준 값들은 시스템 속성을 가장 잘 묘사하는 값으로 결정된다.

<표 3>에서, '컴포넌트크기'에 대한 분류 기준 값으로 'fine grained'는 크기에서 작은 컴포넌트 재사용 접근 방법을 의미한다. 그러므로 이러한 컴포넌트에 대한 재사용은 다양한 도메인에 적용 가능하지만, 개발 생산성은 낮다는 것을 의미한다.

3. 라이브러리 시스템

본 장에서는 전통적인 라이브러리 시스템들에 대해 소개한다. <표 3>에 제시된 시스템 분류 기준에서 컴포넌트를 표현하기 위해 적용되는 '분류' 방법은 라이브러리 시스템 설계의 근본이 된다 [4-6]. 즉, 적용된 컴포넌트 분류 방법에 따라 검색, 저장, 이해 과정에 대해 전반적인 영향을 미친다. 그러므로, 본 논문에서

〈표 3〉 시스템 분류 기준

	분류기준	설 명	값
재사용 접근방법	개발범위	재사용 컴포넌트들이 현재 프로젝트의 내부(혹은 외부)에서 제공되는가를 언급	Internal
			External
	수정	컴포넌트들이 수정없이(혹은 수정) 재사용 되는가를 언급	White Box
			Black Box
	도메인범위	컴포넌트 재사용이 하나의(혹은 둘이상) 도메인에 적용되는가를 언급	Vertical
			Horizontal
	컴포넌트크기	재사용 컴포넌트의 크기를 언급.	Fine-grained
			Coarse-grained
시스템 속성	컴포넌트타입	라이브러리에 존재하는 컴포넌트 타입	
	분류	시스템에 이용되는 컴포넌트 분류 방법	
	검색	후보 컴포넌트들을 검색하는데 이용되는 메커니즘	
	이해	재사용을 위해 후보 컴포넌트를 이해하기 위해 지원되는 방법	
	구현환경	시스템 구현 환경	

전통적인 라이브러리 시스템들에 대한 소개는 컴포넌트 분류 방법에 의해 구분된다.

일반적으로 컴포넌트 분류 방법들은 열거(enumeration), 속성, 패싯, 지식, 명세(specification), 텍스트 기반 방법들로 구분된다. 열거 기반 분류 방법은 도메인으로부터 정의된 계층 클래스 구조로부터 컴포넌트를 가장 잘 묘사하는 클래스를 선택함으로써 컴포넌트를 표현한다. 열거, 속성, 패싯 기반 방법들은 모두 도메인 분석에 의해 미리 정의된 한정된 어휘를 사용하므로 어휘 기반 분류 방법이라고 부른다[2]. 어휘 기반 방법들 중, 대부분의 시스템은 분류/검색의 용이함 때문에 패싯 기반 방법을 지원한다.

본 장에서 4가지 컴포넌트 분류 방법들(패싯, 지식, 명새, 텍스트)을 적용한 대표적인 라이브러리 시스템들이 소개된다. 각 분류 방법에 대한 시스템들은 라이브러리를 구성하는 컴포넌트 타입(예, 함수, 클래스, 서브시스템)의 차이에 의해 선택된다. 소개되는 시스템들에 대한 비교는 <표 3>에 제시된 시스템 분류 기준에 의해 4장에서 논의된다.

3.1 패싯 기반 시스템

Prieto-Diaz[7]에 의해 제안된 시스템은 코드(함수) 컴포넌트 분류를 위해 패싯과 패싯 값을 정의한다 <표 4>. 컴포넌트는 서로 다른 패싯에 대한 적절한 값을 선택함으로써 표현된다. 사용자 질의 관련 후보 컴포넌트에 대한 검색은 패싯에 할당된 값들 사이의 근접 관계를

〈표 4〉 코드 컴포넌트 분류 기준

패싯	값
Function	add
	append
	...
Objects	arrays
	arguments
	...
Medium	array
	buffer
	...
System Type	assembler
	compiler
	...
Functional Area	auditing
	builing
	...
Setting	advertising
	association
	...

〈표 5〉 객체 컴포넌트 분류 기준

패싯	값
Abstraction	financial
	transaction_ handler
	...
Operations	get_invoice
	notify_pay_ invoice
	...
OperatesOn	invoice
	account
	...
Dependencies	C++
	Gardy_ Booch
	...

〈표 6〉 소프트웨어 결과물 분류 기준

패싯	값
Language	Ada
	C
	...
Domain	accounting
	administraticve
	...
Function	add
	analyze
	...
Data	task
	thread
	...
Operation System	AIX
	OS/2
	...

나타내는 개념 근접 그래프(conceptual closeness graph)에 기초된다. 패싯 값들 사이의 근접 관계를 측정하는 개념 근접 그래프는 3.2절에서 언급될 지식 기반 방법의 개념 그래프(conceptual graph)와 유사하다.

REBOOT 프로젝트는 Prieto-Diaz의 패싯 분류 방법을 사용한다[8]. 그러나, REBOOT에서 라이브러리 구성 컴포넌트들은 객체 지향 컴포넌트들이므로, 객체지향 문맥에 적용될 수 있는 패싯들로 구성되는 구조를 정의한다 <표 5>. 개념적으로 관련되는 후보 컴포넌트 검색을 위해 패싯 값들 사이의 관계를 나타내는 용어 공간(term space) 개념을 이용한다. 결국, 용어 공간은 지식 기반 분류 방법의 의미 망(semantic network)과 개념 그래프 개념을 결합한 것이다.

Poulin에 의해 제안된 FRR (Federal Reuse Repository) 시스템[9]은 소프트웨어 생명 주기에서 생성되는 소프트웨어 결과물 (software artifacts)에 대한 패싯들을 정의한다 <표 6>. 특히, 컴포넌트들에 대한 검색, 저장, 평가를 위해 SR(Structural Abstracts)이라 불리는 템플릿을 이용한다. FRR 시스템은 다른 사이트에 존재하는 컴포넌트들을 공유하기 위한 웹기반 시스템이다.

3.2 지식 기반 시스템

Wood에 의해 제안된 분류 방법[10]은 주로 자연어 이해 시스템에 이용되는 개념그래프에 기초된다. 개념그래프는 적용 도메인에 대한 근본적인 개념들과 의미 정보를 표현하기 위해 이용된다. 개념그래프를 컴포넌트 분류에 적용하기 위해 컴포넌트 서술자 프레임(CDF : Component Descriptor Frame) 구조를 개발했다. 기본 개념들은 각 컴포넌트가 수행하는 함수들과, 컴포넌트에 의해 다루어 지는 객체들로 구성된다. 개념적으로 유사한 기능들은 기본 함수로 분류된다. CDF는 이러한 기본 함수들에 대해 개발된다. 이 시스템은 UNIX 컴포넌트 도메인에서 개발된 25개의 기본 함수들에 대한 CDF를 정의했다.

Gangopadhyay에 의해 제안된 시스템은[11] 객체 지향 클래스 컴포넌트들로 구성된 그래픽 도메인 지식을 표현하기 위해 술어 논리(predicate logic) 지식 표현 방법을 이용했다. 객체 지향 프로그래밍과 그래픽 프로그래밍에 대한 지식은 개념 모델(conceptual model)로 표현된다. 즉, 도메인에 존재하는 중요한 개념, 개념에 대한 속성, 그리고 그들 사이의 복잡한 관계는 술어 논리 규칙을 이용하여 정의된다.

3.3 명세 기반 시스템

Chen에 의해 제안된 시스템은[12] 컴포넌트 표현과 질의 표현을 위해 정형 명세 언어인 ASL(Algebraic Specification Langauge)을 이용한다. Ada 패키지와 클래스 사이의 관계는 정의된 오퍼레이션을 이용한다. 컴포넌트 의미 정보는 오퍼레이션 사이의 관계를 정의한 공리(axiom)에 의해 표현된다.

Nelson에 의해 제안된 CSRS(Class Storage and Retrieval System)는[13] 객체 지향 컴포넌트에 대한 인터페이스와 기능을 명세하기 위해 PSDL(Prototyping System Description Language) 언어를 사용한다. 각 컴포넌트에 대한 PSDL 명세는 컴포넌트의 의미 정보를 표현하기 위해 OBJ3 명세 언어를 사용한다.

3.4 텍스트 기반 시스템

Henninger에 의해 제안된 CodeFinder 시스템은[14] 텍스트 기반 인덱싱과 함께, 유사 컴포넌트들에 대한 구별을 위해 spreading activation 기술을 이용하여 Emacs 컴포넌트들 사이의 연관 네트워크를 구성한다. 연관 네트워크는 단어들과 컴포넌트들을 위한 2 계층 노드

로 구성된다. 노드 사이의 link weight는 컴포넌트들의 텍스트 파일로부터 각 컴포넌트에 포함된 단어들의 빈도에 의해 결정된다. 구성된 연관 네트워크에 기초하여 사용자로부터 주어진 컴포넌트와 유사한(연관된)컴포넌트들은 구별된다.

Helm에 의해 제안된 시스템은[15] C++ X11 그래픽 라이브러리 클래스 컴포넌트들에 대한 인덱싱과 함께, 컴포넌트 관련 문서들 사이에서 공통으로 존재하는 단어들을 고려하여 유사 컴포넌트들이 결합된다. 시스템은 개념적으로 관련된 컴포넌트들의 계층 구조를 자동으로 생성한다.

CAASD(Center for Advanced Aviation System Development) 자원 개발 시스템[16]은 텍스트, 이미지, 소프트웨어와 같은 자원(asset) 재사용을 지원한다. 인터넷에서 정보 저장과 검색에 이용되는 WAIS (Wide Area Information Server) 프로그램은 시스템 구현을 위해 이용된다. 후보 컴포넌트 검색은 질의에 포함된 단어와 컴포넌트에 매칭되는 단어의 빈도에 의해 수행된다.

4. 시스템 분류

<표 7>은 <표 3>에서 정의된 시스템 분류 기준에 의해 3장에서 언급된 다양한 라이브러리 시스템 분류 결과를 보인다. <표 7>에서 CG, SA, CDF, ASA, NA는 각각 Conceptual Graph, Structural Abstract, Component Descriptor Frame, Associative Spreading Activation, Not Available을 언급한다.

재사용 접근 방법에 대한 라이브러리 시스템 기준에 의한 분류를 고려할 때, 대부분의 전통적인 시스템들은 프로젝트 외부에서 제공되는 컴포넌트 재사용(external), 컴포넌트의 수정 없는 재사용(black box), 다양한 도메인에 적용되는 컴포넌트 재사용(horizontal), 그리고 함수, 클래스와 같은 작은 크기의 컴포넌트 재사용(fine-grained)을 지원한다는 것을 알 수 있다.

시스템 속성에 의한 분류에서 도메인 분석을 통해 한정된 어휘를 이용하는 라이브러리 시스템들[7-9]은 용이한 분류와 효율적인 검색 때문에 일반적으로 패싯 기반 분류 방법을 사용한다는 것을 알 수 있다. 이러한 시스템들은 질의 관련 후보 컴포넌트 검색을 위해 도메인의 중요한 개념 사이의 관계를 나타내기 위해 개념 그래프를 이용한다.

도메인에 대한 지식을 코드화 하여 컴포넌트를 분류하는 시스템들[10,11]은 도메인에 존재하는 복잡한 개념 관계를 표현하기 프레임, 관계, 규칙들로 구성되는 개념 모델을 지원한다. 또한, 이러한 시스템들은 논리 기반 규칙에 의해 질의와 관련된 후보 컴포넌트들의 검색에 추론 능력을 지원한다.

컴포넌트에 대한 행위 표현과 질의 표현에 대해 정형 명세 언어를 사용하는 시스템들은[12,13] 사용자 입력과 의미적으로 유사한 후보 컴포넌트 검색을 위해 이론 증명기(theorem prover)를 사용한다. 컴포넌트 의미 정보는 컴포넌트 행위 사이의 관계를 정의하는 공리에 의해 표현된다.

컴포넌트 표현과 검색 과정에 대한 시스템 자동화 가능성 때문에 많은 라이브러리 시스템은[14-16] 텍스트 기반 분류 방법을 이용한다. 그러나, 이러한 시스템들은 컴포넌트의 의미, 혹은 그들 사이의 관계에 대한 정보를 표현하는 것은 어렵다. 유사 컴포넌트들에 대한 검색은 컴포넌트에 포함된 질의 관련 키워드 빈도 혹은 컴포넌트 사이의 클러스터링 프로세스를 통해 수행된다.

특히, 웹기반 지식(혹은, 명세) 기반 라이브러리 시스템은 거의 존재하지 않는다. 비록 이러한 분류 방법들이 의미적으로 더욱 유사한 후보 컴포넌트들을 검색할 수 있는 능력을 제공한다 해도, 도메인 지식 표현과 명세 언어

〈표 7〉 시스템 분류 결과

	분류 기준	[7]	[8]	[9]	[10]
재사용 접근방법	개발범위	External	N.A.	External	External Internal
	수정	Black Box	White Box	Black Box White Box	Black Box White Box
	도메인범위	Horizontal	N.A.	Horizontal	Horizontal Domain
	컴포넌트크기	Fine-grained	Fine-grained	Fine-grained	Fine-grained
시스템속성	컴포넌트타입	Functions	OO Classes	Functions Packages ADTs	Functions Packages ADTs
	분류	패싯기반	패싯기반	패싯(+텍스트)기반	프레임기반 (지식기반)
	검색	C.G. Thesaurus	C.G.	S.A.	C.D.F.
	이해	N.A.	N.A.	Description	N.A.
	구현환경	N.A.	N.A.	Web	N.A.
		[11]	[12]	[13]	[14]
재사용 접근방법	개발범위	External	External	Internal	Internal
	수정 도메인범위	White Box Horizontal	White Box Horizontal	White Box Vertical	Black Box Horizontal
	컴포넌트크기	Fine-grained	Fine-grained	Fine-grained	Fine-grained
시스템속성	컴포넌트타입	OO Classes	Packages OO Classes	OO Classes	Functions
	분류	논리기반(지식기반)	공리(명세기반)	공리(명세기반)	텍스트기반
	검색	Relation, Rules	Theorem-prover	Theorem-prover	A.S.A.
	이해	N.A.	N.A.	N.A.	N.A.
	구현환경	N.A.	N.A.	N.A.	N.A.
		[15]	[16]	[2]	
재사용 접근방법	개발범위	External	External Internal	External	
	수정	Black Box White Box	N.A.	N.A.	
	도메인범위	Horizontal	Vertical	Horizontal	
	컴포넌트크기	Fine-grained	N.A.	Fine-grained	
시스템속성	컴포넌트타입	OO Classes	Assets	OO Classes	
	분류	클러스터 (텍스트기반)	텍스트기반	복합클러스터 (텍스트기반)	
	검색	Clustering	Frequency	Clustering	
	이해	Relationships	N.A.	Summary	
	구현환경	N.A.	Web	Web	

* [ ]의 숫자는 참고문헌 번호를 의미한다

처리에 대한 어려움, 그리고 비효율적인 검색 능력은 웹에서 구현되기 힘들다.

SOORLS 시스템은[2] 시스템 속성에 대한 분류 기준으로부터 다른 시스템들과 차이를 보인다. 컴포넌트들의 자동화된 분류를 위해 정보 과학 분야에서 이용되는 계층적 그리고 비계층적 클러스터 방법을 결합하는 복합 클러스터 방법을 지원한다. 또한, 분류 과정에서 자동 생성되는 라이브러리 요약 내용은 사용자로부터 검색 초기에 라이브러리의 전체적인 내용을 이해하도록 지원하며, 효율적인 검색을 돕는다. 또한, SOORLS는 웹기반 환경에서 구현되어 컴포넌트 제출을 포함한 거의 모든 과정에 대한 자동화를 지원한다.

5. 결 론

본 논문에서 컴포넌트 재사용을 지원하는 다양한 라이브러리 시스템들에 대한 분류 기준이 제시되었다. 제시된 분류 기준은 프로젝트에서 컴포넌트 재사용을 위해 접근하는 방법에 대한 기준과 함께, 'with reuse' 방식의 재사용 과정에 적용되는 시스템 속성에 의해 정의되었다. 이러한 분류 기준으로 구성된 구조를 생성하기 위해 컴포넌트 표현을 위해 사용되는 패싯과 속성 기반 분류 방법이 사용되었다. 정의된 분류 기준에 의해 전통적인 라이브러리 시스템들에 대한 분류와 그들의 속성 비교에 대한 용이함을 보였다. 또한, 제시된 분류 기준은 미래의 시스템들에 대한 분류 프레임워크로 이용되는 것이 기대된다.

본 논문에서 제안된 시스템 분류 기준은 라이브러리 시스템에 대한 성능 평가 방법, 컴포넌트 개발 지원 기능, 컴포넌트 저장 방법과 같은 다양한 관점을 고려하여 지속적으로 정제되고, 확대되는 것이 필요하다. 즉, 사용자 노력, 응답 시간, 회상도, 정확도에 의한 시스템 평가 기준, 초기 시스템으로부터 재사용 컴포넌트를 축출(역공학)하는 컴포넌트 개발 방법, 그리고 컴포넌트와 컴포넌트 사이의 관련 정보를 저장하는 방법에 기초한 분류 구조에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] A. Karlsson, *Software Reuse: A Holistic Approach*, John Wiley & Sons, 1995.

[2] Lee, S. K., "Web 소프트웨어 컴포넌트 재사용을 위한 라이브러리 관리와 서비스," 정보과학회논문지, 제29권 제1호, pp. 10-19, 2002.

[3] R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software*, Vol. 4, No. 1, pp. 6-16, Jan. 1987

[4] W. B. Frakes, "A case Study of a Reusable Component Collection in the Information Retrieval Domain," *The Journal of Systems and Software*, Vol. 72, No. 2, pp 265-270, 2004.

[5] W. N. Bruce W. N. and Huilin Ye, "Self-Organization of Software Asset Library for Reuse," *Journal of Systems Research and Information Systems*, Vol. 10, pp 3-21, 2001.

[6] W. B. Frakes and K. Kang, "Software Reuse Research: Status and Future," *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, pp. 529-536, 2005.

[7] R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," *Communications of the ACM*, Vol. 34, No. 5, pp. 89-97, May 1991

[8] A. Karlsson, S. Sorumgard, and E. Tryggeseth, "Classification of Object-Oriented Components for Reuse," *Technology of*

Object-Oriented Language and Systems: TOOLS7 (ed. Heeg, Magnusson, and Meyer), Dortmund, Germany, pp. 21-31, 1992

[9] J. S. Poulin and K. J. Weckman, "Melding Structured Abstracts and the World Wide Web for Retrieval of Reusable Components," *Proceedings of the Symposium on Software Reusability: SSR'95*, Seattle, WA., pp. 160-168, 1995.

[10] M. Wood and I. Sommerville, "An Information Retrieval System for Software Components," *Software Engineering Journal*, Vol. 3, No. 5, Sep. 1988.

[11] D. Gangopadhyay and A. R. Helm, *A Model Driven Approach for the Reuse of Classes from Domain Specific Object -Oriented Class Repositories,* I.B.M. Technical Report, RC 14510(#64975), Thomas J. Watson Research Center, 1989.

[12] P. S. Chen, R. Hennicker, and M. Jarke, "On the Retrieval of Reusable Software Components," *Proceedings of Advances in Software Reuse*, Lucca, Italy, pp. 99-108, 1993.

[13] M. L. Nelson and T. Poulis, "The Class Storage and Retrieval System: Enhancing Reusability in Object-Oriented Systems," *OOPS Messenger*, Vol. 6, No. 2, pp. 28-36, Apr. 1995.

[14] S. Henninger, "Supporting the Construction and Evolution of Component Repositories," *Proceedings of 18th International Conference on Software Engineering (ICSE-18'96)*, pp. 279-288, 1996.

[15] R. Helm and Y. S. Maarek, "Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries," *OOPSLA'91*, pp. 47-61, 1991.

[16] T. Stockwell and M. Krause, "Internet Information Discovery and Retrieval Tools - Cost Effective Building Blocks for Asset Libraries," *Proceedings of the 6th International Workshop on Software Reuse (WISR6)*, 1993.

◐ 저 자 소 개 ◑

이 성 구 (Lee, Sung-Koo)

1980년 중앙대학교 전자계산학과 학사
1989년 중앙대학교 전자계산학과 석사
1993년 애리조나 주립대학 전자계산학과 석사
1998년 애리조나 주립대학 전자계산학과 박사
1999년 ~ 현재 한신대학교 컴퓨터정보소프트웨어 학부 부교수
관심분야는 소프트웨어공학, 인터넷 프로그래밍, 및 시스템 분석/설계