논문 2016-2-4

소프트웨어 재사용을 위한 연관성 분석 모델

권기태*, 이나영

Association analysis model for Software Reuse

Ki-Tae Kwon*, Na-Young Lee

요 약

소프트웨어 재사용은 소프트웨어 개발 비용을 감소하고 소프트웨어 품질을 향상시킨다. 그러나 소프트웨어 재사용을 하기 위해서 기존의 소프트웨어에 대한 전반적인 이해가 필요하다. 그렇지 않으면 새로 개발하는 것이 더 효율적이다. 이에 본 논문에서는 Apriori 알고리즘을 적용하여 소프트웨어 재사용 성공을 위한 연관성 분석 모델을 제안한다. 제안된 모델은 재사용 데이터 세트에 적용하여 재사용 성공요인과 column의 연관성을 분석한다. 실험 결과 소프트웨어 재사용은 Rewards Policy=no, Type of Software production=product-family, Human Factors=yes, Repository=yes, Top Managemanet Commitment=yes 일 때 성공함을 보였다. 그리고 6개의 column은 서로 연관성이 있음을 보였다.

Abstract

Software reuse is to reduce software develop cost and to improve software quality. But the general understanding of existing software is required in order to software reuse. Otherwise It is more efficient to newly software developed.In this study, we proposed an association analysis model for Software Reuse success using Apriori algorithm. We analyzed success factor and column's association of software reuse in software reuse dataset. The evaluation results showed that software reuse is successful when Rewards.Policy=no, Type of Software production=product-family, Human Factors=yes, Repository=yes, Top Management Commitment=Yes. And we showed that six columns have association with each other.

한글키워드 : 소프트웨어 품질, Apriori 알고리즘, 연관성 규칙

keywords : software quality, Apriori algorithm, relation rule

1. 서 론

소프트웨어 재사용은 새로운 시스템을 구성하기 위해 기존의 소프트웨어 컨포넌트를 사용하는 것을 의미한다. 재사용 적용은 단지 일부분의 소스코드만이 아니라 소프트웨어 개발을 하는 동안 이루어지는 요구사항 설계, 시스템 상세화, 구조 디자인, 소프트웨어 생성을 위한 개발자들의 정보 등을 포함한다.[1]

이러한 소프트웨어 재사용은 소프트웨어 개발 비용을 감소하고 소프트웨어 품질을 향상시킨다. 그리고 신뢰할 수 있는 소프트웨어를 사용함으로써 프로젝트의 실패 위험을 감소시킨다. 그래서

* 강릉원주대학교 컴퓨터공학과 교수
(email: ktkwon@gwnu.ac.kr)
접수일자: 2016.11.25. 심사완료: 2016.12.14.
게재확정: 2016.12.22.

최근 10여년간 재사용의 연구는 아이디어와 산업 활용 측면에서 재사용의 이러한 장점을 고려하여 발전하고 있다.[2]

1968년 NATO 소프트웨어 공학 컨버런스에서 소개된 소프트웨어 재사용은 1974년에 재사용이 가능한 소프트웨어 컨포넌트와 자동화된 기술의 라이브러리인 소프트웨어 팩토리 개념을 Mcllroy가 제안하였다. 이것은 시스템 개발 기업의 시스템 팩토리에서 재사용을 기대하는 프로그래머들이 수치 계산, I/O 전환, 텍스트 프로세싱, 동적 저장 분배를 위한 컨포넌트의 라이브러리를 이용하는 것이었다. [3]

1970년 중반에 Robert Lanergan은 조직 레벨에서 재사용 프로젝트를 사용하기 시작했다. 1970년 후반에 재사용은 관심이 대두되어 아카데미를 통해 확산되고 국제적인 워크샵을 통해 연구되어 지기 시작했다.[1]

1983년 Freeman은 새로운 소프트웨어 시스템의 설계를 할 때 기존 소프트웨어 산출물을 사용하였다. 이것은 일부분의 소스코드를 사용하는 것과는 달리 구조디자인, 모듈 레벨 실행구조, 상세화, 문서화, 변형 등을 포함했다.[3]

1980년 후반에 재사용은 라이브러리 시스템, 분류 기술, 재사용 컨포넌트의 생성과 분배, 재사용 지원 환경, 기업 재사용 프로그램 등으로 발전하였고 1990년대는 경영, 경제, 문화, 법과 같은 비전문적인 곳에서도 소프트웨어 재사용이 사용되어 졌다.[1,3]

그러나 다양한 연구기법과 접근방법에도 불구하고 소프트웨어 재사용은 어떻게 무엇을 사용할지에 대한 요소 선정 및 결정 부분에 대한 기준 문제, 표준화 문제, 소프트웨어와 언어의 이질성에 따른 문제, 소프트웨어 요소 내부와 인터페이스의 이해, 유지 보수 관리 등 여러 가지 문제를 가지고 있다. 이에 본 논문은 소프트웨어 재사용 데이터세트에 장바구니 분석에 활용되는 Apriori 알고리즘을 활용하여 재사용이 성공한 요인과 그에 따른 factor의 상관관계를 통해 연관성을 분석한다.

제 2장에서는 소프트웨어 재사용에 관한 방법론과 Apriori 알고리즘을 살펴보고 제3장에서는 연관성 분석을 위한 모델을 제안한다. 제4장은 실험을 통해 모델을 평가하고 분석한 후 제 5장에서 결론에 대해 기술한다.

2. 관련 연구

2.1 소프트웨어 재사용 기법

2.1.1 API(Application programing Interface)

API는 소프트웨어 재사용의 일반적인 방법으로 여러 가지 프로그램의 종류에 의해 서비스를 실행한다. 그리고 주어진 형식으로부터 데이터를 전환하고 리소스, 파일, 데이터베이스 등을 접근하기 위해 주어진 함수를 활용한다.

2.1.2 Framework

프레임워크는 미완성 시스템으로 시스템 표준 구조를 실행하는 클래스와 인터페이스를 구성한다. 이 시스템은 빠진 부분을 채우기 위해 클래스를 추가함으로써 실행된다. 예를 들면 프레임워크에서 추상화된 클래스는 시스템에서 확장되어야 한다. 프레임워크의 주된 특징은 컨트롤의 도치, 사용자에 의한 확장성, 변경할 수 없는 프레임워크 코드, 기본동작이 있다는 것이다. 프레임워크는 컴파일환경과 같은 시스템을 위한 기본 생성을 지원하는 기본 프레임워크, 컴포넌트 통신 사이에 메시지 교환을 지원하는 통합 프레임워크, 웹 어플리케이션과 같은 새로운 어플리케이션 개발을 지원하는 어플리케이션 프레임워크로 분류된다.

2.1.3 Design Patterns

디자인 패턴은 일반적으로 문제에 대해 재사용이 가능한 문제의 해결방법이다. 패턴은 경험에 의한 지식을 사용하는 잘 알려진 가장 좋은 기법중의 하나이다. 그리고 다른 어플리케이션에서 재사용이 지원되기 때문에 완벽한 해결을 나타내지 못한다. 대표적인 방법으로는 GOF(Gangs of Four)를 들 수 있다. 그리고 디자인 패턴은 element, observer로 구분된다. 이것은 이름, 문제 서술, 솔루션, 결과 과정으로 나타나는 데 element는 다른 어플리케이션에서 사용할 수 있는 템플릿과 같은 솔루션이고 observer는 [Figure 1]과 같은 솔루션으로 둘은 구분된다.

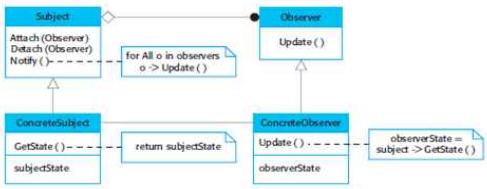


그림 1. Observer solution
Fig. 1. Observer solution

2.1.4 MDD(Model-driven Development)

언어와 플랫폼에 의존적인 세부사항을 포함하는 코드의 재사용은 보통은 성공하기 어렵다. 그래서 추상화 레벨에서 코드 대신에 모델의 재사용을 하는 것을 MDD라고 한다. 코드 생성기의 도움으로 시스템 코드는 자동적으로 생성된다. 이 MDD는오래 가는 것을 기대할 수 있고 개발자 사이의 의사소통을 가능하게 하고 성숙한 소프트웨어 프로세스에서 일반적으로 개발되어 진다. 또한 이것은 디자인 단계에서 개발, 재사용, 유지, 진화가 제안된다.

2.2 Apriori 알고리즘

연관 규칙을 찾기 위해 트랜잭션 데이터베이스의 모든 부분집합에 대한 지지도를 계산하여 각각의 빈도수를 조사하고 그에 따른 빈발항목을 찾아야 한다. 그러나 트랜잭션 수가 많을 경우 부분집합은 그것에 비례하여 지지도의 계산 및 비교가 쉽지 않았다. 그래서 그것을 줄이기 위한 방법으로 모든 가능한 항목집합을 줄이는 방법, 비교하는 수를 줄이는 방법으로 연구가 진행되었다.

1994년 Agrawl과 Srikant가 제안한 Apriori 알고리즘은 모든 가능한 항목 집합을 줄이는 방법으로 부울리아 연관규칙으로 빈발항목집합(frequent itemset)을 분석해내기 위한 가장 기본적인 알고리즘이다. 그리고 이것은 반복적 상향식(bottom up) 접근 방법이고 효과적인 항목집합(itemset)의 후보자(candidate)를 계산하기 위해 너비우선탐색(breadth-first search)와 해싱 트리 구조를 사용한다.

Apriori 알고리즘에서는 “첫째, 하나의 항목집합이 빈발항목집합이면, 이 항목집합의 모든 부분집합도 빈발항목집합이다. 둘째, 하나의 항목집합이 빈발항목집합이 아니면 이 항목집합을 포함하는 모든 집합은 빈발항목집합이 아니다.”란 2가지 법칙이 전제로 되어 있다. 그리고 이 알고리즘은 조인(Join)과 가치치기(Prune) 과정을 통해 해싱트리 구조를 유지하면서 지지도에 따라 빈발항목집합을 선별한다.

Apriori 알고리즘을 표현하면 다음과 같다.[5]

$$
\begin{aligned}
&Apriori(T,\epsilon)\\
&\quad L_1 = large\ 1-itemsets\\
&\quad \text{for}\,(k=2; L_{k-1}\neq 0; k{+}{+})\ do\ begin\\
&\qquad C_k = apriori-\geq n(L_{k-1});\\
&\qquad forall\ \ transactions\ \ t\in\ T\ \ do\ begin\\
&\qquad\quad C_t = subset\ \ (C_k, t);\\
&\qquad\quad forall\ \ candidates\ \ c\in C_t\ do\\
&\qquad\qquad c.count{+}{+};\\
&\qquad\quad end\\
&\qquad L_k = \{c\in C_k | c.count \geq \epsilon\}\\
&\qquad end\\
&\quad Answer = \bigcup_k L_k;
\end{aligned}
$$

T : 트랜잭션 데이터베이스
ϵ : 지지도 한계값
L : 빈발항목집합
C : 빈발항목집합후보자

이 때, 지지도(Support), 신뢰도(Confidence), 향상도(Lift)에 따라 빈발 부분항목집합의 결과가 다르게 나타난다. 규칙이 $A \Rightarrow B$라고 할 때 지지도, 신뢰도, 향상도를 식으로 나타내면 <formula 1>과 같다.

$$\text{지지도}(Support) = \frac{freq.(A,B)}{N} \quad (1)$$
$$\text{신뢰도}(Confidence) = \frac{freq.(A,B)}{freq.(A)}$$
$$\text{향상도}(Lift) = \frac{Support}{Supp(A) \bullet Supp(B)}$$

3. 제안된 모델

3.1 데이터 세트의 선정

공공데이터 세트 중에 소프트웨어 성공과 실패를 예측하기 위한 데이터 세트를 선정하였다. 이것은 소프트웨어 재사용에서 성공과 실패요인들로 구성되어 있는 데이터 세트로 26개의 입력 column은 표 1과 같다.

3.2 제안 모델 세부 내용

제안된 모델은 선정된 데이터 세트의 데이터를 정제하여 Apriori 알고리즘을 적용한 후 연관규칙 기준에 따른 연관성을 분석하는 모델이다. 이러한 모델을 도식화하면 그림 2와 같다.

표 1. Software reuse dataset's input column
Table 1. Software reuse dataset's input column

No.	input column	value
1	Software Staff	L, M, S
2	Overall Staff	L, X, M, S
3	Type of Software Production	product-family,isolated
4	Software and Product	product,alone,process,NA
5	SP maturity	high,middle,low
6	Application Domain	TLC,SE-Tools,Bank,Engine_Controller,FMS,ATC,TS,Space,Manufacturing,Measurement,Finance,Book-Keeping
7	Type of Software	Technical,Business,Embedded-RT,Non-Embedded-RT
8	Size of Baseline	L,M,S,not_available
9	Development Approach	OO,proc,not_available
10	Staff Experience	high,middle,low,not_available
11	Top Management Commitment	yes,no
12	Key Reuse Roles Introduced	yes,no,NA
13	Reuse Processes Introduced	yes,no,NA
14	Non-Reuse Processes Modified	yes,no,NA
15	Repository	yes,NA
16	Human Factors	yes,no
17	Reuse Approach	tight,loose,NA
18	Work Products	D+C,C,R+D+C,NA
19	Domain Analysis	yes,no,NA
20	Origin	ex-novo,as-is,reeng,NA
21	Independent Team	yes,no,NA
22	When Assests Developed	before,justintime,NA
23	Qualification	yes,no,NA
24	Configuration Management	yes,no,NA
25	Rewards Policy	no,yes
26	assests	51_to_100,21_to_50,100+,1_to_20,NA

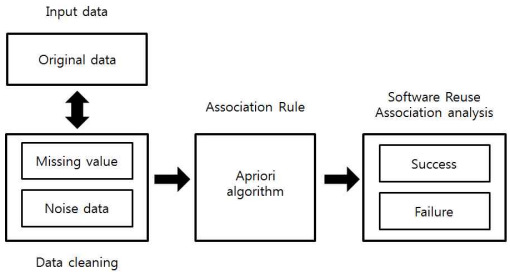


그림 2. 제안된 모델
Fig. 2. Suggested model

3.2.1 데이터 정제

데이터가 불완전하고 노이즈가 있으면 정확한 결과를 도출하기 힘들다. 그래서 데이터 정제는 결측치에 대한 것을 채우고 노이즈를 제거하여 일관성있는 데이터를 만드는 작업이다.

결측치를 회귀분석, 베이지안 공식 등으로 결측값을 예측하는 경우도 있지만 선정한 데이터는 추론이 어려운 경우를 포함하고 있기 때문에 선정한 데이터세트의 결측치의 값은 일괄적으로 NA값으로 채우도록 한다.

3.2.2 Apriori 알고리즘 적용 및 분석

기본적인 연관 규칙을 찾는 방법은 모든 빈발항목 집합을 발견하여 그중에 지지도, 신뢰도를 만족하는 강한 규칙을 찾아내는 것이다. 그러나 Apriori 알고리즘은 앞서 설명한 “하나의 항목집합이 빈발항목집합이면, 이 항목집합의 모든 부분집합도 빈발항목집합이다”는 전제되는 규칙에 의해 빈발항목집합의 개수를 줄여서 하는 방법이다. 그래서 선정된 데이터 세트에 Apriori 알고리즘을 적용하여 소프트웨어 재사용 성공을 위한 연관 규칙을 찾아내고 그에 따라 분석을 한다.

3.2.3 연관 규칙 기준 및 분석

Apriori의 연관 규칙을 생성하기 위해서는 지지도, 신뢰도, 향상도의 값을 조정한다. 이러한 연관규칙을 어떤 조건에서 생성함에 따라 각각의 데이터세트에서 나오는 실험결과가 달라지고 그에 따른 입력 factor의 연관성도 달라진다. 따라서 지지도, 신뢰도의 변경에 따른 결과를 비교하고 그에 따른 연관성을 분석한다.

4. 실험 결과

4.1 실험 개요

소프트웨어 재사용 성공과 실패 예측을 위한 데이터 세트의 데이터 정제 후에 Apriori 알고리즘을 지지도와 신뢰도의 값을 0.1 단위로 변경하여 규칙의 수의 차이가 나타났다. 지지도와 신뢰도에 따른 연관 규칙수의 차이는 표 2와 같다.

지지도와 신뢰도가 높을수록 규칙의 수는 줄

표 2. The number of rules according to support and confidence value
Table 2. The number of rules according to support and confidence value

reuse success	Confidence									
Support	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.1	-	-	-	-	-	-	-	-	-	-
0.2	-	-	-	-	-	-	-	-	-	-
0.3	4,616,989	-	-	-	-	-	-	-	-	-
0.4	2,388,813	2,388,813	2,388,813	2,388,813	2,382,226	2,342,360	2,258,885	2,152,084	1,958,498	1,927,278
0.5	351,795	351,795	351,795	351,795	351,795	350,259	340,091	328,139	296,168	264,948
0.6	213,552	213,552	213,552	213,552	213,552	213,552	212,465	206,803	190,353	159,133
0.7	58,715	58,715	58,715	58,715	58,715	58,715	58,715	58,139	54,877	45,645
0.8	28,639	28,639	28,639	28,639	28,639	28,639	28,639	28,639	26,271	23,521
0.9	2,560	2,560	2,560	2,560	2,560	2,560	2,560	2,560	2,560	2,304
1	192	192	192	192	192	192	192	192	192	192

어들었다. 그리고 지지도가 1일 때에는 신뢰도와 무관하게 연관 규칙수가 192개로 동일하게 나타났다.

소프트웨어 재사용을 위한 데이터 세트에 Apriori 알고리즘을 적용하여 지지도가 1일 때 나온 규칙들은 그림 3과 같다.

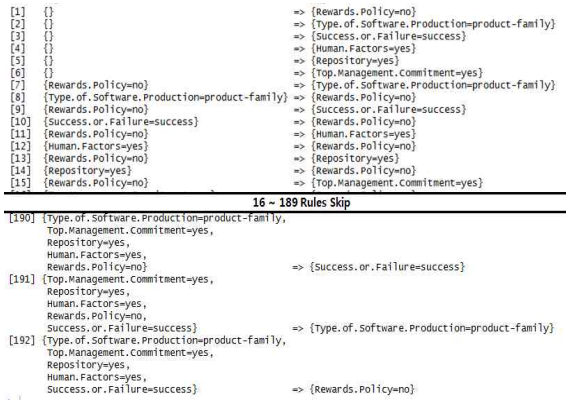


그림 3. 연관 규칙
Fig. 3. Association rules

4.2 실험결과 분석

그림 3에서 나온 192개의 규칙을 토대로 쉬운 분석을 위해 소프트웨어 재사용 성공을 위해 영향을 주는 요인을 알파벳으로 표현하면 표 3과 같다. 그리고 192개의 연관 규칙을 분석하면 표 4와 같은 결과가 나왔다. 이것은 6개의 Column이 서로에게 영향을 주고 그것들 사이에 연관성이 있다는 것을 보여 준다. 그리고 소프트웨어 재사용은 Rewards Policy = no, Type of Software production = product-family, Human Factors = yes, Repository = yes, Top Managemanet Commitment = yes일 때 Success or Failure = Success로 나타났다.

표3. The alphabet of factor
Table 3. The alphabet of factor

No.	Factor
A	Human Factors = yes
B	Rewards Policy = no
C	Success or Failure = Success
D	Repository = yes
E	Top management comminment = yes
F	Type of software production = product-family

표 4. The relation of six columns
Table 4. The relation of six columns

LHS	RHS
B,C,D,E,F	A
A,C,D,E,F	B
A,B,D,E,F	C
A,B,C,E,F	D
A,B,C,D,F	E
A,B,C,D,E	F

5. 결 론

소프트웨어 재사용이 비용의 효율성과 품질을 향상시킴에도 불구하고 기존의 소프트웨어에 대한 전반적인 이해 없이는 재사용을 성공시키기 어렵다. 그리고 소프트웨어 재사용은 어떻게 무엇을 사용할지에 대한 요소 선정 및 결정 부분에 대한 기준 문제, 표준화 문제, 소프트웨어와 언어의 이질성에 따른 문제, 소프트웨어 요소 내부와 인터페이스의 이해, 유지 보수 관리 등 어려가지 문제를 가지고 있다.

이에 본 연구에서는 성공적인 소프트웨어 재사용을 위해 장바구니 분석에 활용되는 Apriori 알고리즘을 적용하여 연관성 분석 모델을 제안하였다. 그래서 연관 규칙을 토대로 분석한 결과 소프트웨어 재사용 성공을 위한 6개의 column을 찾아 낼 수 있었고 그것들의 연관성을 분석할 수 있었다.

향후에는 다른 방법론을 적용하고 Apriori의 효율을 개선하여 모든 지지도와 신뢰도를 가지고 제안한 모델을 여러 데이터를 적용하여 비교할 것이다.

참 고 문 헌

[1] R. Prieto-Diaz, “Status Report : Software Reusability”, IEEE Software, Vol.10, No.3, pp.61-66, 1993.

[2] J.C.S.P. Leite, Y. Yu, L.Liu, E.S.K. Yu, J. Mylopoulos, “Quality-based Software Reuse”, CAiSE, 2005.

[3] W. Krueger, “Software Reuse”, ACM Survey, 1992.

[4] L. Sommerville, “Software Engineering 9th”, PearsonEducationKorea, 2011.

[5] R. Agrawal, R. Srikant, “Fast Algorithms for Mining Association Rules”, VLDB Conference, 1994.

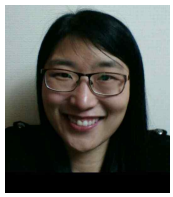
[6] R. Srikant, R. Agrawal, “Mining Generalized Association Rules”, VLDB, 1995.

[7] T. Mitchell, “Machine Learning”, McGraw-Hill, 1997.

저 자 소 개

권기태(Ki-Tae Kwon)

1986년 서울대학교 계산통계학과 학사
1988년 서울대학교 계산통계학과 이학석사
1993년 서울대학교 계산통계학과 이학박사
1996년 Univ. of Southern California, Post-Doc.
<주관심분야 : 소프트웨어 비용산정, 소프트웨어 메트릭스, 소프트웨어 아키텍처, 데이터 마이닝 등>



이나영(Na-Young Lee)

2003년 강릉원주대학교 컴퓨터공학과 졸업
2005년 강릉원주대학교 컴퓨터교육 교육학석사
2015년~현재 강릉원주대학교 컴퓨터공학 박사 수료
<주관심분야 : 소프트웨어 품질, 데이터마이닝 등>