http://dx.doi.org/10.5392/JKCA.2012.12.01.059

변경관리에서 ANP기법을 이용한 컴포넌트 선택 결정 방법

Component Selection Decision Method Using ANP Technique in Change Management

김경훈, 송영재
경희대학교 컴퓨터공학과

Kyoung-Hun Kim(magiclamp@khu.ac.kr), Young-Jae Song(yjsong@khu.ac.kr)

요약

소프트웨어 변경관리는 시스템의 변경된 내용을 프로그램이나 설명문서와 같은 특정 개체의 특성 변경에 초점을 둔 것이다. 변경관리 시 요구사항간의 상호종속적인 관계를 가지고 최적의 상태를 위하여 복잡한 의사결정을 필요로 한다. 본 논문은 소프트웨어 변경관리를 분산환경에서 컴포넌트들간에 시간과 상황에 따른 변화를 관리하는 모델을 설계 한다. 그리고 각 컴포넌트들간의 관계성들에 대한 정의를 하고 ANP 기법을 이용하여 분산환경에서의 각 컴포넌트가 변화되어 참조되는 상호 의존성을 고려하여 종속관계와 피드백을 이용하여 최적의 대안을 선택할 수 있다. 즉, 서로간의 관계된 의존도를 분석하여 3가지 형태의 변경관계를 나타내도록 하였다. 또한 의존도 분석을 통해 이러한 접근 방법의 유효성을 검증하였다.

■ 중심어 : | 소프트웨어공학 | 변경관리 | 네트워크분석기법 | 컴포넌트 |

Abstract

Software change management is focused on the change of a entity like the change of contents of a system or a document. In change management, interactive relationship among requirements and complex decision making is needed to obtain optimized status. In this paper we design a management model of software change management in distributed environment which mange the change among components by time and situation. In addition, each components are defined and use ANP technique for best decision-making by using the subordinate relationship and feedback considering the mutual dependency referring the change of components in distributed environment. Thus, we analyze the dependency among each components and show 3 types of change relationship. Also through analysis of dependency, we verified the effectiveness of such approach.

■ keyword : | Software Engineering | Dhange Management | ANP | Component |

Ⅰ. 서 론

소프트웨어 개발 프로세스에서 절대적으로 필요한 변경관리(Change control)는 개발 후반부에 수행하여 개발에 필요한 컴포넌트를 관리할 수 있다. 또한 컴포넌트 단위에 개발 환경에서는 더욱 복잡하고 많은 버전이 존재하며 개발 시 이용될 컴포넌트의 선정에 유효한 방법을 제공하지 못하고 있다[1].

변경관리는 시스템내에 개체들의 특성변경과 관계성들에 함축된 의미 변화를 관리한다. 이는 시스템에서

접수번호 : #111118-003
접수일자 : 2011년 11월 18일
심사완료일 : 2011년 12월 12일
교신저자 : 김경훈, e-mail : magiclamp@khu.ac.kr

다양한 유형의 개체와 관계성이 존재하며 이들의 특성 및 의미는 시간에 흐름이나 사용자 그리고 환경에 따라 빈번하게 변화된다. 즉 사용목적에 따라 다른 컴포넌트들 간에 결합이 존재한다[2].

일반적인 파일-기초의 버전 컨트롤은 소스 코드에 중점을 두었기 때문에 확장 시 필요한 모듈을 선택하기가 매우 어렵다. 가시성이 제공 되지 않기 때문에 상호작용에 대한 평가가 어려운 실정이다. 그리고 의존도 평가 연구들은 마르코프 모델을 사용하였으나 현재의 분산 환경에는 부적합하다[3][4].

Fine-grained 데이터 모델을 사용한 변경관리는 UML 다이어그램에 요소들로 계층을 표시하여 관리한다. 계층간에 상속, 결합 관계 등을 표현하지만 구조를 분석하는데 많은 노력이 필요하여 새로운 변경에 따른 변화에 적응 못하는 단점이 있다[5][6].

Unified CM Model에서는 삼차원 트리 구조를 사용하여 시간 포인트를 이용한 기법을 사용하고 있다. 문서중심에서 산출물을 관리함으로서 구조화된 객체에 적합하지 않은 단점을 가지고 있다. 즉, 파인-그래인드와 마찬가지로 뷰-기반에 선별 매커니즘에 대한 부분이 부족한 실정이다[7].

NUCM(Network-unified configuration management)에서는 분산환경에 적합한 관리기법을 제공하고 있다. 모델을 구성하여 새로운 변경에 대한 트리 구조를 사용함으로써 추상계층의 특징을 가지고 있다. 그러나 추상계층에서의 관리로 인해 변경에 대한 선택적 상황에 접근하지 못하는 단점이 있다[8].

본 논문은 분산 환경에서 존재하는 많은 컴포넌트에서 가장 많은 참조를 한 요소를 2단계의 깊이로 찾는 방법에 ANP(Analytical Network Process)기법을 적용하였다[9]. ANP기법은 각 계층 간 또는 같은 계층 안에서 상호의존성(Inner-Dependence)이 존재할 경우, 상호 종속관계와 피드백을 포함하는 복잡한 환경하에서 더욱 정확한 접근을 통해 최적의 대안을 제시할 수 있는 모델이다. ANP 방법으로 전체적인 모델을 구성하여 변경 관계를 가시적으로 표현하였다[10][11].

본 연구에서는 ANP기법을 이용하여 컴포넌트 사이의 관계를 정의하여 모델을 구성하고, 그 요소들 간의 의존도를 재 정의하여 명세함으로써 새로운 버전 생성 시 변경할 수 있는 컴포넌트를 선택 할 수 있는 방법을 제시하고자 한다.

제2장에서는 기존 변경관리에 대한 방법과 비교 설명하고, 제3장에서는 분산 환경에 대한 방법을 제시하여 의존도 및 가중치에 대한 방법을 제시하고, 제4장에서 제안한 방법으로 결과를 도출하였으며 제5장에서는 결론 및 향후과제에 대한 논의를 하였다.

즉, 컴포넌트에 변경에서 3가지 형태에 모델을 생성하여 제시함으로써 관리자가 선택적으로 사용할 수 있는 방법을 제시 하고자 한다.

II. 변경관리 기법에 대한 고찰

2.1 변경 관리

소프트웨어 변경관리(Software change Management)란 시스템에 포함한 소프트웨어의 일부분을 변경할 필요가 생겼을 때 전체 시스템에 대한 제어를 하는 것이다. 현재의 분산 환경에서는 각 컴포넌트의 변화가 생기거나 재결합이 필요 시 컨트롤 하는 것으로 진화되었다. 전통적으로는 프로그램, 문서, 데이터베이스등이 포함한 소프트웨어 시스템 개발 관리 문제로 다루워져왔다. 그러나 분산 환경에서는 각각의 컴포넌트들에 대한 정보와 결합조건 그리고 일관성에 대한 구성기법이 필요하다.

소프트웨어 형상항목들의 식별, 변경, 통제, 상태추적 등을 위한 소프트웨어 형상관리(Software Configuration Managements) [12], 시스템에서 데이터 타입과 스키마의 일관성유지하는 스키마변경[13], 그리고 모델 변경 등이 변경관리에 예이다. 또한 버전통제는 히스토리를 유지하여 재구성하기 위한 기법으로 사용된다[14][15]. 변경관리는 공통적으로 전체시스템에서 발생된 각각의 컴포넌트에 대한 갱신, 결합, 재사용 등의 모든 과정들을 포함한다. 즉, 두 컴포넌트간의 관계성이 정의된 의미나 의존관계에 대한 특성으로 관리할 필요가 있다.

변경관리의 식별과 영향에 대한 관계를 정의하고 의존관계를 명확히 하여 변경작업에 상호연관성을 사용

하는 방법이 필요하다. 두 컴포넌트간의 특성 변경에 따른 관리를 위해 서로간의 상호의존성에 대한 평가가 필요하다.

또한, 두 컴포넌트 A와 B의 요소들 간의 상호의존성을 가지고 새로운 변경작업을 위한 선택적 요구가 필요하다. 이러한 선택적 요구를 위해 ANP기법을 이용하여 실제 두 컴포넌트 요소간의 참조되는 의존도를 계산하여 두 컴포넌트에 변경 시 결합 형태를 제시할 수 있도록 하였다.

2.2 기존 변경 관리

Fine-grained 기법에서는 그래프 모델을 통한 관리 방식은 비 순환 방향성 그래프를 이용하여 노드를 통해 변화되는 과정을 관리한다. 즉, 노드의 종류를 세분화하여 각각의 역할을 부여하고 세부적인 모델과 함수를 통해 각각의 기능을 부여한다[5].

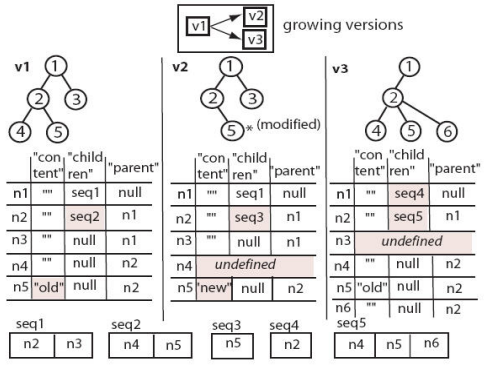


그림 1. 세부적인 관리 기법

[그림 1]은 세부적으로 변화되는 노드를 명세하고 있다.

세부적인 관리 기법은 실제 의존도에 대한 정보를 단순히 XML로 표기만 할 뿐 실제 결합되는 컴포넌트간에 기준점을 제시하지 못하고 있다. 따라서 각 컴포넌트와 요소들간의 의존성에 가중치를 부여하여 서로간에 참조되는 요인을 가지고 기준점을 만들어야 한다.

컴포넌트간에 관계를 그래프로 표현하여 클래스는 노드로 의존정도는 에지(edge)값으로 하는 방향 그래프(directed graph)를 사용할 수 있다.

[그림 2]는 노드 상호간 복잡성의 척도를 표현한 것이다. 그러나 클래스간 입력-출력에 대한 내용은 표시하고 있으나 서로의 변경 시 발생된 관계 정보는 나타내지 못하고 있다[5][6].

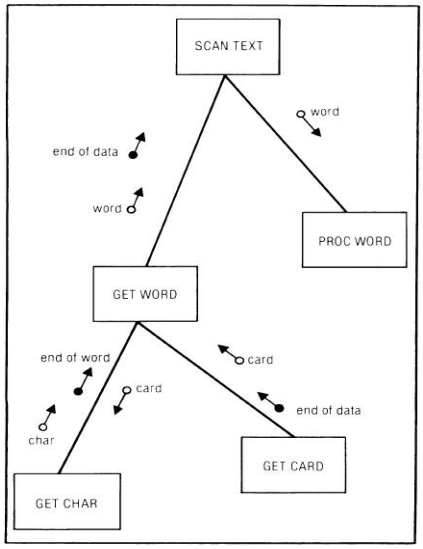


그림 2. 클래스간의 입출력에 대한 의존 복잡도

절차 지향적 개발 방식에서 객체 지향에 의존한 개발 방법은 많은 단계의 버전을 생성하고 많은 문서자료를 만들어 사용한다. 또한 전형적인 SCM툴들은 현재 작업중인 문서와 일치하는 중간단계의 버전을 저장하는 것을 지원하지 않고 다른 개발자가 참고할 수 있는 지속적이고 수평적인 버전의 명세만을 제공한다. 즉, 개발자(관리자)측면에서 실제 개발에 필요한 컴포넌트들 간의 변경된 관계에 관한 정보를 제공하기 어려운 단점이 있다[6][8].

또한 변경관리에서 변경된 정보의 일반적인 저장을 통해 트리 구조 형식으로 컨트롤하여 명세하고 있다[2][8]. 단순한 명세만이 가능하므로 변경정보에 대한 명세와 의존도의 대한 정보가 필요하다

또한 현대 소프트웨어 개발 운영 방식은 컴포넌트의 협동과 서브형태의 계층구조화 방식으로 관리하고 있어 계층적 구조를 지원하는 저장소는 중간 단계에 대한 제한된 변경관리를 제공해야 한다[18].

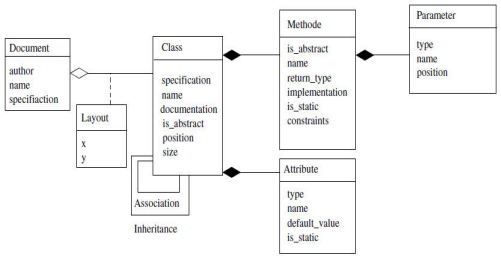


그림 3. 세분화된 데이터 모델

[그림 3]는 세분화된 데이터 모델에서 모든 요소들을 분리하여 설계 되어 모든 관계들의 복잡성이 증가하여 새로운 객체의 의존도와 명세에 대한 요구가 더 필요하게 된다. 즉 세분화의 정도도 기준선을 잡아서 각 컴포넌트에서 객체와 멤버함수까지의 의존도와 명세를 추가하여 모델을 설계하는 것이 이상적이다[16].

변경된 관계를 계층항목으로 평가할 경우 [그림 4]처럼 나타낼 수 있다. 계층적으로 변화된 정보를 바뀐 버전에 대한 간단한 명세로만 표현 될 수 있다[17].

또한 일관성 지수(Consistency Index : CI)가 0.2 이하로 나오면 일관성이 있어서 신뢰할 수 있다[5].

그리고 현재 분산 환경시스템에서는 식별기능만을 제공한다[8].

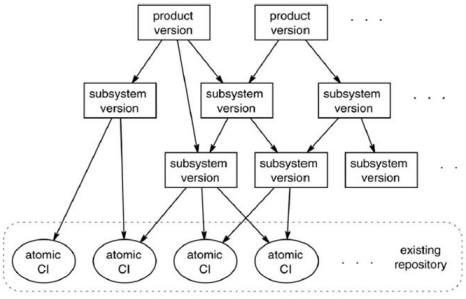


그림 4. 계층적 구조의 형상항목

Ⅲ. ANP 기법을 이용한 관계 모델 설계

3.1 분산환경에서 컴포넌트

본 연구에서는 서로 다른 컴포넌트 간 상호 참조 관계를 모델링하고 그 참조관계의 의존 정도를 정량적으로 설계하기 위해 ANP(Analytic Network Process) 기법을 이용하였다. ANP 기법은 가장 다 기준 의사결정 기법 중 하나로써, 각 요소간 상호의존성(Inner-Dependence)이 존재할 경우 상호 종속관계와 피드백을 고려하여 최적의 대안을 정량적으로 제시할 수 있는 모델이다[10][11].

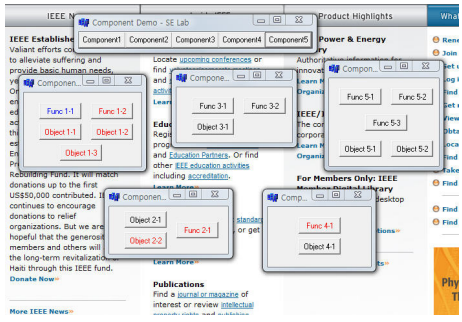


그림 5. 컴포넌트의 의존관계

[그림 5]는 윈도우 환경에서 컴포넌트들에 의존관계를 보여주고 있다. 실제 컴포넌트에서 다른 컴포넌트의 요소를 참조하는 모습과 실제 요소의 값을 선택적으로 보여준다.

실제로 컴포넌트에서 다른 컴포넌트의 요소에 참조하는 내용을 분석하였다.Component1의 Func1-1이 참조하는 상태를 보면 자기 자신의 Func1-2와 Object1-1, Object1-2 그리고 Object1-3을 참조한다. 또한 다른 컴포넌트의 Func2-1, Object2-2와 Func4-1을 참조하고 있다. 이러한 관계를 [그림 7]에 알고리즘을 이용해서 [그림 8]의 의존관계 테이블에 적용하여 각각의 컴포넌트와 요소간의 의존관계 정도를 알 수 있다.

또한 컴포넌트1의 Func1-2 인 경우에는 Func1-2 과 Object1-2 그리고 컴포넌트2의 Object2-1 그리고 Func2-1 과 컴포넌트4의 Func4-1을 참조한다. 이와 같이 모든 컴포넌트와 그 요소의 참조를 가지고 의존도를 측정하여 테이블을 작성하여 변경관리 시 필요한 정보를 추출할 수 있다.

[그림 6]은 컴포넌트간에 참조되는 형태를 표현해주는 코드이다. 코드를 통해 컴포넌트간에 참조하는 요소와 참조 받는 요소를 그래픽적으로 표현해준다.

```
__fastcall TComponent1::TComponent1(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TComponent1::Init(TSpeedButton *OBJ) {
  Main_Component->Clear();
  OBJ->Font->Color = clBlue;
}

void __fastcall TComponent1::Call(TSpeedButton *OBJ) {
  OBJ->Font->Color = clRed;
}

void __fastcall TComponent1::Func1Click(TObject *Sender)
{
  Init((TSpeedButton *)Sender);

  Call(Func2);
  Call(Object1);
  Call(Object2);
  Call(Object3);
  Call(Component2->Func1);
  Call(Component2->Object2);
  Call(Component4->Func1);
}
```

그림 6. 참조관계 코드 분석

3.2 의존도 측정

윈도우환경에서 5개의 컴포넌트 내에 있는 9개 함수와 9개 오브젝트들 간의 상호 참조관계를 측정 하였다.

[그림 7]은 컴포넌트간 의존도를 측정하기 위한 알고리즘으로써, QFD(Quality Function Deployment) 기법과 결함율을 이용한 시그마 계산법을 이용하여 구현하였다[9].

```
// 열과 행의 요소의 영향을 주는지 입력
for(j=0; j<n ; j++)
    for(i=0 ; i<n ; i++)   //n : 요인수
    {
        if(i==j)
            Cij = 1;
        else
            Cij = Cij + input;  //행의 요소에 대해
                                //열마만큼 영향을 주는지 입력
        cnt++;                  //카운팅
    }

for(j=0; j<n ; j++)
    for(i=0 ; i<n ; i++)
        avr_Cij = Cij/cnt;     //표본에 대한 평점 계산

// 표본의 평점의 구간이 50% 이하인 값은 제거
for(j=0; j<n ; j++)
    for(i=0 ; i<n ; i++)
        if(Cij <= theshold)
            Cij
```

```
// 열과 행의 요소의 영향을 주는지 입력
for j = 0 to n
  for i = 0, to n
    if i==j , c=1
        not c=c+input

  count++

// 표본에 대한 평점 계산
for j = 0 to n
  for i = 0 to n
      average

// 표본의 평점의 구간이 50% 이하인 값은 제거
for j=0 to n
  for i=0 to n
    if c<theshold is delete
```

그림 7. 컴포넌트간 의존도 측정 알고리즘

[그림 7]의 알고리즘을 통해 18개 요소간 상호 의존 정도를 분석하였다. 정밀한 분석을 위해 2단계의 깊이로 값을 계산하였다.

Components	Factors	Component1					Component2			Component3			Component4		Component5				
		Func1-1	Func1-2	Object1-1	Object1-2	Object1-3	Func2-1	Object2-1	Object2-2	Func3-1	Func3-2	Object3-1	Func4-1	Object4-1	Func5-1	Func5-2	Func5-3	Object5-1	Object5-2
Component1	Func1-1		O	O	O	O	O		O				O						
	Func1-2	O			O		O	O					O						
	Object1-1													O					
	Object1-2					O		O	O				O	O					
	Object1-3			O	O			O						O					
Component2	Func2-1																		
	Object2-1																		
	Object2-2																		
Component3	Func3-1						O												
	Func3-2							O											
	Object3-1							O											
Component4	Func4-1																		
	Object4-1																		
Component5	Func5-1						O					O					O	O	
	Func5-2							O	O						O				O
	Func5-3							O			O				O				O
	Object5-1															O	O		O
	Object5-2															O	O	O	

그림 8. 각 컴포넌트간의 의존관계 테이블

[그림 8]에서 보는 바와 같이 5개의 컴포넌트 내에 있는 각각의 요소간 의존도를 정방 행렬로 표현하였다. 자기 자신을 제외한 행의 요소가 열의 요소로부터 영향을 받을 경우, 즉 의존 관계가 발생한 경우 'O'으로 표시하였다.

3.3 의존관계 모형

[그림 9]는 의존도측정 알고리즘과 그림 8의 의존관계 테이블 자료를 바탕으로 컴포넌트와 내부 요소간 의존 관계를 네트워크 모형으로 설계하였다.

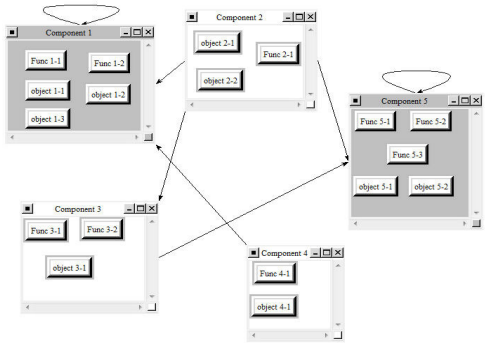


그림 9. 의존관계 모형

[그림 9]에서 보는 바와 같이, 컴포넌트와 컴포넌트가 포함하고 있는 함수와 오브젝트들이 상호 의존관계를 형성하고 있음을 확인할 수 있다. 특히, 컴포넌트1은 과 컴포넌트2와 컴포넌트4로부터 영향을 받고, 컴포넌트5는 컴포넌트2와 컴포넌트4로부터 많은 영향을 받는 것으로 분석되었다.

3.3.1 의존성요인

명명척도는 항목을 분류하고 서열척도로 개별 항목들을 분류한다. 간격척도는 서로간의 1~9단계로 분류하고 비율척도는 비율적 요소를 사용 한다. 서열척도를 사용하여1~9의 의존도로 나누는 방법을 사용한다.

의존 항목간 척도(scale)의 범위는 "1~9"을 사용한다. 또는 역수를 이용한다.

$$x^{-1}x = xx^{-1} = 1 \qquad \text{식1.}$$

의존성 요인을 가지고 노드간의 의존도를 이용해 가중치를 계산하여 의존도를 계산한다.

3.3.2 가중치 계산

선택적 순위 계산을 위한 가중치는 기준 i의 중요도를 C_i (i=1, …, n)이라 하고, 기준 i에 따른 대안 j의 중요도를 P_{ij} (j=1, …, m)라 하면 대안 j의 상대적 중요도의 크기 R_j는 다음 식과 같다.

$$R_i = \sum_{i=1}^{n} C_i P_{ij}^{i=0} \qquad \text{식2.}$$

3.3.3 쌍대비교 시 중요도의 척도

의존성 요인과 가중치 계산의 값을 가지고 쌍대 비교를 통해 하나의 요소가 다른 요소에 대하여 갖는 상대적인 우월성의 값을 이용한다. 즉, 각 컴포넌트간 요소간의 쌍대비교를 통한 데이터를 수집한다.

표 1. 쌍대비교 시 중요도의 척도

중요도	정의	설명
1	비슷함	동일함
3	약간 중요함	약간 선호됨
5	중요함	선호됨
7	매우 중요함	확실히 선호됨
9	극히 중요함	극히 선호됨
2,4,6,8	위 값들의 중간값	위 값들의 중간값

3.3.4 일관성의 측정

고유값(Eigen Value)을 이용하여 상대적 가중치의 값을 측정한다. 즉, 쌍대비교(Paired-comparision)를 통해 얻은 값은 2개의 항목만의 가치를 비교하여 얻었기 때문에 전체적인 일관성을 유지하고 있는가를 평가해야 한다. 쌍대비교 행렬표와 이로부터 얻어진 가중치값으로

$$A_v = \lambda_{\max} v \qquad \text{식3.}$$

관계식을 유도하여 측정한다.

A는 쌍대비교행렬이고 는 A의 최대 고유치이다. 그리고 는 고유벡터이다.

행렬 A에는 n개의 고유치가 있고, 그 합 λ1 + λ2 + • • • + λmax 은 n 이 된다.

$$CI = \frac{\lambda_{\max} - n}{n-1} \qquad \text{식4.}$$

[그림 10]에 가중치 결과값에 의해 일관성 지수(Consistency Index : CI)가 0.2 이하로 나와서 신뢰성을 증명하였다.

그림 10. 의존도의 가중치 결과

3.4 의존관계에 따른 가중치 분석

본 연구에서는 [그림 9]의 의존관계 모형을 바탕으로 5개 컴포넌트와 컴포넌트에 포함된 9개 함수, 9개 오브젝트간의 의존 정도를 정량적으로 분석하였다.

[그림 11]은 각 요소별 의존관계를 정량적으로 분석한 결과이다.

[그림 11]에서 보는 바와 같이, 컴포넌트1과 컴포넌트5에 포함된 함수와 오브젝트들의 의존도 값이 높게 나타났다. 특히, 컴포넌트1의 Func1-1은 26%로 가장 높은 의존도를 나타냈으며, 두 번째로 컴포넌트5의

Func5-1이 13.8%의 의존도를 나타냈다.

이처럼 5개 컴포넌트와 컴포넌트 내에 포함된 9개 함수, 9개 오브젝트의 의존관계를 분석한 결과 컴포넌트1과 컴포넌트5의 의존도가 높게 나타났다. 그리고 이 두 개의 컴포넌트를 중심으로 컴포넌트2~4의 재설계가 가능할 것으로 판단된다.

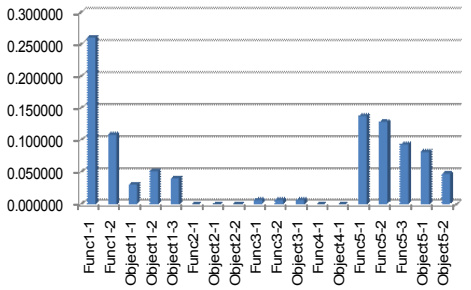


그림 11. 각 요소별 의존관계 지표

Ⅳ. 변경된 시스템에 3가지 형태와 평가

4장에서 각각에 컴포넌트들의 참조와 의존성을 이용하여 관계모형을 만들었다. 이 모델을 토대로 [그림 11]과 같은 3가지 형태에 변경사항에 대한 형태를 볼 수 있다.

1. A와 B가 서로 동등
2. A 가 B에 서브시스템
3. B가 A에 서브시스템

상용화된 시스템에서 필요한 실제 변경에 대한 서로간의 관계성에 대한 정의가 필요하다. 즉, 3가지 형태를 이용하여 변경관리에 프로토타입을 정의 할 수 있다.

정의된 프로토타입을 컴포넌트간에 변경이 생겼을 때 제시한 3가지 형태 중에 하나를 선택할 수 있는 메소드를 제공하여 두 컴포넌트의 변경처리를 할 수 있도록 한다.

재설계 시 컴포넌트의 서로간에 종속성을 선택하게 하여 다른 컴포넌트와 관계를 다시 정의하여 변경할 수 있는 방법을 제시하고 있다.

종속성이 강한 컴포넌트의 서브 컴포넌트로 관계를 정의하여 보다 빠르고 안정적인 서비스를 제공할 수 있다.

실제로 각각의 컴포넌트를 이용하여 새로운 프로젝트가 생성될 때 기존의 결합된 컴포넌트의 종속성을 이용하여 결합하여 보다 안정적이고 새로운 컴포넌트를 구성할 수 있다.

기존 방법에서 평가할 수 없었던 부분을 ANP을 적용하여 의존도 및 의존도 분석을 통해 실체화 하였다.

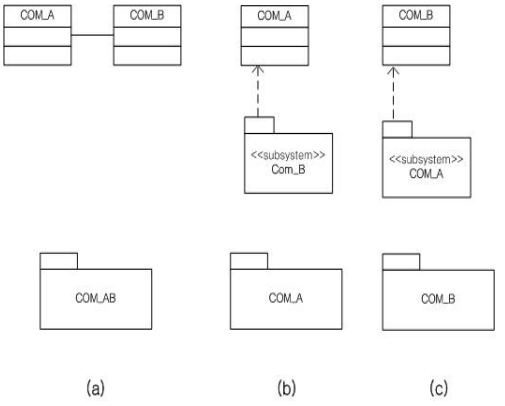


그림 12. 3가지 형태의 변경상태

Ⅴ. 결론 및 향후 연구 과제

소프트웨어 변경관리는 다양한 시스템에서 분산환경에 적합하도록 고려되어야 한다. 기존의 변경관리 시스템에서 구현의 복잡성 때문에 제한된 범위에서의 변경관리만이 가능하다. 본 연구는 분산 환경에서의 변경관리를 위한 컴포넌트의 선택 시 우선순위를 부여하여 3가지 형태를 제시하였다. 3가지 형태에 의존도분성에 ANP을 사용하여 의존도와 가중치를 계산하였고, 의존도가 높은 컴포넌트를 선택하여 안정적인 변경 관리를 할 수 있도록 하였다.

향후에는 변경관리를 위한 에이전트를 사용하여 보다 효과적으로 관리할 필요가 있다. 에이전트를 이용하여 분산환경과 복잡한 시스템에서도 관리하는 기법에 대해 연구가 필요하다. 향후에 에이전트가 각각의 컴포넌트를 관리하여 변경된 히스토리를 저장소에 보관하

는 방법에 대해 논의할 것이다. 현재 에이전트로 관리하는 방법에 대한 연구가 진행 중이다.

참 고 문 헌

[1] "소프트웨어 변경관리 노력 추정 모델", 정보기술논문지, 제5권, pp.27-33, 2007.

[2] 김덕현, 박성주, "확장된 객체지향 데이터 모형을 이용한 소프트웨어 변경관리 시스템", 한국정보과학회논문지, 제22권, 제2호, 1995.

[3] B. W. Jhonson, *Design and Analysis of Fault-Tolerant D12gital Systems*," Addison-Wesley, 1989.

[4] D. K. Pradhan, *Fault-Tolerant Computer System Design*, Prentice-Hall, 1996.

[5] Dirk Ohst and Udo Kelter, "A Fine-grained Version and Configuration Model in Analysis and Design," ICSM, IEEE, 2002.

[6] Tien N. Nguyen and Ethan V.Munson, John T.Boyland, "Multi-level Configuration Management with Fine-grained Logical Units", EUROMICRO-SEAA'05, 2005.

[7] Tien N. Nguyen, "A Unified Model for Product Data Management and Software Configuration Management," ASE'06, IEEE, 2006.

[8] Andre van Hoek, "A Testbed for Configuration Management Policy Programming," IEEE, 2002.

[9] 경태원, 김상국, "BSC 와 ANP기법을 이용한 직무그 룹별 정보시스템 우선순위 분석", 한국컨텐츠학회논문지, 제11권, 제7호, pp.426-436, 2011.

[10] 이선우, "AHP와 ANP를 이용한 ITS서비스의 우선순위결정에 관한 연구", 계명대학교, 2004.

[11] Saaty, T. L., *The Analytic Network Process, Pittsburgh,* RWS Pub, 1996.

[12] E. H. Bersoff, "Elements of Software Configuration Managements," IEEE Computer, pp.30-36, 1983.

[13] J. Banerjee, W. Kim, H. J. Kim, and H. Korth, "Semantics and Implementations of Schema Evolution in Object-oriented Database," Proceedings of ACM SIGMOD Conference, pp.311-322, 1987.

[14] D. H. Kim and S. J. Park, "FORM: A Flexible Data Model for Integrated CASE Environments," Working Paper, Department of Management Science, KAIST, 1992.

[15] K. Dittrich and A. Lorie, "Version Support for Engineering Database Systems," IEEE Transaction on Software Engineering, Vol.14, No.4, pp.429-437, 1988.

[16] Dirk Ohsh and Udo Kelter, "A Fine-grained Version and Configuration Model in Analysis and Design," Proceeding of the International Conference on Software Maintenance (ICSM'02) IEEE.

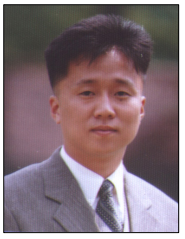
[17] M. Penedo, E. Ploedereder, and I. Thomas, "Object Management Issuse for Software Engineering Environments," Processing of 3rd ACMSIGSOFT, pp.226-234, 1988.

[18] Hagen Vo¨ lzer, Anthony MacDonald, Brenton Atchison, Andrew Hanlon, Peter Lindsay, and Paul Strooper, "SubCM: A Tool for Improved Visibility of Software Change in an Industrial Setting," 2007.

[19] Akao, Yoji, "Development History of Quality Function Deployment," The Customer Driven Approach to Quality Planning and Deployment. Minato-ku, Tokyo 107 Japan: Asian Productivity Organization, 1994.

저 자 소 개

김 경 훈(Kyoung-Hun Kim) 정회원



- 2000년 2월 : 삼육대학교 컴퓨터과학과(이학사)
- 2002년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2011년 현재 : 경희대학교 컴퓨터공학과 박사 수료

<관심분야> : 형상관리, 웹서비스, 의료시스템, 컨텐츠

송 영 재(Young-Jae Song) 정회원

- 1969년 2월 : 인하대학교 전자공학과(공학사)
- 1976년 : 일본 keio 대학교 전산학과 (공학석사)
- 1980년 : 명지대학교 전산학과 (공학박사)
- 1982년 ~ 1983년 : 미국 Maryland University 전산학과 연구교수
- 1989년 ~ 1990년 : 일본 Keio University 전산학과 객원교수
- 1976년 ~ 현재 : 경희대학교 컴퓨터공학과 교수

<관심분야> : 소프트웨어 재사용, CASE 도구, AOP, 요구공학, 컴포넌트웨어, 웹서비스