韓國經營科學會誌
第24卷 第4號
1999年 12月

유지보수성 목표하의 소프트웨어 개발방법 평가에 관한 실증연구 : ANP 기법을 중심으로

윤민석* · 이 영** · 성삼경***

An Analytic Network Process(ANP) Study on the Evaluation of Software Development Methods for Maintainability

Min-Suk Yoon* · Young Lee** · Sam Kyung Sung***

Abstract

Recently expanded and enhanced software system inevitably demands serious managerial considerations on its maintenance cost. This study proposes a systematic and thorough assessment model for software development methods to the end of maintainability, incorporating ISO 9126 quality standards-based criteria. The Analytic Network Process (ANP) is employed in our model to find the effect of any dependency among the items of the criteria and the alternative methods. This study then applies the model empirically in order to evaluate the efficacies of the Structured Method and the Objected-Oriented Method. At first, this study performs the test on the existence and direction of any dependency under the three types of assumptions, and then determines the relative dominance of the two methods.

1. 서 론

정보기술의 발달 및 사용의 확산에 따라 소프트웨어 시스템은 다양한 형태의 업무 처리 욕구를 수용하게 되면서 점점 대규모화되고 복잡해지고 있다. 이와 더불어 소프트웨어 개발 이후, 기능 또는 성능의 향상 및 새로운 환경으로의 적응 활동을 포함하는 유지보수의 어려움과 이의 해결을 위한 비용의 증가가 필수 불가결하게 수반되고 있다[5]. 소프트웨어 유지보수에 관련된 대부분의 문제점은

* 여수대학교 멀티미디어학부
** 소니 인터내셔널코리아(주)
*** 고려대학교 경영대학

소프트웨어가 계획되고 개발되는 방법이 적절하지 못하여 개발 과정과 유지보수 작업간의 연속성 및 일관성의 부재가 주요 원인으로 지적되고 있다[1]. 이러한 문제점의 해소에는 소프트웨어 개발 활동에 공학적 제어 및 원리가 요구되며 그 효과 여부는 사용된 개발방법과 관련되어 있다[22].

소프트웨어 개발 과정의 무결성(integrity)과 일관성 그리고 생산성을 제고하기 위한 체계적인 개발방법들이 대두되었는데 그 중 두드러진 두 가지는 구조적 방법과 객체지향 방법이다. 구조적 방법은 자료와 처리 절차를 구분하여 문제를 도식화하고 구조화하는 반면 객체지향 방법은 자료와 절차를 결합한 객체를 분석단위로 문제를 접근한다. 현실 모형화에 있어 두 개발방법의 사상적 기저의 차이는 소프트웨어 개발 전과정에 걸쳐 원리와 기법(technique)의 차이로 이어진다.

그러나 지금까지 두 개발방법 비교에 관한 연구들의 대부분은 각 방법의 기본 사상과 원리로부터 연역적 접근을 통한 논리적 비교에 치우쳐 있고 실증 연구는 상당히 미흡한 실정이다. 또한 비교를 위한 통일된 평가 기준 체계가 마련되어 있지 않으며 평가 기준간 존재하는 종속성을 반영하는 분석적 평가 모형은 전무한 실정이다. 이에 본 연구에서는 최근 중요성이 높아진 소프트웨어 유지보수 측면에서 개발방법의 우수성을 세심하고 정확하게 평가하여 계량화 할 수 있는 평가 모형의 제시 및 실증 연구를 목적으로 하며 다음과 같은 연구 내용을 포함한다.

첫째, 평가 목표 및 평가 기준의 설정이다. 이를 위하여 유지보수성의 정의를 도입하여 평가의 목표로 설정하고 이에 대한 평가 기준 체계를 마련한다. 평가 기준은 보다 객관성을 확보하기 위하여 ISO 표준을 따르기로 하며 본 연구의 목적에 부합하도록 체계화한다.

둘째, 평가 기법의 선택을 포함하는 평가 모형의 설계에 관한 것이다. 본 연구에서는 평가에 소요되는 자료를 소프트웨어 개발 전문가들의 주관적 판단에 의존하기로 하며 평가 모형에는 평가 기준 체계를 구성하는 요소들간 종속성을 반영하고자 한다. 이를 위하여 ANP (Analytic Network Process) 기법을 도입하기로 하고 가정되는 종속성에 따라 3가지 모형을 설계하여 비교한다.

셋째, 본 연구에서 제시한 평가 모형에 따른 실증 분석이다. 실증 분석 단계는 앞의 3가지 모형으로부터 산출되는 평가 기준 요소들의 중요도를 비교 분석하는 단계와 각 기준 요소들에 대한 소프트웨어 개발방법의 상대적 우수성을 평가하여 종합화하는 단계로 구성된다. 특히 기준 요소들의 중요도 비교 과정에서 요소들 사이의 종속성 존재 유무를 파악할 수 있다.

2. 개발방법 및 유지보수성의 고찰

2.1. 개발방법의 비교 고찰

일반적으로 소프트웨어 개발 단계는 요구사항 분석으로부터 설계, 프로그램 구현, 시험 그리고 유지보수의 프로세스로 구분한다. 전체 프로세스를 일관성 있게 연계시켜 각 프로세스를 구축케 하는 기술적 기본 원리와 기법의 집합이 개발방법이다[22].

1970년대부터 DeMarco[10], Gane & Sarson[13], 그리고 Yourdon & Constantine[33] 등에 의해 발전된 구조적 방법은 자료 흐름(data flow)을 자료 구조(data structure)로부터 구분하여, 자료가 처리되는 절차를 분석 및 설계의 대상으로 하는 처리 중심의 개발방법이다. Ward & Mellor[29]는 구조적 방법을 실시간(real-time) 시스템에 적합하도록 확장시킨 바 있다. 구조적 방법이 자료와 처리를 구분하는 데에 반해 이 둘을 결합한 객체를 분석단위로 하는 것이 객체지향 방법이다. 따라서 객체는 자료의 자체 처리 능력을 갖게되며 다른 객체와의 상호작용도 처리를 통하여 구현하게 된다. 객체지향 방법의 주요 연구자로는 Coad & Yourdon[9], Rambaugh[23], Jacobson[16] 그리고 Booch[8] 등이 있으며, 지금도 계속적인 연구와 함께 Ada 95, C++, Eiffel, Smalltalk 등의 구현 언어(language)도

개발되어 있다. <표 1>에 Fichman & Kemerer[12]와 Pressman[22]의 연구를 종합하여 구조적 방법과 객체지향 방법의 주요 대비점을 제시하였다. 일반적으로 양자간에는 문서화, 설계와 원시코드의 연계성, 모듈성(modularity) 및 확장성, 그리고 기능독립성의 정도에서 차이가 있을 수 있음이 주장되고 있다.

구조적 방법과 객체지향 방법에서 야기되는 문제점으로는 각각 다음과 같은 사항이 지적되고 있다. 구조적 방법은 비록 여러 단계의 자료 흐름도가 처리를 분해함으로써 시스템을 계층적으로 표현할 수는 있으나 분석된 시스템을 모형화하는 과정에서 자료 흐름의 논리적인 중복이나 제어 정보 등을 표현할 수 없다. 또한 현재의 표기법만으로는 좀 더 자세하고 정형적으로 시스템을 표현하는데 적지 않은 제약이 따르며 설계와 구현사이에 불일치성을 내포할 가능성도 높다[6]. 객체지향 방법은 객체 및 클래스의 상속성, 캡슐화, 다형성 구현 등의 특성으로 응집도(cohesion) 및 결합도(coupling) 개선을 통한 모듈성이나 재사용성에 뛰어난 면이 있다고는 하나, 현실적으로 객체지향 특성들이 무절제하게 사용되었을 때는 오히려 프로그램의 이해에 많은 어려움을 주게 된다[19, 30]. 유지보수 단계에서 프로그램 이해의 어려움은 곧 비용의 증가를 의미한다[2].

2.2. 유지보수성 체계

소프트웨어 유지보수란 소프트웨어 수명을 연장시키기 위한 일련의 수정활동이며 유지보수성(maintainability)은 '규정된 수정을 수행하는데 필요한 노력과 관련된 속성들의 집합'으로 정의된다[14]. 소프트웨어 수정의 범주에는 목적에 따라 하자보수, 기능보수, 새로운 환경으로의 적응, 그리고 변화에 대비하는 예방적 조치 등이 포함된다[22]. 소프트웨어 유지보수가 중요하게 부각된 이유는 다음과 같이 요약될 수 있다. 첫째, 소프트웨어 예산에서 유지보수비용이 차지하는 비중이 급격히 높아지고 있어 이에 관한 대책이 요구된다. 둘째, 질적인 측면에서 소프트웨어 업무에 종사하는 전문가들의 대부분이 앞으로는 신규 프로젝트보다는 기존 소프트웨어 개선에 더욱 많이 투입되리라는 전망이다. 셋째, 소프트웨어 기술의 발전에 따른 패키지 확산은 다수 고객들을 위한 서비스 및 유지

〈표 1〉 구조적 방법과 객체지향 방법의 대비

	구조적 방법	객체지향 방법
주요 연구자	Stevens, Myers, Constantine(1974) Ross&Schoman(1977), Yourdon&Constantine(1978), DeMarco(1979), Gane&Sarson(1982), Ward&Mellor(1985)	Booch(1994), Coad&Yourdon(1991), Jacobson(1992), Rambaugh(1991), Wirfs, Wilkerson, Weiner(1990)
모형화의 주요 요소	Data Dictionary Entity-Relationship Diagram Data Flow Diagram Data Object Description Process Specification State-Transition Diagram	Use Cases Class-Responsibility-Collaborator Index Cards Object-Relationship Model Object-Behavior Model Attribute, Operations, Collaborators
특 징	순차(sequence), 선택(selection), 반복(repetition) 등을 통한 절차의 구조화	추상화(abstraction), 상속성(inheritance), 캡슐화(encapsulation), 다형성(polymorphism)
분석단계에서의 비교 차원	1. 개체의 식별/분류 2. 특수/일반 및 부분/전체의 개체들간 관계 3. 다른 개체와의 관계 4. 개체 속성에 관한 서술 5. 대규모 모형 분할 6. 상태 전이 7. 기능에 대한 세부 명세 8. 하향식 분해 9. end-to-end 프로세싱 순서 10. 배타적인 서비스의 식별 11.개체간 의사소통	
설계단계에서의 비교 차원	1. 모듈의 계층에 관한 표현 2. 데이터 정의들의 명세, 3. 절차 논리의 명세 4. end-to-end 프로세싱 순서 표시 5. 객체 상태의 전이 표시 6. 클래스와 계층 표현 7. 클래스 연산 할당 8. 연산의 구체적 정의 9. 메시지 연결의 명세 10. 고유한 서비스 식별	

〈표 2〉 유지보수성 관련 품질특성 및 정의

구 분			유지보수성				정 의
외부 품질 특성+			분석성	변경성	안정성	시험성	부적합의 진단이나 고장 원인 분석 또는 수정된 부분의 식별을 위하여 필요한 노력과 관련된 소프트웨어 속성
							결함 제거 또는 환경 변화를 위하여 필요한 노력과 관련된 소프트웨어 속성
							수정에 따른 예기치 않은 결과로부터의 위험과 관련된 소프트웨어 속성
							수정된 소프트웨어를 확인하기 위하여 필요한 노력과 관련된 소프트웨어 속성
내부 품질 특성	그룹 I	추적성	O	O	O	O	구현된 요구 사항과 객체간의 관계에 관한 추적의 용이성에 영향을 미치는 소프트웨어의 속성
		일관성	O	O	O	O	설계와 구현에 사용된 기술, 표기법, 용어 및 부호의 일관성에 영향을 미치는 소프트웨어 속성
		모듈성	O	O	O	O	소프트웨어 구조, 소프트웨어에 대한 변경 및 수정의 지역화[국부화]에 영향을 미치는 소프트웨어 속성
		단순성	O	O	O	O	소프트웨어 사양을 간단, 명료하게 구현하는 능력에 영향을 미치는 소프트웨어 속성
		도구성	O	O	O	O	프로그램의 상태를 감시하는 능력에 영향을 미치는 소프트웨어 속성
		확장성	O	O	O	O	요구 사항의 변경 또는 추가를 쉽게 하는 능력에 영향을 미치는 소프트웨어 능력
	그룹 II	자기 기술성	O	O		O	기능 및 기능들간의 관계를 설명할 수 있는 능력에 영향을 미치는 소프트웨어 속성
		S/W 독립성	O	O		O	소프트웨어가 특정 소프트웨어 환경 (OS, 컴파일러, 프로그래밍 언어, 유틸리티 및 다른 응용 시스템)으로부터 독립적으로 운용될 수 있게 하는 능력에 영향을 미치는 소프트웨어 속성
		기계 독립성	O	O		O	소프트웨어가 특정 하드웨어 환경(시스템 환경, 기계 유형, 장치, 터미널 등)으로부터 독립적으로 운용될 수 있게 하는 능력에 영향을 미치는 소프트웨어 속성
	그룹 III	간결성	O	O			짧고 명확한 표현을 사용하는 능력에 영향을 미치는 소프트웨어 속성
		제품 관리성	O	O			사용자가 제품의 구성, 버전, 일관성 또는 내력을 쉽게 관리할 수 있는 능력에 영향을 미치는 소프트웨어 속성

+ 본 연구의 외부 품질 부특성은 유지보수성에 한정되므로 외부 품질특성으로 칭하기로 함
※ O는 상호 관련되어 있음을 나타냄

보수에 보다 많은 소프트웨어 엔지니어의 투입이 요구되고 있다[5]. 이와 같은 중요성을 감안하여 본 연구에서는 소프트웨어 유지보수성을 소프트웨어 개발방법 평가의 목표로 설정하고자 한다.

소프트웨어 유지보수성을 위한 평가 기준의 체계는 우선 유지보수성을 소프트웨어 개발의 전체 과정 속에서 파악하여야 보다 객관적이고 불편(unbiased)한 접근이 가능할 것으로 판단된다. 최근에 논의된 소프트웨어 개발에 관한 포괄적 접근 중 하나는 소프트웨어 품질 보증으로 개발 과정의 품질 보증과 제품의 품질 평가로 구별되며 ISO 9000-3과 ISO 9126이 각각 대응되는 표준으로 제정되었다. 소프트웨어 품질 평가와 관련해서는 McCall [20], Bohem[7], Dromey[11] 등의 연구도 있으나, ISO 9126이 소프트웨어 개발 단계나 제품의 종류에 구애받지 않는 범용 목적의 품질 평가 체계를 제시하였다. ISO 9126의 평가 체계는 소프트웨어 사용상의 관점인 외부 품질특성(external quality characteristics)과 개발상의 제품 특성으로 내부 품질특성(internal quality characteristics)으로 되어 있다. 외부 품질특성은 유지보수성을 포함하여 기능성, 신뢰성, 사용성, 효율성, 이식성 등 6개의 품질특성(quality characteristics)과 각각 하위에 2-5개씩 총 21개의 품질 부특성(sub-characteristics)이 계층 구조를 이루고 있다. 또한 40개의 내부 품질특성이 함께 제시되고, 소프

트웨어 품질 생명 주기상 내부 품질특성과 외부 품질특성은 상호 연관되어 있다[14, 15].

본 연구는 ISO 9126 품질 체계를 수용하여 소프트웨어 개발방법의 평가를 위한 기준으로 삼고자 한다. 이를 통하여 사용자 관점의 외부 품질로부터 연계된 내부 품질특성에 대하여 개발 과정의 제어 원리로서 소프트웨어 개발방법의 평가가 가능하게 된다. 소프트웨어 유지보수성에는 분석성, 변경성, 안정성 그리고 시험성 등 4개의 품질 부특성이 포함되어 있고 내부 품질특성 40개 중 11개의 특성이 앞의 4개 품질 부특성 중 하나 이상에 관련되어 있다[4, 15]. <표 2>에 이들 관련성을 각 품질특성의 정의와 함께 나타내었으며 내부 품질특성은 다시 3개의 그룹으로 분류하였다. 구분의 기준은 외부 품질특성 요소들과의 관련 정도로, 4개의 외부 품질특성에 대하여 이들 모두와 관련된 내부 품질특성들을 첫째 그룹으로, 이들 중 공통적으로 3개 및 2개와 관련이 있는 품질특성들을 각각 둘째 및 셋째 그룹으로 하였다. 이렇게 구분한 이유에 관해서는 후술하기로 한다.

3. ANP 기법을 이용한 평가 모형의 설계

앞서 제시된 평가 기준 체계에 따른 적절한 평가 기법의 선택에는 다음과 같은 두 가지 사항이 고려되었다. 우선 ISO 9126에 제시된 품질특성의 측정을 위한 객관화된 지표(objective indicator)가 아직까지 검증되지 않았으므로[18], 본 연구는 소요되는 평가 자료의 획득 방안으로 전문가 의견법을 채택하기로 하였다. 따라서 전문가들에 의한 주관적 평가 자료를 객관화하는데 있어 이론적 근거를 갖춘 기법이 요구된다. 다음으로는 지금까지 많은 의사결정 문제가 해결 방안의 단순화를 위하여 고려되는 구성 요소들간 독립성을 가정하였지만 현실적으로 종속성을 고려해야 할 경우가 적지 않다[27]. 따라서 본 연구에서는 평가 기준들간에 또는 평가 기준과 개발방법의 특징들간에 상호 종속성이나 피드백(feedback) 효과 등이 존재한다면 이를 최대한 반영할 수 있는 기법을 선택하고자 하였다.

첫 번째 요구 사항에 대하여 많은 기법들 중 계층 분석과정(Analytic Hierarchy Process : AHP) 기법이 주관적 판단을 객관화하는데 비교적 과학적이며 이론적 근거를 갖춘 것으로 알려져 있다[3]. 그러나 AHP 역시 요소들의 독립성을 가정하고 있어 두 번째 요구를 만족시키지 못한다. 이에 본 연구에서는 AHP의 확장 형태인 네트워크 분석과정(Analytic Network Process : ANP) 기법을 채택하여 본 연구의 평가기법으로 활용하고자 한다.

3.1. 네트워크 분석과정 (Analytic Network Process : ANP)

ANP는 AHP의 확장된 이론이나 문제 접근의 사상적 기저는 다소 상이하다. AHP가 목표-기준-대안의 순차적 흐름으로 표현되는 단방향 트리(tree) 구조의 의사결정 시스템인 반면, ANP는 목표, 기준 그리고 대안 상호간의 종속성이나 피드백을 포함하는 네트워크 구조의 의사결정 시스템이다. 이 때 종속성은 기준 집단 또는 대안 집단을 구성하는 요소들간 존재하는 영향력의 종속성을 의미하며 집단간 요소들 사이에 작용하는 외적 종속성(inter-dependence)과 집단 내 요소 사이에 작용하는 내적 종속성(inner-dependence)으로 구분된다. Saaty[25]에 의해 개발된 AHP는 지난 20여 년간 광범위한 분야에 적용되어 널리 알려진 기법이므로 본 연구에서는 AHP[26]와 ANP[27]의 차이를 중심으로 ANP의 특징을 기술한다.

우선 AHP가 비교되는 요소들간에 목표의 속성(property)을 계층적으로 배분하는 과정이지만, ANP는 목표에 대한 요소들의 영향(influence)을 배분하는 과정이다. AHP는 상위 계층에 속하는 요소의 속성을 하위 계층의 요소들이 얼마나 보유하고 있는가를 비교하여 중요도 또는 우선 순위를 산출하는 이른바 속성의 지배(dominance of property) 정도를 측정한다. 이 때 요소간에는 독

립성을 가정하므로 동일 요소에 속한 하위 요소들만을 비교하는 하향(top-down) 방식의 선형적 분화가 사고의 기본 메커니즘이다.

ANP는 요소들 상호간 네트워크 구조의 영향력을 발휘할 때 각 요소가 네트워크 내에서 발휘하는 궁극적인 영향력, 즉 영향력의 지배(dominance of influence) 정도를 측정한다. 모든 의사결정 문제가 계층적으로 구조화되는 것은 아니다. 의사결정에 관여하는 기준의 중요도가 대안의 결정에 영향을 미칠 수도 있지만 반대로 대안마다 지니는 특성이 기준의 중요도에 영향을 미칠 수도 있다. 또한 요소들간 상호작용에 의한 상승(synergy) 효과가 존재하는 경우 및 하위 계층의 요소로부터 상위 계층의 요소에 미치는 피드백 효과가 존재하는 경우가 드물지 않다. 이러한 문제의 구조는 계층적 분화의 의미를 퇴색시키고 네트워크 설정을 정당화 할 것이다. 의사결정 문제에 따라서는 몇 개의 네트워크가 형성될 수도 있으며 이 때 네트워크를 형성케 하는 기준들을 통제 기준(control criteria)이라 하고 이들의 계층적 구조를 통제 계층(control hierarchy)이라 한다.

네트워크 구조의 한 특수한 형태가 트리(tree) 구조이므로 ANP의 연산은 AHP의 연산 절차를 포함하는 포괄적이며 다소 복잡한 절차를 거친다. 네트워크 상에 존재하는 각 요소의 지배 정도는 그래프(graph) 이론을 동원하면 특정 값에 수렴함을 밝힐 수 있다. 이 때 얻어지는 수렴 값은 요소들간에 존재하였던 종속관계 및 상승 효과 그리고 피드백 효과 등이 모두 작용된 결과로 도출된다. 실제 계산은 네트워크의 그래프적 표현을 수퍼매트릭스(Supermatrix : W)라는 대수적 표현으로 전환하여 수행된다. 수퍼매트릭스는 2차원 매트릭스로 열은 영향을 미치는 요소를, 행은 영향을 받는 요소를 나타낸다. 특정 열의 요소에 대하여 영향관계가 전혀 없는 행의 요소들은 그 열과의 각 교차점에 0이 삽입되며 영향을 받는 행의 요소들은 영향의 크기가 각 교차점에 삽입된다. 이를 각 열별로 그 합이 1이 되도록 정규화하여 확률행렬(stochastic matrix)로 만들면 최초의 수퍼매트릭스가 완성된다. 이 행렬의 고유치(λ) 중 최대값($\lambda_{\max}$)은 1이다.

수퍼매트릭스의 극한(W^{∞}) 연산은 구성된 수퍼매트릭스의 대수적 특성에 따라 상이하며 다음 두 가지 기준을 적용하여 구분할 수 있다. 첫째, 그 고유치 중 1이 되는 근의 개수에 따른 구분(simple/multiple)과 둘째, $\lambda_{\max}=1$ 이외에 복소수 평면에서의 크기가 1이 되는 근의 존재 여부에 따른 구분이다. Saaty는 앞의 두 가지 기준의 조합에 의하여 발생 가능한 경우를 3가지로 분류한 후 각각에 대한 극한 계산식을 제시한 바, 다음과 같다(유도과정은 Saaty[27], pp.108-127 참조).

Case 1 :

$(|\lambda_i|<1 \text{ for } i>1) \wedge (\lambda_1=1 \textit{ simple})$,

$$\frac{(I-W)^{-1}\triangle(1)}{\triangle'(1)}=\frac{Adjoint(I-W)}{\triangle'(1)},$$

열의 합을 1로 정규화 (1)

Case 2 :

$(|\lambda_i|<1 \text{ for } i>1) \wedge (\lambda_1=1 \textit{ multiple}(n_1))$,

$$n_1\sum_{k=1}^{n_1}(-1)^k\frac{n_1!}{(n_1-k)!}\frac{\triangle^{(n_1-k)}(\lambda)}{\triangle^{(n_1)}(\lambda)}(\lambda I-W)^{-k-1}\Big|_{\lambda=1} \quad (2)$$

Case 3 :

$|\lambda_i|\le 1$ for *several i*.

$$\frac{1}{c}(I-W^c)(I-W)^{-1}(W^c)^{\infty} \quad c\ge 2 \quad (3)$$

여기서, $\triangle^{(m)}(\lambda)=\prod_{i=1}^{n}(\lambda-\lambda_i)$의 m차 미분계수, $\lambda_1=\lambda_{\max}$이며 c는 수퍼매트릭스가 극한을 취할 때 동일한 결과가 반복되는 순환 주기(cyclicity)이다.

3.2. 네트워크 모형의 설정 및 연산 절차

3.2.1. 네트워크 모형

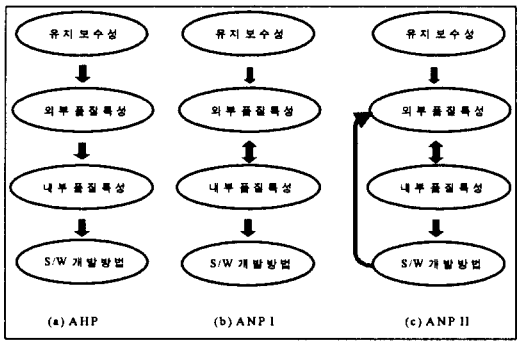
본 연구의 목적을 수행하기 위한 네트워크 모형의 설정은 목표, 기준, 대안의 체계를 유지하고 이들 사이에 상호 영향력의 방향을 결정함으로써 완

성된다. 모형의 최상위에 통제 기준으로서 유지보수성이라는 목표가 설정되고 평가 기준에는 유지보수성을 구성하는 4개의 외부 품질특성군과 이들과 관련되는 11개의 요소로 구성된 내부 품질특성군이 포함된다. 11개의 요소들은 다시 3개의 그룹으로 구분하였는데(<표 2> 참조), 그 이유는 본 연구의 자료 획득 원천이 전문가의 지식과 경험을 토대로 하는 주관적 판단이며 이때 판단의 신뢰성을 높이기 위하여 동시에 고려하는 비교대상의 수를 Miller[21]의 연구 결과로 알려진 7±2 이내로 설계하고자 했기 때문이다. 또한 동시에 비교되는 대상들 사이에 요구되는 동질성(homogeneity)의 기준은 각 그룹의 요소들과 공통적으로 관련되는 외부 품질특성으로 하였음을 이미 밝힌 바 있다. S/W 개발방법 대안은 본 연구에서 비교하고자 하는 구조적 방법과 객체지향 방법이다.

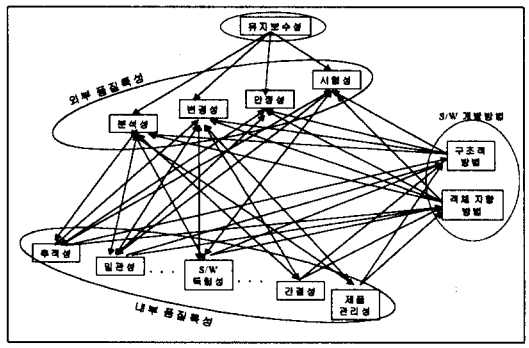
유지보수성 목표 하의 3가지 요소군의 상호 영향력의 가정에 따라 [그림 1]과 같이 서로 상이한 네트워크를 형성하게 된다. ISO 9126[14] 제정 과정에 적용되었던 원칙 중 하나는 동일 계층 내에 존재하는 품질특성간 독립성을 최대한 유지하는 방향이었으며 이를 본 연구의 모형 설정에서 반영하였다. 따라서 제시된 통제 모형에는 같은 군에 속하는 요소들간 작용하는 내적 종속성은 모두 배제되어 있다.

본 연구는 집단간 요소들 사이에 작용하는 외적 종속성 중 개연성이 인정되면 모두 모형에 포함시켜 3가지 네트워크 모형을 설정하였다. 모형 (a)는 목표로부터 기준 및 대안까지 속성의 순차적 분화를 평가하는 전통적 AHP 모형으로 목표를 위한 기준의 중요도가 결정되고 각 기준에 따라 대안의 우수성이 비교되는 절차를 정형으로 한다. 모형 (b)와 모형 (c)은 계층적 분화에 대한 하나 이상의 피드백을 포함하는 ANP 모형으로 각각 ANP I 모형 및 ANP II 모형으로 명명하였다. 두 ANP 모형은 공히 외부 품질특성군과 내부 품질특성군 사이의 양방향 영향력을 포함하는데 이는 두 품질특성군 사이의 상호 종속성을 의미한다. ANP II 모형은 개발방법 대안군으로부터 외부 품질로의 피드백이 추가되어 ANP I 모형과 차이를 보인다.

첫 번째의 양방향 영향력은 유지보수성을 위한 속성의 보유 정도와 속성이 미치는 강도(intensity)의 차이를 의미한다. 유지보수성에 대하여 외부 품질특성에 요구되는 속성의 보유 정도를 평가하여 각 내부 품질특성의 중요도를 결정하고, 반대로 내부 품질특성이 각 외부 품질특성에 미치는 영향의 강도에는 차이가 있으므로 이를 평가하여 외부 품질특성의 중요도를 결정한다. 이때 속성의 보유 정도와 영향의 강도 사이에 상호 종속성이 존재할 수 있다. ANP II 모형의 개발방법 대안군으로부터 외부 품질특성군으로의 피드백도 같은 논리의 전개가 가능하다. 각 내부 품질특성에 따른 개발방법의



[그림 1] 3가지 네트워크 모형



[그림 2] ANP II의 요소간 영향도

우수성은 앞의 내부 품질과 외부 품질의 상호 종속성이 작용된 결과를 거쳐 외부 품질특성에 따른 우수성으로 의미된다. 한편, 각 개발방법이 지니고 있는 특징에 따라 외부 품질특성의 중요도가 달리 평가 될 가능성을 배제하지 않았다. [그림 2]에 ANP II의 경우를 들어 가정되는 요소간 구체적 영향의 관계를 나타내었다.

3.2.2. 수퍼매트릭스의 형태와 연산 절차

본 연구에서 설정한 3가지 네트워크 모형에 대응되는 수퍼매트릭스의 형태는 [그림 3]과 같다. 각 수퍼매트릭스의 행과 열은 평가 모형 내에 존재하는 모든 요소들에 대응되며 각 요소들은 e_{jk}(j번째 요소군의 k번째 요소)로 표기한다. 수퍼매트릭스의 내부는 요소군별로 블록화하여 나타내었다. 이들 중 W_{ij}로 표기된 블록은 i번째 군과 j번째 군의 요소간 관계를 나타내는 비영(non-zero) 행렬로 괄호 안의 $n \times m$이 행렬의 차원을 나타낸다. 비어 있는 블록은 영 행렬을 의미하며 단위 행렬(I)은 확률행렬을 구성하기 위해 도입된 것으로 결과에는 아무런 영향이 없다. 최초 수퍼매트릭스의 작성은 각 열별로 다음과 같은 절차를 따른다(<부록 I> 참조).

step 1 : 현재 열을 나타내는 요소에 직접 영향을 받는 요소들을 선정하여 집합 S라 한다.

step 2 : S내의 동일 군에 속하는 요소들간 쌍대비교(pairwise comparison)에 의한 쌍대비교 행렬을 구성한다. 쌍대비교 행렬의 최대 고유치를 산출하고 이에 대응되는 고유벡터를 구하여 합이 1이 되도록 정규화한다(내부 품질특성군에 속하는 요소들은 그룹별 비교를 먼저하고 그룹내의 요소들간 비교를 한다).

step 3 : 수퍼매트릭스에서 현재의 열과 S내의 각 요소를 타나내는 행과의 교차점에 step 2에서 구한 해당 요소의 중요도를 기록한다.

step 4 : S에 포함되지 않은 요소들을 나타내는 행과의 교차점에는 0을 기록한다.

step 5 : 현재 열의 합이 1이 되도록 정규화 한다.

AHP 모형에 대한 극한 수퍼매트릭스 연산은 앞서 제시된 식 (2)의 다소 복잡한 계산 절차가 적용된다.[1] 그러나 AHP의 계층적 분화를 이용한 보다 간단한 연산 절차가 이미 널리 알려져 있고 소프트웨어 분야에서도 여러 차례 적용되어 온 바[3, 4, 17, 32], 본 연구에서는 이를 생략하고 ANP에 의한 두 평가 모형의 연산 절차에 관하여 언급하기로 한다.

ANP I 및 ANP II 모형의 네트워크 형태는 각각 Intarchy 및 Primintarchy로 불리우며(Saaty[27], p.84), 모두 목표로부터 중요도가 분화된다. 이러한 경우의 극한 수퍼매트릭스 연산은 목표(e_{11})를 나타내는 행과 열을 제외한 부분행렬(sub-matrix)의 극한 수퍼매트릭스를 구하여도 같은 결과에 도달하며, 연산 결과에서 목표를 나타내는 행은 퇴화되어 0이 산출된다. 또한 행렬의 특성과 관련한 다음의 정의를 이용하면 본 연구의 수퍼매트릭스 극한 연산은 더 수월하게 결과를 얻을 수 있다[27].

Irreducibility : 비음 확률 행렬(non-negative stochastic matrix)의 0 아닌 고유치가 유일하다. $W_{(n,n)}$가 이를 만족하기 위한 필요충분 조건은 $(I+W)^{n-1}>0$이다. 이와 상대되는 개념은 Reducibility이다.

Primitivity : 비음 확률 행렬의 최대 고유치 이외에 1이 되는 고유치는 없다. W가 이를 만족하기 위한 필요충분 조건은 어떤 양의 정수 m에 대하여 $W^m>0$이다. 이와 상대되는 개념은 Imprimitivity이다.

ANP I 모형은 개발방법 대안군으로부터 피드백이 존재하지 않으므로 분리 가능(reducible)하고 이러한 형태의 수퍼매트릭스 극한 연산은 다소 간단

1) 만약 상당히 큰 m에 대하여 W^m를 취하면 퇴화(degeneracy)가 발생하여 대안을 제외한 요소 행에는 모두 0이 산출되고 각 대안의 목표에 대한 기여도만이 최종 결과로 나타난다.

하게 조정 가능하다. 모형에서 분리 불가능한(irreducible) 네트워크에 해당하는 부분행렬에 대하여 수퍼매트릭스 연산을 적용하고 그 결과에 대안의 비교 결과를 결합하는 것이다. 본 연구의 수퍼매트릭스 (b)에서 대안군의 요소(e_{41}, e_{42})에 해당되는 행과 열을 제외한 부분행렬에는 주기가 2인 순환이 존재하며 식 (3)이 적용된다. 한편, ANP II 모형은 ANP I 모형에 대안군으로부터 외부 품질 특성군으로의 피드백을 설정하여 대안군의 분리가 불가능하다. 또한 수퍼매트릭스 (c)에서 목표(e_{11})를 나타내는 행과 열을 제외한 부분행렬($\overline{W}$)

구분	목표 e11	외부품질 e21-e24	내부품질 e31-e3,11	대안 e41 e42
e11				
e21 | e24	W_{21} (4x1)			
e31 | e3,11		W_{32} (11x4)		
e41 e42			W_{43} (2x11)	I (2x2)

(a) AHP

구분	목표 e_{11}	외부품질 e_{21}-e_{24}	내부품질 e_{31}-$e_{3,11}$	대안 e_{41} e_{42}
e_{11}				
e_{21} | e_{24}	W_{21} (4x1)		W_{23} (4x11)	
e_{31} | $e_{3,11}$		W_{32} (11x4)		
e_{41} e_{42}			W_{43} (2x11)	I (2x2)

(b) AHP I

구분	목표 e_{11}	외부품질 e_{21} e_{24}	내부품질 e_{31} $e_{3,11}$	대안 e_{41} e_{42}
e_{11}				
e_{21} | e_{24}	W_{21} (4x1)		W_{23} (4x11)	W_{24} (4x2)
e_{31} $e_{3,11}$		W_{32} (11x4)		
e_{41} e_{42}			W_{43} (2x11)	

(c) AHP II

[그림 3] 수퍼매트릭스 형태

은 어떤 양의 정수 m에 대하여 $\overline{W}^m > 0$를 만족하는 행렬(primitive matrix)이다. 이러한 경우 원칙적으로 식 (1)이 적용되어야 하나 단순히 m을 상당히 크게 하여 $\overline{W}^m$을 구하여도 같은 결과를 얻는다.

4. 실증 분석

4.1. 표본의 특성 및 조사 방법

본 연구의 설문 응답자는 소프트웨어 개발방법에 관한 지식과 경험이 있는 전문가이어야 하므로 다음과 같은 제한을 두었다. 첫째, 소프트웨어 개발 분야에 3년 이상 종사한 전문가로 하였다. 이는 요구되는 지식과 함께 경험적 판단을 필요로 했기 때문이다. 둘째, 본 연구의 목적에 따라 구조적 방법과 객체지향 방법 모두를 경험하고 또한 유지 보수 업무에 관하여도 정통한 전문가로 하였다. 셋째, 본 연구의 결과에 신뢰도를 높이기 위해 비교적 규모가 큰 국내 시스템 통합(System Integration) 업체를 포함한 총 21개 업체의 전산/정보 시스템 개발직에 종사하는 전문가들로 하였다.

조사 방법으로는 우편을 이용한 방법을 주로 이용하였고 직접 면담법을 일부 병행하였다. 설문의 신뢰성을 높이기 위하여 설문에는 응답자의 성명을 포함한 연락처를 기재하도록 하였다. 1998년 4월부터 5월까지 약 1개월에 걸친 조사를 통하여 총 150부의 설문을 배포하고 47부를 회수(회수율 31.3%)하였다. 설문내용이 소프트웨어 공학에 관한 상당한 전문적 지식과 경험을 요구했기 때문에 설문에 응할 수 있었던 전문가가 그리 많지 않았던 것으로 판단된다. 이 중 AHP 기법에서 제공하는 일관성 비율(Consistency Ratio, Saaty[26], p.21)을 참고하여 신뢰성이 있다고 판단되는 설문 24부(24/47=51.1%)를 분석에 사용하였다.

본 연구에서 비교 분석할 모형의 종류는 세 가지이나, 조사할 자료 항목의 측면에서 ANP II 모형이 다른 두 가지 모형을 모두 포함한다. 따라서 설문지는 ANP II 모형에 소요되는 자료를 획득하

도록 작성하고 분석 단계에서 획득된 자료를 각 모형별로 구분하여 사용하였다. 설문지 구성은 크게 둘로 나누어 Part-I에서 유지보수성 목표 → 외부 품질특성 → 내부 품질특성 → 소프트웨어 개발방법 순서로 요소들을 쌍대비교 하였고, Part-II에서는 내부 품질특성 → 외부 품질특성 및 소프트웨어 개발방법 → 외부 품질특성의 영향력 파악을 위한 요소간 쌍대비교를 하였다. 쌍대비교를 위한 설문은 Saaty[26]에 의해 정확성과 타당성이 검증된 9점 비율척도(ratio scale) 및 그 역수들로 작성하였다(부록 II 참조).

4.2 모형간 차이 검정 방법

각 모형간 차이 유무는 수퍼매트릭스 연산 결과로 얻어진 품질특성 요소들의 중요도간 차이를 분석함으로써 밝힐 수 있다. 차이에 대한 검정 방법으로서 Saaty[27]가 제안한 두 역수행렬(reciprocal matrix) 사이의 일치성 지수(Compatibility Index : *SI*)를 활용하였다. n차원의 두 역수행렬 $A=(a_{ij})$와 $B=(b_{ij})$에 대한 SI는 다음과 같이 정의된다.

$$SI_{AB}=n^{-2}\cdot e^{T}A\odot B^{T}e \qquad (4)$$

여기서, $\odot$표기는 Hadamard 곱으로 $A\odot B^{T}=(a_{ij}\cdot b_{ji})_{n\times n}$, $e^{T}=(1,1,\cdots,1)$.

만약 두 행렬이 완전히 일치한다면 두 행렬간 SI의 값은 1이 된다. 두 행렬의 일치 정도가 적을수록 SI의 값은 1보다 점점 커지게 된다. 두 행렬의 일치 정도는 각 행렬로부터 얻어지는 고유벡터의 일치 정도를 나타내게 된다.

행렬 A의 λ_{max}에 대응되는 고유벡터 요소들간 쌍대비교로 이루어진 행렬을 V라 하면, 두 행렬 A와 V사이에 $SI_{AV}=n^{-2}\cdot e^{T}A\odot V^{T}e=\lambda_{max}/n$의 관계가 존재하여 λ_{max}를 매개로 일관성 지수(Consistency Index : $CI=(\lambda_{max}-n)/(n-1)$)에 대응된다. 따라서 허용될 만한 비일관성 수준을 결정하는데 적용했던 판단기준으로 허용될 만한 불일치성 수준을 결정할 수 있다. Saaty는 일치성 여부의 판단 기준이 되는 SI 값으로 대략 1.1을 제안한 바 있다(Saaty[27], pp.58-65).

4.3 결과 분석

4.3.1. 모형에 따른 요소의 중요도 차이 분석

수집한 자료는 동일한 항목별로 기하평균을 구하였다. 쌍대비교의 척도는 상대적 비율척도이며, 이때 구성되는 쌍대비교 행렬은 역수행렬이 된다. 복수의 측정 결과를 요약할 때 기하평균만이 역수행렬 조건을 만족하게 된다. 고유치 방법(eigenvalue method)을 통하여 산출된 결과로 본 연구에서 설정한 3가지 모형에 따른 최초 수퍼매트릭스를 작성하였다. 전절에서 언급한 연산 방법에 따라 각 수퍼매트릭스의 최종 결과를 산출하여 <표 3>에 정리하였다. 각 모형별로 최초 수퍼매트릭스, 극한수렴 수퍼매트릭스 그리고 블록 정규화 수퍼매트릭스(동일 그룹내 중요도 합이 1이 되도록 정규화)가 산출된다. <부록 I>에 ANP-II에 대응되는 3가지 수퍼매트릭스를 예시하였다.

〈표 3〉 모형에 따른 중요도 결과

구 분		평가 모형에 따른 중요도		
		(a) AHP	(b) ANP I	(c) ANP II
외부 품질 특성	분석성	0.3225 (2)	0.3334 (2)	0.2956 (2)
	변경성	0.2021 (3)	0.3383 (1)	0.3373 (1)
	안정성	0.3388 (1)	0.1438 (4)	0.1685 (4)
	시험성	0.1366 (4)	0.1846 (3)	0.1985 (3)
내부 품질 특성	추적성	0.0950 (5)	0.0859 (5)	0.0869 (4)
	일관성	0.1518 (2)	0.1240 (2)	0.1278 (2)
	모듈성	0.1869 (1)	0.1645 (1)	0.1683 (1)
	단순성	0.1011 (4)	0.0838 (6)	0.0869 (4)
	도구성	0.0639 (6)	0.0581	0.0589
	확장성	0.1301 (3)	0.1217 (3)	0.1218 (3)
	자기기술성	0.0596	0.0742	0.0714
	S/W독립성	0.0612	0.0799 (7)	0.0774 (7)
	기계독립성	0.0379	0.0491	0.0474
	간결성	0.0496	0.0728	0.0707
	재품관리성	0.0629 (7)	0.0861 (4)	0.0824 (6)

※ 괄호 안은 중요도 순위

각 모형에 따른 중요도 산출 결과의 특징적인 면만을 살펴보기로 한다. 유지보수성을 위한 4개의 외부 품질특성에 대하여, AHP 모형에 의하면 수정된 소프트웨어 확인과 관련된 안정성이 가장 중요하고 분석성, 변경성, 시험성의 순서로 중요도가 산출되었다. 반면, 두 ANP 모형에서는 결함을 제거하거나 환경 변화를 반영하기 위한 노력과 관련되는 변경성과 부적합의 진단이나 고장 원인을 분석하는 분석성의 중요도가 높게 나타났고 시험성 및 안정성의 중요도는 상대적으로 낮게 나타났다. 내부 품질특성의 중요도와 관련하여, 총 11개의 품질특성 중 모듈성의 중요도가 가장 크게 나왔으며 일관성 및 확장성도 상당히 중요한 품질특성으로 확인되었다. 이는 기존 문헌에서 주장된 내용과도 일치하여 본 연구 결과의 신뢰성을 가늠해 볼 수 있다. 이외에도 추적성, 단순성, 제품 관리성, S/W 독립성 등이 비교적 중요한 것으로 나타났다. 전체적인 중요도 순위의 양상에서 AHP 모형과 두 ANP 모형사이에 다소의 차이를 나타내었다.

모형간 차이에 대한 검정은 3개 모형을 둘씩 짝지어 3개 조합에 대하여 수행하였으며 외부 품질특성과 내부 품질특성은 구분하여 식 (4)를 적용하였다. 식 (4)의 SI를 구하는데 소요되는 각 모형의 쌍대비교 행렬은 <표 3>에 제시된 각 항목의 중요도 결과를 쌍대비교하면 얻을 수 있다. <표 4>의 분석 결과는 두 ANP 모형 사이에 차이가 거의 없음에 비하여 AHP 모형과 ANP 모형 사이에 비교적 확연한 차이를 보여준다. 특히 외부 품질특성에 대하여 두드러진 차이가 나타나는데 이는 외부 품질특성의 중요도 결정에 내부 품질특성과의 상호작용에 의한 효과가 상당함을 의미한다. 내부 품질특성 중요도에 대한 차이는 비록 1.1보다 작기는 하지만 일정 정도 차이가 있는 것으로 해석할 수 있다. 종합적으로 볼 때 외부 품질특성과 내부 품질특성 차이에 대한 지수 평균은 1.1보다 크므로 AHP 및 ANP 모형간 차이가 존재한다고 결론지을 수 있다. 이는 외부 품질특성과 내부 품질특성 사이에 유의한 상호 종속성이 존재함을 의미한다.

〈표 4〉 모형간 일치성 지수

구 분	SI_{ab}*	SI_{ac}*	SI_{bc}*
외부 품질특성	1.3059	1.2441	1.0105
내부 품질특성	1.0470	1.0355	1.0008
평 균	1.1765	1.1398	1.0057

* a : AHP, b : ANP I, c : ANP II

ANP I 모형과 ANP II 모형에 의한 결과에 큰 차이가 없음은 ANP II 모형에 포함된 개발방법 대안으로부터 외부 품질특성으로의 피드백의 효과가 미미하게 작용했음을 의미한다. 즉, 두 방법의 특징적인 면을 고려하여 외부 품질특성을 평가하더라도 중요도에는 변화가 거의 없음이 검증되었다. 이러한 결과는 본 연구에서 설정한 평가 기준이 비록 소프트웨어 제품의 평가를 위한 기준으로부터 차용된 것이지만 소프트웨어 개발방법 평가에도 적용할 수 있는, 개발방법과 독립된 평가 기준임을 보여주는 것이다. 한편, 각 방법에 요구되는 자료 항목 수를 고려한다면 ANP I 모형이 ANP II 모형보다 더 효율적이라고 판단된다.

4.3.2. 소프트웨어 개발방법 비교 결과

두 개발방법의 상대적 우수성 비교 결과를 내부 품질특성별로 구분하여 <표 5>에 제시하였다. 총 11개의 품질특성 중 단순성을 제외한 10개의 품질특성에 대하여 객체지향 방법이 더 우수한 것으로 나타났다. 특히 확장성, S/W 독립성, 모듈성 그리고 기계 독립성에서 객체지향 방법은 월등히 우수한 것으로 분석되었다. 확장성 및 모듈성의 우수성은 근본적으로 객체의 캡슐화 및 다형성의 특성에 기인하는 것으로 판단된다. 또한 S/W 독립성은 주로 운영체제(Operating System)로부터의 독립으로 인식되는 점을 감안하고 여기에 H/W를 포함하는 플랫폼(Platform) 독립성에서 객체지향 방법이 상당히 우수한 것으로 사료된다.

두 개발방법의 종합적인 우수성은 내부 품질특성의 중요도와 각 품질특성에 따른 개발방법의 상대적 우수성을 가중 평균한 결과로 비교된다. <표 5>에서

품질특성별 중요도는 ANP II 모형을 적용했을 때의 결과를 이용하였다. 이는 <부록 I>의 수퍼매트릭스 연산에서도 같은 결과를 나타낸다. 가중 평균 결과, 구조적 방법 대 객체지향 방법이 약 0.35 대 0.65로 산출되었다. 이러한 수치의 의미는 유지보수성 정의가 유지보수를 위한 노력의 정도임을 상기하면 객체지향 방법이 구조적 방법에 비하여 유지보수 측면에서 1.8(=0.65/0.35)배 정도 효율적인 것으로 받아들일 수 있다.

〈표 5〉 대안의 상대적 우수성 비교 결과

내부 품질특성	(중요도)	상대적 우수성	
		구조적 방법	객체지향 방법
추적성	(0.0869)	0.4110	0.5890
일관성	(0.1278)	0.4445	0.5555
모듈성	(0.1683)	0.2535	0.7465
단순성	(0.0869)	0.5426	0.4574
도구성	(0.0589)	0.3922	0.6078
확장성	(0.1218)	0.1895	0.8106
자기기술성	(0.0714)	0.3395	0.6605
S/W독립성	(0.0774)	0.1943	0.8057
기계독립성	(0.0474)	0.2600	0.7400
간결성	(0.0707)	0.3893	0.6107
제품관리성	(0.0824)	0.4883	0.5117
종합 비교 결과		0.3480	0.6520

3가지 모형에서 산출된 내부 품질특성 중요도에 따른 개발방법의 종합적인 우수성 결과를 <표 6>에 정리하였다. AHP 모형에 의한 내부 품질특성의 중요도 양상이 다른 두 ANP 모형에 의한 중요도 양상과 다소 상이함에도 불구하고 모두 비슷한 결과가 도출되었다. 이는 객체지향 방법이 각 내부 품질특성에 대하여 골고루 우수한 것으로 평가되었기 때문인 것으로 판단된다.

〈표 6〉 3가지 모형에 따른 종합 비교 결과

구 분		평가 모형		
		AHP	ANP I	ANP II
종합 비교 결과	구조적 방법	0.3505	0.3472	0.3480
	객체지향 방법	0.6495	0.6528	0.6520

5. 결 론

본 연구는 소프트웨어 개발방법 평가를 위한 모형을 제시하고 모형에 따른 실증 분석을 수행하였다. 평가 목표로서 소프트웨어 유지보수성을 설정하고 이에 부합되는 평가 기준 체계를 수립하였다. 평가 기법으로서는 구성 요소간 독립성 가정 완화에 적절한 ANP 기법을 이용하였다. 제시한 평가 모형에 따라 대표적 소프트웨어 개발방법이라 할 수 있는 구조적 방법과 객체지향 방법의 우수성 평가를 전문가 의견을 통하여 실사하였다. 본 연구의 결과에서 새로이 밝혀진 사실은 다음과 같이 요약된다.

첫째, 소프트웨어 외부 품질특성과 내부 품질특성간 상호 종속성의 존재다. 이는 유지보수성을 위한 각 외부 품질특성 및 내부 품질특성의 기여도가 독립적으로 파악되는 것이 아니라, 상호 종속의 효과가 반영되어야 보다 정확하게 파악될 수 있음을 의미한다.

둘째, 소프트웨어 품질특성과 소프트웨어 개발방법은 서로 독립에 가깝다. 이는 제시한 평가 기준으로서 품질특성이 특정 개발방법의 특징에만 적용되는 것이 아니므로 개발방법 평가에 있어 불편한(unbiased) 기준임을 시사하는 것이다.

셋째, 유지보수를 위한 노력의 측면에서 객체지향 방법이 구조적 방법에 비하여 실증적으로 우수하며 그 정도는 1.8배 정도인 것으로 결과되었다.

제시된 평가 모형은 유지보수성 이외에도 기능성, 사용성, 신뢰성 등 소프트웨어 품질 전체로 확대 가능성이 있고 본 연구에서 비교한 개발방법 이외에 다양한 대안에 대하여도 접근이 가능할 것으로 판단된다. 또한 본 연구에서 설정한 다양한 모형을 통한 검증 절차는 향후 연구에 효율성을 더할 것으로 사료된다. 한편, ANP 기법은 AHP 기법에 비하여 정확성에 장점이 있으나 자료 항목의 수를 증가시킨다는 단점이 있으므로 적용에 있어 평가 사안의 중요성이 감안되어야 할 것이다.

부 록 1

수퍼매트릭스 예 : ANP II 모형

Initial Supermatrix

구분	목표	외부 품질특성				내부 품질특성										방법 대안	
	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_{17}
e_1	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_2	.3225	.0000	.0000	.0000	.0000	.1954	.1327	.1038	.1655	.1245	.1144	.2760	.2535	.1565	.2079	.2066	.3768
e_3	.2021	.0000	.0000	.0000	.0000	.1199	.1587	.1338	.1222	.0887	.1943	.1377	.1655	.1658	.2921	.2934	.2025
e_4	.3388	.0000	.0000	.0000	.0000	.0765	.0987	.1535	.1053	.1929	.0641	.0000	.0000	.0000	.0000	.0000	.2173
e_5	.1366	.0000	.0000	.0000	.0000	.1082	.1097	.1088	.1070	.0939	.1272	.0863	.0810	.1777	.0000	.0000	.2034
e_6	.0000	.0934	.0432	.1139	.1285	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_7	.0000	.1061	.0825	.2110	.1114	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_8	.0000	.1457	.1182	.2576	.2107	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_9	.0000	.0682	.0419	.1552	.1279	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{10}	.0000	.0590	.0373	.0783	.0787	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{11}	.0000	.1291	.1143	.1540	.0966	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{12}	.0000	.1056	.0691	.0000	.0847	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{13}	.0000	.0915	.1014	.0000	.0819	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{14}	.0000	.0588	.0602	.0000	.0496	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{15}	.0000	.0496	.1663	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{16}	.0000	.0932	.1627	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{17}	.0000	.0000	.0000	.0000	.0000	.2055	.2222	.1267	.2713	.1961	.0947	.1698	.0971	.1300	.1946	.2441	.0000
e_{18}	.0000	.0000	.0000	.0000	.0000	.2945	.2778	.3733	.2287	.3039	.4053	.3302	.4029	.3700	.3054	.2559	.0000

Limiting Supermatrix

구분	목표	외부 품질특성				내부 품질특성										방법 대안	
	e_{18}	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
e_{18}	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_1	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184	.1184
e_2	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351	.1351
e_3	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675	.0675
e_4	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795	.0795
e_5	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348
e_6	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512	.0512
e_7	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674	.0674
e_8	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348	.0348
e_9	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236	.0236
e_{10}	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488	.0488
e_{11}	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286	.0286
e_{12}	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310	.0310
e_{13}	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190	.0190
e_{14}	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283	.0283
e_{15}	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330	.0330
e_{16}	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697	.0697
e_{17}	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306	.1306

Block-Normalized Supermatrix

구분	목표	외부 품질특성				내부 품질특성										방법 대안	
	e_{17}	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
e_{17}	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
e_{18}	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956	.2956
e_1	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373	.3373
e_2	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685	.1685
e_3	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985	.1985
e_4	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869
e_5	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278	.1278
e_6	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683	.1683
e_7	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869	.0869
e_8	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589	.0589
e_9	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218	.1218
e_{10}	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714	.0714
e_{11}	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774	.0774
e_{12}	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474	.0474
e_{13}	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707	.0707
e_{14}	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824	.0824
e_1	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480	.3480
e_2	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520	.6520

부 록 2

쌍대비교 설문 예

예) 유지보수성의 외부 품질특성 중요도 평가

특성 A	A가 절대로 중요		A가 확실히 중요		A가 매우 중요		A가 약간 중요		서로 비슷한 정도		B가 약간 중요		B가 매우 중요		B가 확실히 중요		B가 절대로 중요	특성 B
	9	8	7	6	5	4	3	2	1	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	
분석성					√													변경성
분석성											√							안정성
분석성						√												시험성

응답 예의 설명
유지보수성에 대하여
첫째 행: 특성 B란의 변경성 보다 특성 A란의 분석성이 매우 중요하다.
둘째 행: 특성 A란의 분석성 보다 특성 B란의 안정성이 약간 중요하다.
셋째 행: 약간 중요와 매우 중요의 사이인 경우

예) 내부 품질특성 중 추적성이 외부 품질특성에 미치는 영향의 강도 평가

특성 A	A에게 절대로 더 영향		A에게 확실히 더 영향		A에게 매우 더 영향		A에게 약간 더 영향		서로 비슷하게 영향		B에게 약간 더 영향		B에게 매우 더 영향		B에게 확실히 더 영향		B에게 절대로 더 영향	특성 B
	9	8	7	6	5	4	3	2	1	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	
분석성					√													변경성
분석성											√							안정성
분석성									√									시험성

응답 예의 설명
추적성의 품질 수준은
첫째 행: 특성 B란의 변경성 보다 특성 A란의 분석성에 매우 더 강하게 영향을 미친다.
둘째 행: 특성 A란의 분석성 보다 특성 B란의 안정성에 약간 더 강하게 영향을 미친다.
셋째 행: 서로 비슷하게 영향을 미치는 경우

참 고 문 헌

[1] 류성열, 백인섭, 김하진, 유지 보수 관리 체계의 정형화 및 비용 예측 모델에 관한 연구「정보처리논문지」 제3권 제4호 (1996), pp.846-854.

[2] 문양선, 유철중, 장옥배, "인지심리 이론을 반영한 객체지향 설계 및 프로그래밍 스타일 지침,"「정보과학회논문지(B)」, 제25권 3호 (1998), pp.530-542.

[3] 이상석, 윤민석, "통계처리용 소프트웨어 패키지의 품질 비교에 관한 연구,"「품질경영학회지」, 제27권 1호 (1999), pp.195-210.

[4] 이종무, 정호원, "AHP를 이용한 소프트웨어 내부 품질특성의 선정 방법,"「정보과학회논문지(B)」, 제24권 6호 (1997), pp.640-649.

[5] 이주헌,「실용 소프트웨어 공학론」, 법영사, 1993.

[6] 최영근, 허계범,「객체지향 소프트웨어 공학」, 한국실리콘, 1998.

[7] Boehm, B.W., J.R. Brown, M. Lipow, G.J. MacLeod, and M.J. Merritt, *Characteristics of Software Quality*, North-Holland, 1978.

[8] Booch, G., *Object-Oriented Analysis and Design*, 2nd edi. Benjamin Cummings, 1994.

[9] Coad, P. and E. Yourdon, *Object-Oriented Analysis*, 2nd Edi. Prentice-Hall, 1991.

[10] DeMarco, T., *Structured Analysis and System Specification*, Prentice-Hall, 1979.

[11] Dromey, R.G., "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, Vol.21, No.2(1995), pp.146-162.

[12] Fichman, R.G. and C.F. Kemerer, "Object-Oriented and Conventional Analysis and Design Methodologies," *Computer*, Vol.25, No.10 (1992), pp.22-39.

[13] Gane, T. and C. Sarson, *Structured Systems Analysis*, McDonnell Douglas, 1982.

[14] ISO 9126, *Information Technology-Software Product Evaluation-Quality Characteristics and Guidelines for Their Use*, ISO, 1991.

[15] ISO/IEC 9126-1 (draft), Information Technology Software Quality Characteristics and Metrics Part 1 : Quality Characteristics and Sub-Characteristics, ISO, 1996.

[16] Jacobson, I., *Object-Oriented Software Engineering*, Addison-Wesley, 1992.

[17] Jung H.W. and M.S. Yoon, "A Software Product Quality Evaluation and Resource Allocation Model," *5th European Conference on Software Quality*, Conference Proceeding (1996), pp.286-294.

[18] Kitchenham, B. and S.L. Pfleeger, "Software Quality: The Elusive Target," *IIIE Software*, January (1996), pp.12-21.

[19] Lejter, M., S. Meyers, and S.P. Reiss, "Support for Maintaining Object-Oriented Programs," *IEEE Transactions on Software Engineering*, Vol.18, No.12 (1992), pp.1045-1052.

[20] McCall, J.A., P.K. Richards, and G.F. Walters, *Factors in Software Quality (3 Vol.)*, RADC-TR-77-369, 1977.

[21] Miller, G.A., "The Magical Number Seven Plus or Minus Two : Some Limits on our Capacity for Processing Information," *Psychological Review*, Vol.63 (1956), pp.81-97.

[22] Pressman, R.S., *Software Engineering*, 4th ed., McGraw-Hill 1997.

[23] Rambaugh, J., et al., *Object Oriented Modeling and Design*, Prentice-Hall, 1991.

[24] Ross, D. and K. Schoman, "Structured Analysis for Requirements Definition," *IEEE Transactions on Software Engineering*, Vol.3, No.1 (1977), pp.6-15.

[25] Saaty, T.L., "A Scaling Method for Priorities in Hierarchical Structures," *Journal of Mathematical Psychology*, Vol.15 (1977), pp.234-

281.

[26] Saaty, T. L., The Analytic Hierarchy Process McGraw-Hill, 1980.

[27] Saaty, T. L., The Analytic Network Process RWS Publications, 1996.

[28] Stevens, W., G. Myers, and L. Constantine, "Structured Design," *IBM Systems Journal*, Vol.13, No.2 (1974), pp.115-139.

[29] Ward, P.T. and S.J. Mellor, *Structured Development for Real-Time Systems*, Yourdon Press, 1978.

[30] Wilde, N. and R. Huitt, "Maintenance Support for Object-Oriented Programs," *IEEE Transactions on Software Engineering*, Vol.18, No.12 (1992), pp.1038-1044.

[31] Wirfs, R.,B. Wilkerson, L. Weiner, Designing *Object-Oriented Software*, Prentice-Hall, 1990.

[32] Yoon, M.S., *Software Quality Evaluation Model Using The AHP-Developing a New Judgment Aggregation Method*, Ph.D. Dissertation, Graduate School of Korea University, 1997.

[33] Yourdon, E.N. and L.L. Constantine, *Structured Design*, Yourdon Press, 1978.