경영정보학연구
제13권 제1호
2003년 3월

사례의 수정최소화 기법에 의한 소프트웨어 프로젝트 네트워크 생성시스템

이 노 복*, 이 재 규**

Case-based Software Project Network Generation by the Least Modification Principle

No Bok Lee, Jae Kyu Lee

Software project planning is usually represented by a project activity network that is composed of stages of tasks to be done and precedence restrictions among them. The project network is very complex and its construction requires a vast amount of field knowledge and experience. So this study proposes a case-based reasoning approach that can generate the project network automatically based on the past cases and modification knowledge.

For the case indexing, we have adopted 17 factors, each with a few alternative values. A special structure of this problem is that the modification effort can be identified by each factor independently. Thus it is manageable to identify 85 primitive modification actions (add and delete activities) and estimate its modification efforts in advance. A specific case requires a combination of primitive modifications.

Based on the modification effort knowledge, we have adopted the Least Modification approach as a metric of similarity between a new project and past cases. Using the Least Modification approach and modification knowledge base, we can automatically generate the project network. To validate the performance of Least Modification approach, we have compared its performance with an ordinary minimal distance approach for 21 test cases. The experiment showed that the Least Modification approach could reduce the modification effort significantly.

* 국방과학연구소 팀장
** 한국과학기술원 테크노경영대학원 교수

Ⅰ. 서 론

효과적으로 소프트웨어 프로젝트를 관리하기 위하여 오늘날 프로젝트 관리자들에 의하여 직면하고 있는 주요한 문제는 정확하게 새로운 프로젝트의 특성을 고려하여 소프트웨어 프로젝트의 계획을 작성하는 일이다[21]. 정확하게 프로젝트 계획을 작성하는 작업은 새로운 프로젝트의 특성과 개발자들의 능력 등이 고려되어야 하기 때문에 많은 도메인 지식이 필요하며 매우 어려운 일이다.

이러한 문제를 해결하기 위하여 일부 프로젝트 개발방법론이나 시스템에서는 일반적이고 표준화된 템플릿을 제공하고 있으나, 프로젝트 관리자들이 프로젝트의 유형이나 특성들에 적합하게 수정(Customize)을 해야 한다.

이러한 수정작업을 위해 방법론들을 사용하는 노하우와 경험이 부족하고, 정확한 규칙이나 방법론이 존재하지 않기 때문에, 프로젝트 관리자들은 일반적으로 과거의 유사한 프로젝트를 찾게 되고, 그 유사한 프로젝트의 계획을 새로운 프로젝트의 특성에 맞게 수정하게 된다.

그러나 그들은 과거의 사례를 사용함에 있어, 거의 프로젝트 관리자들의 과거의 경험이나 주관적인 지식에 의존하기 때문에 프로젝트 관리자들의 주관적인 판단으로 체계적이지 못하고 일관성이 결여 되고있는 실정이다.

과거의 성공적인 정보시스템 개발 프로젝트의 결과인 관례(best practices), 경험, 그리고 소프트웨어 프로세스(software process) 등의 재사용이 새로운 소프트웨어 프로젝트의 계획을 수립하는 문제에 대한 오늘날 실행 가능한 해결책으로 여겨진다[6, 27]. 따라서 과거의 성공적인 프로젝트 사례들을 사례베이스에 축적하고 이를 조직차원에 재사용할 수 있는 메커니즘을 필요로 한다[23].

과거의 경험과 도메인 지식을 재사용하기 위해서는 이러한 경험과 지식들을 표현하고, 과거의 유사한 사례를 검색하고, 그 가장 유사한 사례를 새로운 프로젝트에 적용하기 위한 프레임웍이 필요하다. 사실 이러한 특성들이 본 논문에서 적용한 사례기반추론(CBR) 접근을 위한 근거이다.

본 논문에서는 새로운 프로젝트의 계획을 작성하기 위한 프로젝트 네트워크를 자동으로 생성하는 사례기반의 접근방법을 개발하였다. 유사한 사례를 사용하기 위한 가장 중용한 문제는 사례 베이스로부터 가장 유사한 사례들을 식별하고 검색하는 것이다. 이를 위하여 사례들을 분류하고 식별을 용이하게 하기 위하여 소프트웨어 프로젝트 네트워크의 구조에 영향을 주는 요소들(factors)을 식별하고, 이 요소는 사례 인덱싱, 사례 검색, 그리고 사례 수정을 위한 기준(criteria)으로 적용하였다.

가장 유사한 과거의 사례를 선정하기 위하여, 유사한 사례를 새로운 프로젝트에 적용하기 위해 요구되는 노력을 예측 하는 수정노력(modification effort) 기반 사례검색 방법을 제시하였다. 선정된 가장 유사한 사례를 새로운 프로젝트에 적용하기 위해서 가장 유사한 사례와 새로운 프로젝트간의 상이한 요소들을 식별하고, 이 상이한 요소들에 관련된 도메인 지식을 사용하는 지식(knowledge)기반 사례수정 방법을 제안하였다.

또한, 본 연구에서 제안한 수정노력 최소화 방법에 의한 사례의 수정은 프로젝트 특징의 유사도에 의해 선택된 사례를 수정하는 방법보다 통계적으로 유의하게 더 효율적임이 실험적으로 검증되었다.

Ⅱ. 소프트웨어 프로젝트 계획을 위한 사례기반 추론 접근

2.1 소프트웨어 프로젝트 계획의 문제점

소프트웨어 프로젝트의 계획은 정보시스템을

개발 할 때 수행되어야 하는 활동(activities)들이 식별되고, 우선순위에 따라 활동들의 순서가 정해지는 과정(process)으로 정의 되며, 일반적으로 간트 차트(Gantt chart)나 활동 네트웍(activity network) 등으로 표현된다[12, 24]. 프로젝트 관리자는 프로젝트 초기단계에서 프로젝트의 계획을 작성하기 위해 개발 생명주기(life cycle)를 선택하고, 요구되는 활동들을 식별하고, 우선순위에 따라 그 활동들의 순서를 정하고 명확히 정의된 단계(phase)별로 활동들을 조직화 한다[25, 28].

소프트웨어 프로젝트의 계획을 작성하는 작업은 프로젝트에 적용된 프로젝트의 유형, 적용기술, 개발자의 능력 등 프로젝트 수행에 필요한 특성을 고려하여 프로젝트 수행에 필요한 활동(activity) 및 태스크(task)들을 식별하고, 이들의 순서를 정해야 한다[7, 13, 25].

이러한 프로젝트의 특성들에 따라 프로젝트 수행에 필요한 활동 및 태스크들을 식별하는 일은 많은 도메인 지식이 필요하며, 많은 시간과 노력을 필요로 하는 매우 어려운 작업이다.

이러한 문제점들을 극복하기 위하여 소프트웨어 개발 프로세스 관련 표준[12, 17], 연구[9, 11, 13, 23] 그리고 방법론[20, 25]들이 출현하고 있으나, 일반적인 템플릿(template)과 추상적인 테일러링(tailoring) 지침(guide)으로 한 조직에서 실질적인 프로젝트에 맞는 계획을 작성하는데 노하우(know-how)와 경험이 부족하기 때문에 어려운 실정이다[23].

2.2 사례기반 추론 접근을 사용하는 근거

프로젝트 관리자들은 새로운 프로젝트의 계획을 수립하기 위해서 일반적으로 과거의 유사한 사례를 찾고, 자기의 경험과 도메인 지식을 사용해서 그 유사한 사례를 새로운 프로젝트의 특성에 맞게 수정하게 된다. 이 지식은 과거의 경험과 과거의 유사한 사례를 참조해서 획득하게 된다.

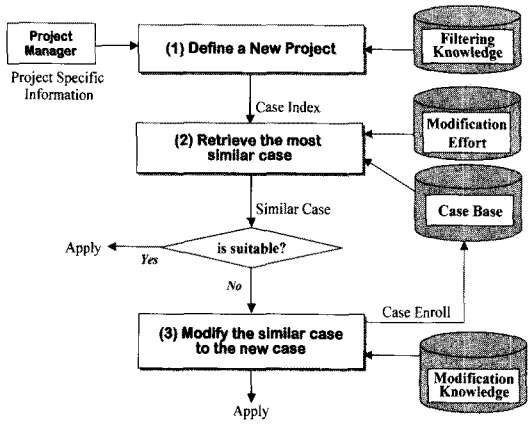
그러나 유사한 프로젝트 임에도 불구하고 각 프로젝트 관리자에 따라 다르게 프로젝트 계획을 작성한다면, 조직차원에서 성공적인 사례를 공유할 수 없을 것이다. 조직차원에서 과거의 성공적인 프로젝트의 사례를 사례베이스로 구축하여 이들을 재사용할 수 있다면, 조직차원의 유용한 자산이 될 것이다. 따라서 과거의 경험과 과거의 유사한 사례를 분류하고 관리하여 재사용할 수 있는 메카니즘이 필요하다.

사례기반 추론(CBR) 기법은 가장 유사한 사례로부터 해결책을 찾고, 그 유사한 사례와 새로운 사례와의 차이를 고려해서 그 해결책을 수정하는 방법을 제시한다[6, 18]. 지금까지 제조, 건축 분야 등의 건설공정[16, 19] 분야에서 사례기반 추론 기법을 적용하여 프로젝트 계획의 작성에 관한 연구가 이루어져 왔으며, 소프트웨어 컴포넌트의 재사용[5]과 비용 예측[8, 26] 등 소프트웨어 공학 분야에서도 사례기반 추론 기법을 적용한 연구가 시도 되고 있다. 이러한 관점에서 볼 때, 정확한 방법론이나 규칙이 없는 소프트웨어 개발 프로젝트 네트워크를 작성하기 위해서 사례기반 추론 기법을 적용하는 것은 현실적으로 실현 가능한 해결책으로 보여 진다.

2.3 소프트웨어 프로젝트 계획 수립 절차

사례기반 추론 기법을 적용하여 소프트웨어 프로젝트의 계획을 작성하기 위해서는 새로운 프로젝트의 특성에 따라 과거의 가장 유사한 사례를 검색해야 하며, 검색된 가장 유사한 사례는 새로운 프로젝트의 특성에 맞게 수정되어야 한다. 본 연구에서는 프로젝트 관리자가 과거의 유사한 사례를 사용하여 새로운 프로젝트의 계획을 작성하기 위한 프로젝트 네트워크의 작성 절차는 <그림 1>과 같으며, 본 연구에서는 이러한 절차를 지원하는 시스템인 SPP(Software Project Planner)를 구현하여 실질적으로 적용할 수 있

도록 하였다.



<그림 1> 사례기반 프로젝트 계획 수립 절차

2.3.1 새로운 프로젝트의 정의

새로운 소프트웨어 프로젝트를 정의하는 단계로서,프로젝트 관리자는 새로운 프로젝트의 특성 즉, 프로파일(profile)을 정의해야 한다. 프로젝트 이름, 프로젝트 시작 및 종료일 그리고 새로운 프로젝트의 계획을 작성하는 프로젝트 관리자 등 프로젝트에 관련 기본 정보를 입력한다.

본 연구에서 프로젝트의 프로파일을 정의하기 위하여 소프트웨어 개발비용 산정 모델, 소프트웨어 개발 생산성 향상 모델, 사례 분석, 그리고 도메인 전문가들을 통해서 17개 요소와 그 요소가 갖을 수 있는 값들을 식별하였다. 이러한 요소 값들의 조합은 새로운 프로젝트의 인덱스가 되어 효과적으로 유사한 사례를 검색하기 위한 질의로 사용된다.

또한 프로젝트 관리자가 새로운 프로젝트의 프로파일 정보를 입력할 때 그 요소들의 값들 간에는 서로 중복하여 프로젝트 네트웍의 구조에 영향을 줄 수 있기 때문에 서로 영향을 주는 요소의 값은 배제해야 한다.

본 연구에서는 프로젝트 프로파일 정보를 입력할 때, 지식(filtering knowledge)에 의해서 각 요소간의 상호 영향(interaction)을 검토하여 영향을 주는 요소를 정제(filter)할 수 있도록 하였다.

2.3.2 유사한 사례 검색

새로운 프로젝트의 프로파일은 사례를 검색하기 위한 인덱스로 과거의 사례들의 프로파일과 비교되어 유사한 사례를 선정하는데 사용된다. 본 연구에서는 프로젝트 유형에 따라 프로젝트 네트워크에 영향을 주는 요소가 다르기 때문에 2단계 절차를 적용하였다.

첫째 단계에서는 새로운 프로젝트의 특성인 프로젝트 유형(project type)에 의해서 사례 그룹을 선정하고, 둘째 단계에서는 선정된 사례 그룹 내에서 새로운 프로젝트와 비교하여 과거의 사례들의 수정노력(modification effort)이 가장 작은 값을 갖는 사례를 가장 유사한 사례로 선정한다.

새로운 프로젝트와 유사한 사례들과의 유사도 측정에 관한 방법은 다음 4장에서 자세하게 설명된다.

2.3.3 유사한 사례의 수정

선정된 유사한 사례가 새로운 프로젝트와 정확히 일치하지 않는다면, 그 유사한 사례를 새로운 프로젝트의 특성에 맞게 수정해야 한다. 유사한 사례를 수정하기 위해서는 우선 새로운 프로젝트의 프로파일과 유사한 사례의 프로파일을 비교하여 각 요소별 값을 비교하여 상이한 값을 갖는 요소들을 식별하여 프로파일과의 차이점을 해결해야 한다.

이러한 차이점은 각 상이한 요소에 대해 새로운 프로젝트의 요소의 값을 유사한 사례의 요소 값으로 대체 시키는 방법으로 유사한 사례를 새로운 프로젝트의 특성에 맞게 수정하도록 하였다.

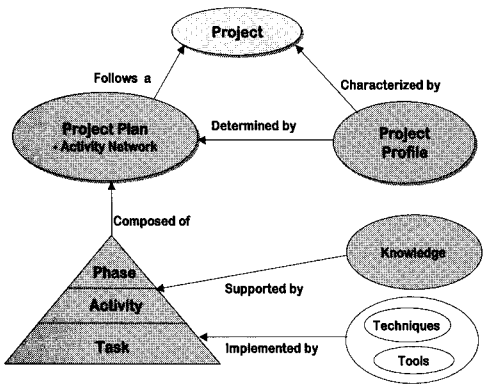
새로운 프로젝트의 요소의 값을 유사한 사례의 요소의 값으로 대체 시킴에 따른 대응되는

수정지식(modification knowledge)에 의해서 유사한 프로젝트의 네트워크를 수정한다. 수정지식은 규칙(rule), 프레임(frame)들로 구성되며, 프로젝트 네트워크의 구조 즉, 활동(activity)과 태스크(task)들을 추가, 수정, 삭제하는 데에 활용된다.

Ⅲ. 소프트웨어 프로젝트 네트워크에 영향을 주는 요소

3.1 요 소

사례기반 접근방법을 이용하여 소프트웨어 프로젝트의 네트워크를 생성하기 위해서는 과거의 사례들을 일정한 형식으로 표현해야 한다. 본 논문에서는 사례를 프로젝트의 프로파일, 프로젝트 네트워크, 그리고 프로젝트 네트워크을 관리하기 위한 지식 등으로 구분하여 <그림 2>와 같이 구성하였다.



<그림 2> 프로젝트 사례의 구성 요소

프로젝트의 프로파일 정보는 프레임 형식으로 표현되며, 프로젝트 네트워크는 계층적 및 네트워크 특성에 따라 단계(phase), 활동(activity), 그리고 태스크(task)로 표현된다.

본 연구에서는 프로젝트 계획의 각 단계, 활동, 그리고 태스크들을 [2]와 같은 방법으로 프레임 기반으로 표현 하였다.

지식은 프로젝트 조직이나 프로젝트 관리자에 따라서 활동이나 태스크를 표현하는 수준이 다를 수 있지만[1], 소프트웨어 프로세스 관련 표준, 과거의 사례들을 분석하여 프로젝트 네트워크에 영향을 주는 요소(factor)에 관련된 지식을 정리하여 프로젝트 계획의 각 단계, 활동, 그리고 태스크들을 수정할 수 있도록 별도의 언어를 정의하여 규칙 형식으로 표현하였다.

소프트웨어 프로젝트의 네트워크 구조는 프로젝트에서 적용하는 특성들에 의하여 결정되기 때문에 프로젝트 네트워크에 영향을 주는 요소를 식별하는 것이 중요하다.

본 연구에서는 과거에 수행되었던 60개 사례들의 특성을 분석하고, 소프트웨어 공학 분야의 개발 비용산정 모델[4, 8], 개발 생산성 향상 모델[10, 14, 15] 등으로부터 소프트웨어 프로젝트 네트워크에 영향을 줄 수 있는 요소들을 식별하였다.

그러나 소프트웨어 프로젝트의 네트워크는 프로젝트 유형(project type) 즉, 'Strategic Planning, Application Maintenance, Reengineering, Data Warehouse, New Development, Package Customization' 등에 따라 영향을 주는 요소들이 다르다. 따라서 프로젝트 유형에 따라 사례들을 그룹 하여 구분할 수 있도록 분류하였다.

이러한 요소들은 사례를 분류하고 식별하기 위한 인덱스로 사용되며 유사한 사례를 검색하고, 그 유사한 사례를 수정하기 위한 지식을 생성하는 기준(criteria)으로 프로젝트의 프로파일 정보가 된다. 이 요소들은 크게 프로젝트에 적용되는 기술, 개발환경과 개발환경에 대한 개발자들의 경험 정도로 구분된다.

본 연구에서는 <그림 3>과 같이 프로젝트 유형이 'New Development'인 프로젝트 네트워크에 영향을 주는 요소로 제한 하였으며, 소트웨어 프로젝트 네트워크의 구조 즉, 활용들이나 태스크들을 수정하는 지식으로 표현이 가능한 17개

요소들을 추출하고 그 요소가 갖을 수 있는 값들을 정의 하였다.

Project Type: {Strategic Planning | Application Maintenance | Reengineering | Data Warehouse *New Development* | Package Customization}
1. **Analysis of Current System** {Necessary | Unnecessary
2. **BPR** {Necessary | Unnecessary}
3. **Requirements Clarity:** {Structured | Unstructured}
4. **Adoption of New Technology:** {Proven | New}
5. **Data Migration:** {Necessary | Unnecessary}
6. **System Architecture:** {Terminal/Host | Client/Server | Web/Server}
7. **Development Methodology:** {Structured | Object oriented}
8. **Prototyping:** {GUI | Performance | Feasibility | Unnecessary}
9. **CASE Tool Use:** {Necessary | Unnecessary}
10. **Reuse:** {Necessary | Unnecessary}
11. **Staff Experience of Application Domain:** {Experience | Not Experience}
12. **Staff Experience of Methodology:** {Experience | Not Experience}
13. **Staff Experience of OS Platform:** {Experience | Not Experience}
14. **Staff Experience of CASE Tool:** {Experience | Not Experience}
15. **Staff Experience of Programming Language:** {Experience | Not Experience}
16. **Staff Experience of Middleware** {Experience | Not Experience}
17. **Staff Experience of DBMS:** {Experience | Not Experience}

<그림 3> 영향 주는 요소

3.2 수정 지식

수정지식은 유사한 사례의 프로파일과 새로운 프로젝트의 프로파일 사이의 상이한 요소들 중에서, 각 상이한 요소별로 유사한 사례의 요소의 값을 새로운 프로젝트의 요소의 값으로 변경시킴에 따라 유사한 사례의 프로젝트 네트워크를 새로운 프로젝트의 네트워크로 수정시키는 도메인 지식이다. 식 (1)의 M_i는 17개 모든 요소 i 에 대해 과거 사례의 프로파일의 j 값을 새로운 프로젝트 프로파일의 j값으로 대체 시키는 경우에 따른 수정 지식들의 집합이며, 식 (2)의 r_{ijk}는 특정 i 요소에서 과거 사례 프로파일의 j 값을 새로운 프로젝트 프로파일의 k 값으로 대체시키는데 요구되는 수정 지식이다.

$$M_i = \{ r_{ijk} \} \quad for\ all\ j,\ k \tag{1}$$

$$r_{ijk};\ a_{ij}^{past} \rightarrow a_{ik}^{new} \tag{2}$$

'System Architecture' 요소의 경우에 과거 사례의 요소 값과 새로운 프로젝트의 요소 값들간에 필요한 지식들의 식별자들을 <표 1>에서 보여주고 있다.

특히 'Rule 6-8'는 과거 사례에서는 'Client/Server'구조를 적용하였으나 새로운 프로젝트에서는 'Web/Server'구조를 채택할 경우에 과거사례의 요소 값을 새로운 프로젝트의 요소의 값으로 대체 함에 따라 과거 사례의 프로젝트 네트워크를 수정하기 위해 필요한 지식으로 <그림 4>와 같이 표현된다.

<표 1> 과거사례와 새로운 사례간 요소별 지식의 예

Past Case \ New Case		Terminal/Host	Client/Server	Web/Server
		1	2	3
Terminal/Host	1	Rule 6-1	Rule 6-4	Rule 6-7
Client/Server	2	Rule 6-2	Rule 6-5	Rule 6-8
Web/Server	3	Rule 6-3	Rule 6-6	Rule 6-9

Rule 6-8(r_{623})
Past: *System_Architecture Client/Server*
New: *System_Architecture Web/Server*
Action: *DELETE_TASK Select_GUI_Prototyping_Tool of GUI_Prototyping*
ADD_TASK Design_Web_page of Preliminary_Design
ADD_TASK Define_multimedia_source of Preliminary_Design
DELETE_TASK Design_client_modules of Detail_design
ADD_TASK Create_Multimedia_content of Construction_and_Unit_Testing
DELETE_TASK Code_and_Compile_Client_Program of Construction_and_Unit_Testing

<그림 4> 수정지식의 예

이 지식은 프로젝트 네트워크를 수정하는 규칙으로 단계, 활동 그리고 태스크 수준에서 추가(ADD_PHASE, ADD_ACTIVITY, ADD_TASK), 삭제(DELETE_PHASE, DELETE_ACTIVITY, DELETE_TASK) 그리고 수정(UPDATE_PHASE, UPDATE_ACTIVITY, UPDATE _TASK)하는 언어를 정의하여 표현 하였다.

Ⅳ. 수정노력 기반 사례 검색

4.1 사례 인덱싱

사례의 인덱스는 사례를 분류하여 식별할 수 있는 식별자 이다. 새로운 프로젝트에 가장 과거 유사한 사례를 선정하기 위해서는 과거의 사례들을 유일하게 식별할 수 있으면서 새로운 프로젝트와 과거 사례들과의 유사성을 용이하게 측정할 수 있어야 한다[22].

3장에서 언급했듯이 본 연구에서는 소프트웨어 프로젝트 네트워크에 영향을 주는 요소들의 값들을 조합해서 인덱스를 생성하여 사례를 식별할 수 있도록 하였다. 프로젝트 관리자가 새로운 프로젝트의 프로파일 정보를 입력함으로써 각 요소별 값들의 조합 자체가 유사한 사례를 검색하기 위한 질의어가 되며, 새로운 사례로 등록될 때 인덱스가 되어 사례베이스에 분류되어 등록된다.

또한, 프로젝트 관리자가 새로운 프로젝트의 프로파일 정보를 입력할 때, 각 요소간에는 상호영향(interaction effect)이 있기 때문에 서로 중복하여 프로젝트 계획의 구조에 영향을 주어 활동이나 태스크들이 중복되거나 서로 모순을 야기 할 수 있다. 따라서 프로젝트 관리자가 프로파일 정보를 입력할 때, 지식(filtering knowledge)에 의해서 각 요소들의 값들 간에 상호 영향을 줄 수 있는 값들은 사전에 선택할 수 있는 가능성을 배제하게 함으로써 각 요소의 값들간의 일관성을 유지하도록 하였다.

4.2 수정노력

새로운 프로젝트의 네트워크를 작성하기 위해서는 유사한 사례를 새로운 프로젝트의 특성에 적합하게 수정 되어야 한다. 따라서 새로운 프로젝트 네트워크에 가장 유사한 사례를 선정할 수 있다면 수정노력(modification effort)이 최소화 될 것이다.

본 연구에서는 실질적으로 각 요소별 값이 달라짐에 따라 프로젝트 네트워크 구조가 달라지기 때문에, 과거 사례의 프로젝트 네트워크를 새로운 프로젝트에 적용하기 위해 과거 사례의 프로젝트 네트워크를 수정하는데 요구되는 노력을 유사성 측정의 기준으로 하였다.

수정노력은 식 (3)과 같이 모든 요소 i에 대해 과거 사례의 요소 값(j)을 새로운 프로젝트의 요소 값(k)으로 대체 시킴에 따른 과거 사례의 프로젝트 네트워크를 수정하는데 요구되는 노력들의 합으로, 새로운 프로젝트의 프로파일과 과거 사례 프로파일을 비교하여 상이한 요소 값들 쌍간의 수정 노력들의 합으로 측정된다.

식 (4)의 e_{ijk}는 특정 요소 i에 대해 과거 사례

<표 2> 사례간 수정노력의 예

Past Case \ New Case	Terminal/Host	Client/Server	Web/Server
Terminal/Host	0	24	25
Client/Server	24	0	6
Web/Server	25	6	0

의 요소 값(j)을 새로운 프로젝트의 요소 값(k)으로 대체 시킴에 따른 노력(effort)을 표현한 것으로, 과거 사례의 프로젝트 네트워크에 새롭게 추가되는 활동들의 수(*NOAT*: *number of add task*), 삭제되는 활동들의 수(*NODT*: *number of delete task*), 그리고 수정되는 활동들의 수(*NOUT*: *number of update task*)를 합산하여 계산된다.

$$E_i(a_{ij}^{past}, d_{jk}^{new}) = \sum_{i=1}^{n} w_i(e_{ijk})$$

$$for\ some\ j,\ k,\ w_i = 1 \quad (3)$$

$$e_{ijk} = NOTE + NODT + NOUT \quad (4)$$

'System Architecture' 요소의 경우에 과거의 사례와 새로운 프로젝트의 요소 값들간의 모든 경우에 대한 수정노력을 <표 2>와 같이 표현할 수 있다.

과거 사례의 'Terminal/Host'를 새로운 프로젝트의 'Client/Server'와 'Web/Server'로 각각 전환함에 따라 과거의 사례를 새로운 프로젝트로 적합 시키기 위해서는 각각 24, 25만큼의 노력이 요구되며, 과거 사례의 'Client/Server'를 새로운 프로젝트의 'Web/Server'로 적합 시키기 위해 필요한 지식인 <표 1>의 'Rule 6-8'과 관련된 그림 4의 지식에 의해 수정노력은 6(*NOAT* = 3, *NODT* = 3) 만큼의 노력이 요구된다는 것을 보여주고 있다.

4.3 수정노력기반 사례 검색

새로운 프로젝트와 가장 유사한 사례를 선정하기 위해서는 과거의 사례들 중에서 새로운 프로젝트의 네트워크를 생성하는데 요구되는 노력이 가장 작은 사례를 선정해야 한다.

가장 작은 수정 노력이 요구되는 사례를 선정하기 위해서는 새로운 프로젝트의 프로파일 정보와 과거 사례들의 프로파일 정보를 비교하여 수정노력의 값을 계산해야 한다.

본 연구에서는 사례베이스로부터 유사한 사례를 검색하기 위하여 SPP(Software Project Planner)를 개발하였다. <그림 5>와 같이 정의된 새로운 프로젝트에 대한 유사한 사례를 선정하기 위하여, SPP를 사용하여 사례 베이스로부터 유사한 사례를 검색하였으며, 검색된 결과는 <그림 6>과 같이 10개의 사례들이 검색되었다.

검색된 10개의 사례들은 각 사례별로 수정노력의 값에 의해서 수정노력이 작은 순으로 우선순위(rank)가 정해진다. 또한, 새로운 프로젝트의 프로파일과 사례들의 프로파일을 비교하여 상이한 요소들의 수와 수정노력은 비례하지 않음을 보여 주고 있다. 이러한 현상은 본 연구에서 제안한 방법의 타당성을 검증하는데 언급되며 6장에서 상세하게 언급하였다.

<그림 6>의 유사한 사례들 중 Case 1, Case 2, Case 10의 수정노력은 각 요소별로 수정노력의 합산으로 계산되는데, 각 요소별 수정노력은 표 3과 같으며 Case 1의 수정노력의 값이 39로 3개의 사례 중에서 제일 적은 수정노력이 요구되어 가장 유사한 사례라고 예측 할 수 있다.

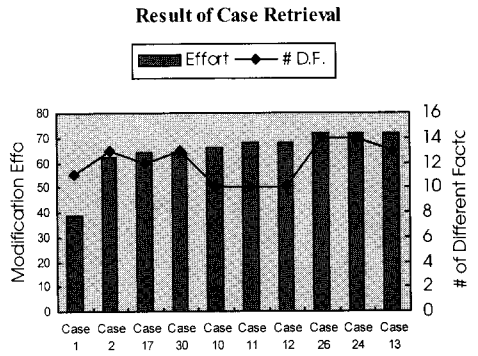
또한, <그림 6>에서는 유사한 사례의 수정노력의 값은 새로운 프로젝트와 유사한 사례들과의 상이한 요소들의 수(# of Different Factor)와

는 비례하지 않다는 것을 발견할 수 있다. 각 요소에 따라 네트워크에 영향을 주는 가중치 즉, 활동이나 태스크의 수가 다르기 때문이다.

따라서 새로운 프로젝트와 과거사례와의 단순한 요소들을 비교하여 유사성을 측정하는 방법과는 다른 결과가 발생한다는 것을 알 수 있다. 본 연구에서 제안한 수정노력 최소화 방법에 기반 한 유사한 사례를 검색하는 방법의 타당성에 관한 검증은 6장에서 구체적으로 기술할 것이다.

{{*Profile_of_New_Case*
IS-A: Project
Project_Definition:
(Project_Name: MMA_HRIS)
(Start-Date: 01/04/2000)
(End-Date: 11/11/2001)
(Project_Manager : No Bok Lee)
Project_Characteristics:
(Project_Type: New_Development)
(Analysis_of_Current_System: Unnecessary)
(Business Process Reengineering: Unnecessary)
(Requirements_Clarity: Unstructured)
(Adoption_of_Technology: New)
(Data_Migration: Unnecessary)
(System_Architecture: Web/Server)
(Development_Methodology: Object_oriented)
(Prototyping: GUI)
(CASE_Tool_Use: Necessary)
(Reuse: Unnecessary)
(Staff_Experience _of_Application_Domain:
HRM Not-Experience)
(Staff_Experience _of_Methodology:
Object_oriented Not-Experience)
(Staff_Experience _of_OS_Platform:
Guardian Not-Experience)
(Staff_Experience _of_CASE_Tool:
ROSE Not-Experience)
(Staff_Experience _of_Program-Language:
Java Not-Experience)
(Staff_Experience _of_Middleware:
WebLogic Not-Experience)
(Staff_Experience _of_DBMS:
Tandem_SQL/MP Not-Experience)}}

<그림 5> 새로운 프로젝트의 프로파일



<그림 6> 수정노력과 상이한 요소 수의 관계

<표 3> 요소별 수정 노력들

Affecting Factors	Case 1	Case 2	Case 10
Analysis of Current System	7	0	0
Business Process Reengineering	0	0	0
Requirements Clarity	0	9	0
Adoption of Technology	6	6	0
Data Migration	9	9	0
System Architecture	6	6	25
Development Methodology	0	15	15
Prototyping	0	0	15
CASE Tool Use	0	0	0
Reuse	0	6	0
Staff Experience of Application	1	1	1
Staff Experience of Methodology	3	3	3
Staff Experience of OS Platform	1	1	1
Staff Experience of CASE Tool	2	2	2
Staff Experience of Program Language	1	1	1
Staff Experience of Middleware	1	1	1
Staff Experience of DBMS	2	2	2
Total Effort	39	62	66

V. 지식기반 사례수정

5.1 새로운 프로젝트와 유사한 사례의 차이점 식별

가장 유사한 사례를 선정한 후에는 새로운 프로젝트의 특성에 맞게 프로젝트 네트워크를 수정되어야 한다. 가장 유사한 사례를 수정하기 위해서는 우선 선정된 유사한 사례의 프로파일과 새로운 프로젝트의 프로파일을 비교하여 차이점을 식별하여 이 차이점을 해결해야 한다.

{(Different_Factors_for_ Modification IS-A : Difference_of_profile	Knowledge
(Analysis_of_Current_System: Unnecessary Necessary)	Rue 1-3
(Adoption_of_Technology: New Proven)	Rue 4-2
(Data_Migration: Unnecessary Necessary)	Rue 5-3
(System_Architecture: Web/Server Client/Server)	Rue 6-8
(Staff_Experience_of_Application_Domain: Not_Experience Not_Experience)	Rue 11-1-4
(Staff_Experience_of_Methodology: Experience Not_Experience)	Rue 12-0-3
(Staff_Experience_of_OS_Platform: Experience Not_Experience)	Rue 13-1-3
(Staff_Experience_of_CASE_Tool: Experience Not_Experience)	Rue 14-0-3
(Staff_Experience_of_Program_Language: Experience Not_Experience)	Rue 15-1-3
(Staff_Experience_of_Middleware: Not_Experience Not_Experience)	Rue 16-1-4
(Staff_Experience_of_DBMS: Experience Not_Experience) }}	Rue 17-1-3

<그림 7> 상이한 요소와 관련 지식

이 차이점을 해결하기 위해서는 새로운 프로젝트와 비교해서 유사한 사례의 각 상이한 요소의 값을 새로운 프로젝트의 요소의 값으로 대체함에 따른 각 요소별로 관련된 수정지식을 사용하여 유사한 사례의 프로젝트 네트워크를 수정하게 된다.

<그림 7>에서는 새로운 프로젝트(<그림 5>)와 수정노력이 최소인 가장 유사한 사례인 그림 6의 Case 1간에 11개의 상이한 요소가 있으며, 이 상이한 한 요소별로 Case 1의 요소 값을 새로운 사례의 요소값으로 전환과 대응된 수정 지식들을 보여주고 있다.

5.2 사례수정

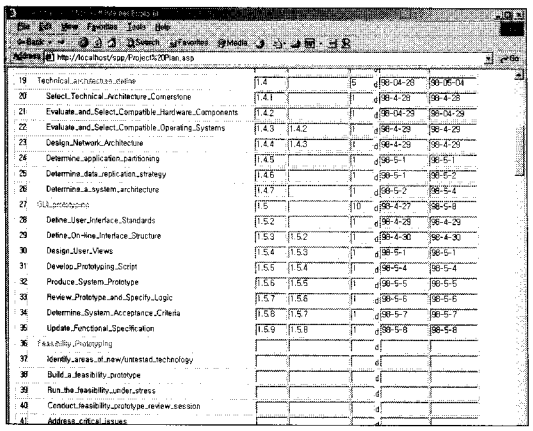
유사한 사례를 새로운 프로젝트에 적합 시키기 위해서는 <그림 7>과 같이 상이한 각 요소들에 관련된 지식(*Rule 1-3, Rule 4-2, Rule 5-3, Rule 6-8, Rule 11-1-4, Rule 12-0-3, Rule 13-1-3, Rule 15-1-3, Rule 16-1-4, Rule 17-1-3*)에 의해서 유사한 사례의 소프트웨어 프로젝트의 네트워크를 수정해야 한다.

Rule 5-3(r_{512})
Past: *Data_Migration* *Necessary*
New: *Data_Migration* ***Unnecessary***
Action: **DELETE_ACTIVITY** ***Conversion_Design*** *of Design*
DELETE_TASK ***Create_Data_Acquisition_Strategy*** *of Conversion_Design*
DELETE_TASK ***Define_Data_Conversion_Sources*** *of Conversion_Design*
DELETE_TASK ***Define_Conversion_Rules_and_Logic*** *of Conversion_Design*
DELETE_TASK ***Design_Conversion_Transactions*** *of Conversion_Design*
DELETE_TASK ***Determine_Conversion_Procedures_and_Controls*** *of Conversion_Design*
DELETE_TASK ***Convert_Static_Data_and_Acquire_New_Data*** *of Installation_and_Handover*
DELETE_TASK ***Convert_Dynamic_Data*** *of Installation_and_Handover*
DELETE_TASK ***Evaluate_Results_of_Conversion*** *of Installation_and_Handover*

<그림 8> 상이한 요소에 관련된 수정지식

11개의 상이한 요소 중에서 'Data_Migration' 요소에서는 유사한 사례에서는 기존의 시스템에서 'Data Migration' 작업을 수행하였으나, 새로운 프로젝트에서는 'Data Migration' 작업을 수행할 필요가 없는 경우로 <그림 8>과 같은 활동과 태스크들을 삭제하는 지식에 의해서 유사한 사례의 프로젝트 네트워크를 수정하게 된다.

새로운 프로젝트와 유사한 사례와의 상이한 요소를 식별한 후, SPP를 사용하여 새로운 프로젝트 네트워크를 작성하기 위하여 <그림 7>의 각 요소별 수정지식을 사용하여 유사한 사례인 소프트웨어 프로젝트의 네트워크를 수정한 결과는 <그림 9>와 같다.



<그림 9> 수정된 프로젝트 계획

Ⅵ. 사례의 수정최소화 기법의 검증

6.1 실험 방법의 설계

사례기반 접근 시스템에서 가장 중요하게 해결해야 하는 문제는 새로운 프로젝트에 적용하기 위해 최소의 수정노력이 요구되는 사례를 선정하는 일이다. 따라서 새로운 프로젝트와 과거의 사례들간의 유사도 측정을 위해 본 연구에서 제안한 수정최소화 방법(LMP: least modification principle)과 일반적으로 적용되고 있는 일치되는 요소의 수로 유사도를 측정하는 프로젝트 특징의 매치 방법(FM: factor match)을 비교하여, 수정 최소화 방법에 의한 사례기반 소프트웨어 프로젝트 네트워크를 작성하는 방법을 실험을 통해서 유용성을 검증하였다.

FM 방법을 LMP 방법과 비교하기 위해서는 비교하는 단위가 같아야 한다. FM 방법에 의한 유사도 측정은 유사한 사례와 새로운 프로젝트와의 일치하는 요소의 수로 측정하기 때문에 실질적으로 LMP 방법과 비교하기 위해서 FM 방법에 의해 선정된 사례의 유사도 값을 수정노력 값으로 환산하였다.

사례의 수정최소화 기법에 의한 사례기반 소프트웨어 프로젝트의 계획을 작성하는 방법을 검증하기 위하여, 과거에 수행되었던 68개 사례들 중에서 임의로 21개 테스트 사례를 선정하였다.

21개의 테스트 사례는 새로운 프로젝트 및 과거의 사례로 사용하여, 실험의 편리를 위하여 각 유사도 측정 방법으로 21회에 걸쳐서 유사한 사례들을 검색하고, 유사한 사례들의 수정노력들을 비교하여 통계적으로 검정(paired t-test)하였다.

6.2 두 방법에 의한 유사한 사례의 검색

LMP 방법과 FM 방법을 비교하기 위해서는 각 방법을 적용하여 유사한 사례를 검색하여 그 결과를 비교하는 실험이 필요하다. SPP에 의해서 각 새로운 프로젝트별로 유사한 사례를 검색하고 비교하는 기법은 [3]에서 제공하는 사례기반 접근 시스템을 검증하기 위한 기법을 적용하였으며, 본 실험을 위해 적용하는 알고리즘은 다음과 같다.

1) Copy the 21 test cases of the case base to a new project case list.
2) Until the new case list is not empty:
   a. Point to the first new case.

b. Remove the test case corresponding to the new case.
c. Input profile of the new case to the SPP
d. Retrieve and rank the test cases by using the SPP
e. Select the most similar test case(s)
   If test is designed for Least Modification Principle, add it to the LMP-case list.
   Else test is designed for Factor Match, add it to the FM-case list.
f. Return the removed test case back into the case base.
g. Delete the new case from the new case list.
h. Go to step a.

3) Compare the modification efforts in the LMP-case list and the FM-case list.
4) Determine whether the Least Modification Principle technique is better.

각 방법을 적용하여 21개 새로운 프로젝트에 대한 유사한 사례를 검색한 결과는 <표 4>와 같으며, 이 결과에 의하면 각 방법에 의해 검색된 유사한 사례들이 다르다는 것을 알 수 있다.

<표 4> 두 방법에 의해 검색된 유사한 사례들

New Case	Similar Cases						
	LMP		FM				
New Case 1	Case 2(38)	Case 30(38)	Case 17(50)	Case 24(41)	Case 30(38)	Case 18(58)	Case 2(38)
New Case 2	Case 1(38)		Case 3(39)	Case 20(42)			
New Case 3	Case 5(15)		Case 4(18)	Case 5(15)			
New Case 4	Case 5(15)		Case 3(18)	Case 5(15)			
New Case 5	Case 3(15)	Case 4(15)	Case 3(15)	Case 4(15)			
New Case 6	Case 11(14)		Case 11(14)				
New Case 7	Case 12(9)		Case 10(14)	Case 12(9)			
New Case 8	Case 11(9)		Case 11(9)				
New Case 9	Case 28(32)	Case 27(32)	Case 26(35)	Case 27(32)	Case 12(39)		
New Case 10	Case 20(16)		Case 20(16)				
New Case 11	Case 23(21)		Case 22(22)	Case 21(22)	Case 17(22)		
New Case 12	Case 22(11)		Case 22(11)				
New Case 13	Case 17(16)	Case 21(16)	Case 17(16)	Case 21(16)			
New Case 14	Case 22(13)		Case 20(16)				
New Case 15	Case 19(11)		Case 19(11)				
New Case 16	Case 17(18)		Case 17(18)				
New Case 17	Case 30(15)		Case 27(22)	Case 30(15)			
New Case 18	Case 28(15)		Case 27(16)				
New Case 19	Case 28(13)		Case 28(16)				
New Case 20	Case 27(13)		Case 28(15)	Case 27(13)			
New Case 21	Case 24(15)		Case 27(23)	Case 24(15)			

6.3 실험 결과

앞 절에서 각 새로운 프로젝트에 대하여 사례의 검색 과정에서 두 방법을 적용한 결과인 유사한 사례들과 그 사례들의 수정노력들을 살펴보았다. LMP 방법에 의한 유사도 측정은 유사한 사례를 수정하여 새로운 프로젝트에 적합 시키는 노력의 값으로 측정한다.

그러나 FM 방법에 의한 유사도 측정은 유사한 사례와 새로운 프로젝트와의 일치하는 요소의 수로 측정하기 때문에 실질적으로 LMP 방법과 FM 방법을 비교하기 위해서는 FM 방법에 의해 검색된 유사한 사례도 새로운 프로젝트에 맞게 수정하는 데 요구되는 노력으로 환산해야 한다. <표 4>의 FM 방법에 의해 각 새로운 프로젝트별로 검색된 유사한 사례들의 값은 그 유사한 사례를 수정하여 새로운 프로젝트로 적합 시키기 위해 요구되는 노력이다.

각 새로운 프로젝트에 대하여 검색된 유사한 사례가 2개 이상인 경우에는 그 유사한 사례들의 수정노력을 산술평균 하였다. 두 방법의 평균 수정노력(MME)은 21회에 걸친 유사한 사례들의 평균 수정노력으로 계산하였으며, 각 방법에 의해 검색된 유사한 사례들의 평균 수정노력들은 <표 5>와 같이 요약된다.

<표 5> 두 방법에 의한 사례별 수정 노력

방 법	MME(Mean of Modification Efforts)
LMP	38,38,15,15,15,14,12,9,32,16,21,11,16,13,11,18,15,15,13,13,15
FM	45,41,17,17,15,14,12,9,35,16,22,11,16,16,11,18,19,16,16,14,19

각 방법에 의한 평균 수정노력들을 통계적으로 검정(t-value = 3.88476, p-value = 0.00046)한 결과 LMP 방법에 의한 수정노력들이 FM 방법에 의한 수정노력들보다 작다고 주장할 수 있다. 따라서 본 연구에서 제안한 사례의 수정최소화 기법에 의한 사례의 수정은 프로젝트 특징의 유사도에 의해 선정된 사례를 수정하는 방법보다 통계적으로 더 효율적이다 라고 주장할 수 있다.

Ⅶ. 결 론

새로운 프로젝트의 계획을 수립하기 위하여 과거의 사례와 지식을 사용하여 프로젝트 네트워크를 자동으로 생성하는 사례기반 접근 방법을 개발하였다.

사례들을 효과적으로 분류하고 식별하기 위하여 소프트웨어 프로젝트 네트워크의 구조에 영향을 주는 17개 요소를 도출하였으며, 이 요소는 사례의 인덱스, 사례의 검색, 그리고 사례의 수정을 위한 기준으로 적용하였다.

유사한 사례를 검색하기 위하여 과거 사례를 수정하는데 요구되는 노력을 예측할 수 있는 최소 수정노력기반 사례선정 방법을 개발하였고, 과거의 유사한 사례를 수정하기 위하여 요소별 수정지식을 사용하는 지식기반 사례수정 방법을 개발하였다.

본 연구에서 제안한 수정노력 최소화 방법에 의한 사례 수정은 프로젝트 특징들의 일반적인 거리 최소화 방법에 의해 선택된 사례를 수정하는 방법보다 통계적으로 유의하게 더 효율적임이 실험적으로 검증되었다. 따라서 이 방법을 활용하면 과거사례와 지식을 사용하여 자동으로 프로젝트 네트워크를 생성할 수 있으며, 과거의 프로젝트 경험이 체계적으로 축적되어 재 활용될 수 있다. 또한 다른 분야의 네트워크를 생성하는 문제에도 적용될 수 있을 것이다.

〈참 고 문 헌〉

[1] A. Cimitile, G. Visaggio, "Managing Software Projects by Structured Project Planning," *International Journal of Software Engineering and Knowledge Engineering*, World Scientific Publishing, Vol. 7, No. 2, 1997, pp. 169-170.

[2] Arvind Sathi, Mark S. Fox, Michael Greenberg, "Representation of Activity Knowledge for Project Management," *IEEE Transaction on pattern analysis and machine intelligence*, Vol. PAMI-7, No. 5, September 1985.

[3] Avelino J, Gonzalez, Lingli Xu, and Uma M. Gupta, "Validation Techniques for Case-Based Reasoning Systems," *IEEE Transactions on Systems. Man. and Cybernetics-part A: System and Humans*, Vol. 28. No. 4, July 1998.

[4] BARRY W. BOEHM, "Software Engineering Economics," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 1, 1984.

[5] Carmen Fernandez-Chamizo, Pedro A. Go nzales-Calero, Luis Hernandez-Yanez, and Alvaro Urech-Baque, "Case-Based Retrieval of Software Components," *Expert Systems with Applications*, Vol. 9, No. 3, 1995, pp. 397-405.

[6] Carsten Tautz and Klaus-Dieter Althoff, "Using Case-Based Reasoning for Reusing Software Knowledge," *Second ICCBR-97 Providence, RI, USA, 1997 Proceeding*, 1997, pp. 156-167.

[7] DoD., *Guidebook for MIL-STD-498: Overview and Tailoring*, DoD., 1995.

[8] Eung Sub Jun, Jae Kyu Lee., "Quasi-optimal case-selective neural network model for software effort estimation," *Expert Systems with Applications*, Vol. 21, 2001, pp. 1-14.

[9] Frank McGarry, Rose Pajerski, Gerald Page, Sharon Waligora, Victor Basili, Marvin Zelkowitz, "Software Process Improve in the NASA Software Engineering Laboratory," Technical Report CMU/SEI-94-TR-22, ESC-tr-94-022, December 1994.

[10] Finnie, G.R., Wittig, G.E., & Petkov, D.I., "Prioritizing software development using the analytic hierarchy process," *System Software*, Vol. 22, 1993, pp. 129-139.

[11] Humphrey, Watts S., *Managing the software Process, The SEI Series in Software Engineering*, Addison-Wesley Publishing Company, Inc., 1989.

[12] ISO/IEC JTC1 SC7., "*ISO/IEC 12207-Software lifecycle process*," ISO/IEC, 1995.

[13] ISO/IEC JTC1/SC7/WG7 N94., "Information Technology-Guide for ISO/IEC 12207-Software Lifecycle Processes," Feb 1998.

[14] Joseph D. Blackburn, Gary D. Scudder, Luk N. Van Wassenhove, "Improving Speed and Productivity of Software Development, A Global Survey of Software Developers," *IEEE Trans on Software Engineering*, Vol. SE-22, No. 12, 1996.

[15] Katria D. Maxwell, Luk Van Wassenhove, and Soumitra Dutta, "Software Development Productivity of European Space, Military, and Industrial Applications," *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, October 1996.

[16] Lee, K.J., Kim, H.W., Lee, J. K.&Kim, T.

H., "FASTrak-APT: Case and Constraint-Based Apartment Construction Project Planning System," *AI Magazine*, Vol. 19(1), 1998, pp13-24.

[17] Martha Branstad, Patrica B., "Software Engineering Project Standards," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 1, January 1984.

[18] Mary Lou Maher and Andres Gomez de Silva Garza., "Case-Base Reasoning in Design," *IEEE Expert*, 1997.

[19] Min Soo Suh, Won Chul Jhee, Young Kwan Ko, Albert Lee, "A case-based expert system approach for quality design," *Expert Systems with Applications*, Vol. 15, 1998, pp. 181-190.

[20] Philippe Kruchten, *The Rational Unified Process*: Addison Wesley 1999.

[21] Richard H. Thayer, Arthur B. Pyster, "Major Issues in Software Engineering Project Management," *IEEE Trans on Software Engineering*, Vol. SE-7, No. 4, 1981.

[22] R. Pietro-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software*, January 1987, pp. 6-16.

[23] Sergio Bandinelli, Alfonso Fuggetta, "Modeling and Improving an Industrial Software Process," *IEEE Transaction on software engineering*, Vol. 21, No. 5, May 1995.

[24] S.J. Noronha, V.V.S. Sarma, "Knowledge-Based Approaches for Scheduling Problems: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No 2, June 1991.

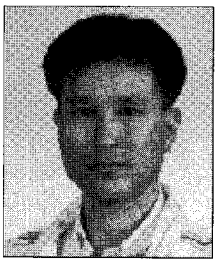
[25] Sterling Software, Inc., *Project Manager Handbook*, Part Number 2579017-0004, Fourth Edition, August 1997.

[26] Tridas Mukhopadhyay et. al., "Examining the feasibility of a Case-Based Reasoning Model for Software Effort Estimation," *MIS Quarterly*, June 1992, pp. 155-171.

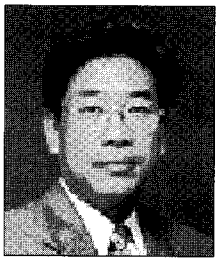
[27] Victor R. Basili, H.Dieter Rombach, "Support for Comprehensive Reuse," *Software Engineering Journal*, Vol. 6.5, Sep. 1991, pp. 303-316.

[28] William R. Duncan, *A Guide to The Project Management Body of Knowledge*, PMI Standards Committee, 1996.

◆ 저자소개 ◆



이노복 (Lee, No Bok)
중앙대학교에서 전산학 학사와 컴퓨터공학 석사학위를 하였고, KAIST 테크노경영대학원에서 박사과정중이다. 현재 국방과학연구소에서 팀장으로 근무하고 있다. KIDA와 IDIS에서 연구위원으로 근무하였으며, 정보시스템 컨설팅 및 개발, CALS/EC 분야에 연구개발 경험이 있다. 주요 관심분야는 소프트웨어 프로세스, 소프트웨어 프로젝트 관리, EJB기반 Component Based Development, 지능형 정보시스템 등이다.



이재규 (Lee, Jae Kyu)
서울대학교 산업공학과 학사, 한국과학기술원 산업공학과 석사, University of Pennsylvania의 The Wharton School에서 박사를 취득하였다. 현재 한국과학기술원 테크노경영대학원 교수 및 (사)국제전자상거래연구센터 소장으로 재직 중이며, Journal Electronic Commerce Research and Applications의 편집장, Decision Support Systems, Expert Systems with Applications, International Journal of Electronic Commerce등의 국제학술지에서 편집위원으로 활동하고 있다. 한편, 한국지능정보시스템학회 학회장을 역임하였고, The 3rd World Congress on Expert Systems(1996)와 International Conference on Electronic Commerce(1998, 2001)에서 학술대회장을 역임하였다. 주요 연구분야는 전자상거래와 지능정보시스템 등이다.

◈ 이 논문은 2002년 6월 10일 접수하여 1차 수정을 거쳐 2003년 1월 6일 게재확정되었습니다.