TRIZ의 모순 해결 이론을 이용한 창의적 요구사항 충돌 해결

(Creative Resolution for Requirement Conflict Using Conflict Resolution Theory of TRIZ)

정 지 영 † 김 진 태 †† 박 수 용 †††
(Jiyoung Jung) (Jintae Kim) (Sooyong Park)

요 약 요구사항 충돌은 시스템의 기능의 오작동이나 프로젝트 전체의 실패를 가져올 수 있다. 현재 요구사항 충돌연구는 식별에 치중되어 있고 해결에 관한 연구는 우선순위에 의하여 선택을 하는 것이 대부분이다. 요구사항 충돌을 해결하기 위해 본 논문에서는 TRIZ의 이론을 적용하여 창의적인 아이디어를 제시할 수 있도록 돕는 요구사항 충돌 해결 원리(CRRC)를 제안한다. TRIZ는 모순을 해결하여 아이디어를 내는 데 특화된 이론으로 200만 건 이상의 특허 사례를 바탕으로 만들어졌다. CRRC는 요구사항 충돌을 분류하고 유형에 적합한 TRIZ이론을 소프트웨어에 맞게 적용하였다. 대조 실험 적용 결과 CRRC를 제공하면 다양한 종류의 창의적인 요구사항 충돌 해결 방안을 제시할 수 있었다.

키워드 : 요구사항 충돌, 요구사항트레이드오프분석, 요구사항 충돌 분류, 트리즈, 요구사항 충돌 해결

Abstract The Conflicts between requirements may cause a failure of functions or even project. Currently, most of researches have focused on identifying requirements and some researchers have tried to resolve requirements conflicts but it was only based on requirement priority. This paper proposes the Creative Requirements Conflict Resolution (CRRC) to resolve requirement conflicts in a creative way using TRIZ methodology. TRIZ, which means the theory of solving inventor's problems, is made based on the analysis of over 2 million patent cases and helpful for developing a creative solution to resolve conflicts. CRRC classifies requirement conflicts into groups and then apply TRIZ theory related to each group. At the result of control experiment, CRRC provides the various kinds of creative solution for requirement conflicts.

Key words : Requirements conflict, requirements trade-off analysis, Requirements conflict classification, TRIZ, conflict resolution

1. 서 론

소프트웨어의 규모가 점차 커지고 다양해짐에 따라 요구사항 관리의 중요성은 더욱 부각되고 있다. 특히 잘못된 요구사항은 프로젝트의 실패의 원인이 된다[1]. 요구사항은 빠르게 변화하는 시장의 상황과 하드웨어의 발전 등의 이유로 소프트웨어 개발 기간 내내 변경이 일어난다. 특히 다양한 이해당사자들이 관련되어 있는 개발 환경의 경우 요구사항에 대해 서로 다른 의견과 제약사항들이 존재한다.

이러한 상황에서 요구사항을 기술하거나 변경할 때 요구사항 충돌이 발생할 수 있다. 요구사항 충돌이란 "둘 이상의 요구사항이 같이 만족되어야 할 때 예상치 못한 영향을 미치거나 부정적인 영향을 미쳐 하나 이상의 요구사항이 만족되지 못하는 현상"이다. 요구사항을 기술할 때는 다양한 이해당사자들로부터 다양한 요구사항이 나타나므로 그로 인해 다양한 요구사항들 간의 충

† 학생회원 : 서강대학교 컴퓨터공학과
edie.jung@gmail.com
†† 정 회 원 : 소프트웨어공학엑스퍼트그룹(주) 대표이사
canon.kim@gmail.com
††† 정 회 원 : 서강대학교 컴퓨터공학과 교수
sypark@sogang.ac.kr
논문접수 : 2010년 1월 14일
심사완료 : 2010년 3월 9일

정보과학회논문지: 소프트웨어 및 응용 제37권 제5호(2010.5)

돌이 발생할 수 있다. 또한 요구사항 변경 시 변경 프로세스의 빈약함으로 요구사항 변경영향 분석과 같은 과정 없이 무분별하게 변경되게 되어 요구사항 충돌이 발생할 수 있다.

이러한 요구사항 충돌을 해결하는 비용은 시스템 개발 후반부로 갈수록 급속히 증가한다[2]. 이는 충돌 문제를 해결할 때의 복잡도가 요구사항 단계보다 솔루션 단계에서 해결할 때 50배 이상 증가하기 때문이다[3]. 그러므로 요구사항 충돌을 요구사항 단계에서 해결하는 것은 비용을 감소하고 프로젝트를 성공하기 위해 필요하다.

본 논문의 목적은 요구사항 충돌이 식별된 후 요구사항 단계에서 양자택일이 아닌 다양한 해결방법을 제시하는 것이다. 이를 위해 TRIZ[4,5]를 활용하였다. TRIZ는 발명의 원리를 집대성한 지식 체계로써 새로운 해결방법을 찾았던 경험들을 분석하고, 일반화하여 다른 문제에서도 적용 가능하다. TRIZ를 활용하여 요구사항 충돌을 해결하는 본 연구의 특징은 다음과 같다.

요구사항 충돌의 유형을 정리하여 그것에 적합한 TRIZ의 모순 해결 이론을 매핑하였다. 그것을 잘 설명하기 위한 템플릿에 맞추어 TRIZ의 이론이 소프트웨어에 적용될 수 있도록 유형별로 패턴화하였다.

2. 충돌 해결 원리 관련연구

Isabel Sofia Brito외[6]는 AHP(Analytic Hierarchy Process)를 적용하여 다중기준에 의한 우선순위를 매겨 요구사항을 선택하도록 하였고, 김민성 외[7]은 요구사항의 충돌의 유형을 체계화하고, 목표(Goal)를 기반으로 요구사항에 우선순위를 매겨 선택할 수 있도록 하였다. 이 연구들은 요구사항 충돌의 해결 방법이 목표의 우선순위와 이해당사자들이 생각하는 중요도에 따라 우선순위를 매겨 분석가가 충돌이 일어난 요구사항 중 택일하도록 도왔다.

[8]에서는 요구사항 불일치의 타입을 프레임워크에 따라 정의하고 프레임워크를 구성하는 개체에 따라 그들 간 관계를 결정한다. 이렇게 기술된 목표와 요구사항 명세로부터 n-ary충돌을 식별하기 위한 정형화된 기법을 제공하여 충돌을 해결한다. [9]에서는 목표단계에서 충돌을 조절하기 위해 추상화 레벨에서 목표를 모델링하고 명세하여 목표에 대한 증명 및 검증 충돌관리를 가능하게 하는 KAOS라는 방법론을 사용한다. 또William N. Robinson [10]은 요구사항 온톨로지의 지식을 사용하여 요구사항 컴포넌트를 추가하거나 제거하는 방식으로 해결방법을 자동으로 생성하였다.

Barry Boehm[1][11]은 "Win conditions"을 설정한 뒤 "win condition"을 만족하기 위한 아키텍처 전략을 식별하고, 각 아키텍처 전략들이 미치는 영향을 분석하였다. 그것을 바탕으로 "win condition"에 부정적 영향을 끼칠 수 있는 요구사항 충돌을 식별하고 지식을 기반으로 한 해결방안들을 제시하여 충돌을 해결하도록 도왔다. 이 연구는 충돌을 해결하기 위해 아키텍처 수준의 지식이 필요하며 해결방안의 다양성은 아키텍처 전략 지식의 양에 한정되어 있다.

위 논문들은 해결 방법이 선택이라는 한가지 방법으로 제한적이거나 경험에 의해 문제를 해결할 경우 경험의 크기가 도메인에 한정되어 있다. 또 정형언어를 사용하고 도메인을 온톨로지화하는데 많은 노드가 발생된다는 문제가 발생한다. 따라서 본 논문에서는 요구사항 충돌 유형별로 적합한 TRIZ의 이론을 적용하여 충돌해결의 경험을 바탕으로 다양한 해결방안을 요구사항 단계에서 제시할 수 있도록 하였다.

3. CRRC 생성 과정

본 논문에서는 요구사항 충돌이 식별되었다는 가정하에 CRRC를 이용하여 해결 방안을 제시한다. CRRC를 생성하는 과정은 4 단계로 이루어진다. 첫째, TRIZ를 학습하고 둘째, 충돌유형을 분류한다. 셋째, CRRC를 설명하기 위한 템플릿을 생성하고 넷째, 각 충돌 유형에 적합한 TRIZ의 해결원리를 매핑하고 요구사항 충돌에 적합하도록 소프트웨어화 하여 패턴을 생성한다.

3.1 TRIZ의 학습

TRIZ(Teoriya Reshnya Izobretalskikh Zadatch) [4,5]은 겐리흐 알트슐러가 200만 건 이상의 특허를 분석하여 창조적 문제 해결에 사용되는 공통 원리를 추출, 통계적으로 분석하여 문제 해결의 원리로 정리해 놓은 것이다. TRIZ의 이론은 방대한 것으로 본 논문에서 사용되는 TRIZ 이론은 모순 해결을 위해 특화된 분리의 원리와 발명원리를 사용한다. 분리의 원리란 물리적 모순이 있을 때 시간, 공간, 조건 등을 분할하여 해결점을 찾는 원리이다. 예를 들면 비행기가 착륙과 비행시 바퀴가 있어야 하지만 동시에 없어야 할 때 시간에 따라 바퀴를 몸체 안으로 넣음으로써 문제를 해결하였다. 발명원리란 물체의 속성을 나타내는 표준인자들 간에 충돌이 일어났을 때, 즉 하나의 속성을 증진시키려 하면 다른 속성이 감퇴할 경우의 문제를 해결하기 위한 것으로 진공 청소기의 압력을 높이려 할 때 소리가 커지는 유해한 부작용이 발생할 경우 소리가 나는 부분을 소리가 잘 들리지 않는 곳으로 분할하여 집에서 소리가 안 들리게 하는 것이 그 예이다. TRIZ는 소프트웨어에서 사용되도록 여러 가지 시도가 있었으나[12,13] 요구사항의 충돌을 해결하기 위해 사용하는 것은 본 논문이 처음이다.

3.2 충돌 유형 분류

요구사항 충돌의 유형은 크게 기능적 요구사항 충돌과 비기능적 요구사항 충돌로 나눌 수 있다. 기능적 요구사항의 충돌은 크게 액티비티 충돌과 자원충돌[7]로 나누어지며, 비기능에 포함되는 요구사항 충돌은 신뢰성, 유지보수성, 가용성, 사용용이성, 이식성 등의 품질속성간에 부정적 영향을 미치는 충돌[14]현상들로 나누어진다. 액티비티 충돌은 행위가 문제가 되는 충돌로 다른 객체가 동시에 같은 행위를 취하거나 반대되는 행동을 취하는 경우 액티비티 충돌이 발생한다. 예를 들면 주택의 안전성, 보안성 등을 위해 제작된 주택 통합시스템에서 침수방지기능과 화재방지기능을 만족시키기 위해 "집안의 수위가 10cm를 넘으면 수도를 차단한다."라는 요구사항과 "화재가 발생하면 스프링쿨러를 작동한다"라는 요구사항이 있을 때, 하나의 요구사항은 수도를 차단하려 하고 다른 하나의 요구사항은 수도를 계속 흐르게 해야 한다. 이러한 경우를 액티비티 충돌이라 한다. 비기능 요구사항 충돌의 경우 두 개의 품질속성 사이에 미치는 영향이 '-'일 때 즉 하나의 속성이 높아질 때 다른 하나의 속성이 낮아지는 경우를 비기능 요구사항 충돌이라 한다. 예를 들면 무결성을 높이는 경우 정보의 암호화 작업이나 여러 단계를 거쳐 시스템의 처리가 이루어지게 되므로 처리 속도가 늦어지게 된다. 이때 고객이 무결성과 효율성을 동시에 중요하게 생각한다면 무결성과 효율성 사이에 충돌이 발생한 것이다.

3.3 CRRC 템플릿을 항목 설계

여러 가지 패턴 책[15,16]을 참조하여 요구사항 충돌 해결원리에 적합한 CRRC의 템플릿을 만들었다.

충돌 해결 원리 이름	이름		
충돌 유형	기능, 비기능 여부: 세부 유형 (A증가시 B감소)		
충돌 요약	충돌의 의미		
충돌의 대표적인 예	충돌이 일어나는 예를 통해 유형을 설명		
해당 원리 Mapping Table (비기능 요구사항 일 경우에만 존재)		B에 해당하는 표준인자	
	A에 해당하는 표준인자	해결 원리 번호	
해결 원리	•해결원리들		
해결 방안	해결원리에 대한 소프트웨어 적용 방안		
해결 예제	해결원리 적용 예		

그림 1 CRRC 템플릿 항목

3.4 각 충돌 유형에 대한 CRRC 생성

요구사항 충돌 유형 별로 CRRC를 생성한다. 각 유형에 적합한 해결원리는 유형의 특성에 따라 결정한다. 비기능 유형의 경우에는 품질속성과 유사한 TRIZ의 기술 파라메터를 매핑한다. 예를 들면 그림 3에서의 효율성은 TRIZ의 기술 파라메터 속도, 생산성, 시간손실, 움직이지 않는 물체에 사용된 에너지에 해당한다. 기술 파라메터 간의 충돌에 대해 해결원리를 TRIZ에서 제시하면

충돌 해결 원리 이름	행위의 분리
충돌 유형	기능 : 요구사항간 행위 충돌
충돌 요약	하나의 객체가 동시에 다른 기능을 수행하도록 요구될 때 발생
충돌의 대표적인 예	자동차 소프트웨어에서 사고가 발생했을 때 R1: GPS 위치를 송신하는 기능 R2: 폭발 방지를 위한 전원차단 기능
해결 원리	•시간에 의한 분리 •조건에 의한 분리 •공간에 의한 분리
해결 방안	• 시간에 의한 분리 충돌이 일어나는 원인이 되는 행동을 시간에 따라 분리한다. 즉 기능이 실행되는 시간을 나눈다. • 조건에 의한 분리 이미 기능이 실행 중일 때 특정 조건을 만족할 때만 다른 기능을 실행한다. 특정 조건은 사용자가 선택하는 경우가 해당된다. • 공간에 의한 분리 기능을 공간으로 나눈다. 기능이 실행되는 부분을 나누어 충돌을 없앤다.
해결 예제	1. 시간에 의한 분리 GPS위치를 송신한 후 바로 전원을 차단한다.(우선순위에 의해 시간 분리) 2. 조건에 의한 분리 사고 발생시 GPS를 송신하는 장치에만 전원을 공급하는 기능을 추가한다. 3. 공간에 의한 분리 GPS송신장치에 전원 공급장치를 따로 설치하여 폭발의 위험을 없앤다.

그림 2 행위의 분리

충돌 해결 원리 이름	효율성을 유지하며 무결성 수준 높이기				
충돌 유형	비기능 : Integrity ↑ →Efficiency ↓				
충돌 요약	무결성을 높이는 기능을 추가하면 효율성이 낮아지는 경우				
충돌의 대표적인 예	정보의 손실과 해킹의 위험을 막기 위해 복잡한 사용자 인증 절차를 추가하면 처리 속도가 낮아지게 된다.				
해당 원리 Mapping Table		속도	움직이지 않는 물체에 사용된 에너지	시간 손실	생산성
	정보의 손실	26, 32		24, 26, 28,32	13, 23, 15
	물체에 작용하는 유해요소	21,22,35,28	10,2,22,37	35,18,34	22,35,13,24
해결 원리	• 2. Extraction(필요한 것만 뽑아내라) • 10. Prior action 미리 조치하라 • 13. Do it in reverse 반대로 하라 • 15. Dynamicity 부분적으로 자유롭게 하라, • 18. Mechanical Vibration 진동을 이용한다 • 21. Rushing through 유해하다면 빨리 진행하라 •22. Convert harm into benefit 안 좋은 것을 좋은 것으로 바꿔라. • 23. Feedback 피드백을 이용하라 • 24. Mediator 직접 하지 말고 중간 매개체를 이용하라 • 26. Copying 플편하고 복잡한 것 대신 간단한 것으로 복사한다. • 28. Replacement of Mechanical System 기계적 시스템은 광학, 음향시스템 등으로 바꾼다. • 32. Changing the color 색깔 변화 등 광학적 성질을 변화시킨다 • 34. Rejecting and regenerating parts 다 쓴 것은 버리거나 복구한다 • 35. Transformation properties 물질의 속성을 변화한다. 실험계획법을 사용하여 최적화 한다. • 37. Thermal expansion 열팽창				
해결 방안	26. Copying 플편하고 복잡한 것 대신 간단한 것으로 복사한다. 복제는 정보가 손실되기 쉬운 서비스 대신에 간단하고 값한 복제 서비스를 이용하도록 하는 것이다. DB 시스템에 View를 도입하여 사용자에게 복잡한 보안 절차를 줄이고 정보의 손실이 없을 만한 내용만을 보여주는 것이다. 32. Changing the color 색깔 변화 등 광학적 성질을 변화시킨다 광학 특성 변경은 대상 기능의 투명도를 바꾸는 것이다. 예를 들면 컴퓨터 시스템의 방화벽이 될 수 있다. 이 방화벽은 정당한 사용자에게는 투명하지만, 동시에 중요한 정보를 훔치려는 사람에게는 절투하지 못하게 할 수 있다.				
해결 예제	26. 복제의 경우, 은행의 ATM 에서 다양한 서비스를 할 때 사용자에게 모든 정보를 보여주지 않고 사용자에게 필요한 정보만 제공한다. 복잡한 인증과정을 줄이고 무결성을 높이면서도 실행 속도를 늦추지 않을 수 있다. 32. File Sharing System에서 각 사용자에게 접근 레벨을 주어 일일이 폴더에 접근하려 할 때마다 사용자 인증을 하지 않고, 아예 접근 가능한 폴더만을 보여주면 효율성이 떨어지지 않을 수 있다.				

그림 3 효율성을 유지하며 무결성 수준 높이기

그것을 바탕으로 요구사항 충돌에 대한 해결 방안을 제시하였다. 그림 2와 3은 각각 기능 유형과 비기능 유형의 충돌의 CRRC의 예이다.

4. 검 증

본 논문에서 제안한 CRRC를 제공하면 다양한 해결방법을 제시할 수 있는가를 검증하기 위해 다음과 같은 대조 실험을 하였다. 총 77명(산업체 경력자: 32명, 대학생: 45명)의 실험 그룹을 경력과는 무관하게 두 그룹으로 나누어 집단조사법을 통한 결과를 얻었다. 그룹 1은

경력자 14명, 대학생 23명으로 구성되어 있고, 그룹 2는 경력자 18명, 대학생 22명으로 구성되었다. 두 그룹에 동일한 요구사항의 충돌 예제를 주고 해결방안을 제시하도록 하였다. 요구사항 충돌 예제는 HIS(Home Integration System)에서 발생한 요구사항(화재 방지, 홍수 방지, 침입 탐지, 온도 조절) 충돌 사례이다. 제시된 요구사항 충돌 사례는 아래와 같다.

RA. 화재 방지를 위한 요구사항(HIS는 전기 누전이 발생시 전원 차단 기능)

RB. 침입자 방지를 위한 요구사항(HIS는 모션 센서를 이용한 침입자 감시)

고객은 두 요구사항을 모두 구현하길 원한다. 그러나 전기가 없을 경우 모션센서의 동작이 불가능하므로 양립할 수 없다.

실험 결과 표 1과 같은 결과가 나타났다.

표 1 실험 결과 데이터

	CRRC	요구사항 A선택	요구사항 B선택	A, B둘 다 선택	총 인원	아이디어의 수
그룹 1	무	24	6	10	37	8
그룹 2	유	7	2	28	40	15

본 실험은 CRRC가 사용되었을 때 다양한 아이디어를 제시할 수 있는가를 검증하는 것이 목적이다. 결과의 아이디어의 수는 실제로 적용이 가능하다고 판단되는 경우이다. CRRC를 제공하면 다양한 아이디어를 제시하는지를 증명하기 위해 모비율의 차에 의한 검정을 하였다. A, B 둘 다 선택한 비율이 관심비율일 때 모비율은 '둘 다 선택한 사람/전체 사람수'이다. CRRC를 준 그룹의 모비율이 CRRC를 안준 그룹의 모비율 보다 큰지에 대해 관심이 있는 것이므로 가설을 다음과 같이 설정한다[17]. 여기서 p1은 문제만 제시한 그룹의 모비율이고 p2는 CRRC를 제공한 그룹의 모비율이다.

위무가설 (H_0): p1-p2=0

대립가설 (H_1): p1-p2<0

유의 수준 $\alpha = 0.05$

임계치 및 기각영역

임계치: $z_{(\alpha/2)}=1.96$

기각영역: Z>1.96 또는 Z<-1.96

표본 통계량의 계산 및 임계치와의 비교는 아래의 수식과 같다.

$$z=\frac{\hat{p}_1-\hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}}=\frac{0.25-0.75}{\sqrt{\left(\frac{38}{77}\right)\left(\frac{38}{77}\right)\left(\frac{1}{37}+\frac{1}{40}\right)}}$$

$$= -4, < 1.96$$

계산된 Z값은 기각영역에 포함되므로 위무가설을 기각한다. 따라서 유의수준 5% 하에서 CRRC를 제공하면 A와 B둘 다 선택한 비율이 더 크다고 할 수 있다. 검증 결과를 통하여 CRRC를 사용하면 다양한 아이디어를 제시할 수 있음을 알 수 있다.

5. 결 론

요구사항은 변경이 잦고 다양한 이해당사자가 참여하는 특성을 가지고 있다. 따라서 요구사항 충돌이 빈번히 발생할 수 밖에 없다. 이런 요구사항 충돌은 시스템의 기능 장애를 발생시키고 더 나아가서는 시스템 전체의 실패를 불러올 수 있다. 요구사항 충돌은 대부분 우선순위에 따른 택일로 해결되어 고객의 만족도를 떨어뜨린다. 본 논문에서는 요구사항 충돌이 식별되었다는 가정하에 충돌을 해결하는 방안을 다양하게 제시할 수 있는 해결 원리들을 제시하였다.

CRRC는 하드웨어나 일반적인 문제에 사용되던 TRIZ를 요구사항 충돌에 사용하였다. CRRC를 사용하면 요구사항 충돌 발생 시 단순히 요구사항 하나를 선택하는 것이 아닌 둘 다 만족하는 해결 방법을 제시할 수 있다. 요구사항 충돌에 대해 유형을 나누어 TRIZ의 모순해결 이론을 소프트웨어에 적합하도록 확장하여 CRRC를 만들었다.

요구사항 충돌 예제에 대해 실험을 해본 결과 CRRC를 적용하지 않은 집단에 비해 다양한 아이디어를 냈으며 이를 통해 TRIZ의 문제해결원리를 이용하여 요구사항 충돌 문제를 해결할 수 있음을 알 수 있다.

참 고 문 헌

[1] Barry Boehm and Hoh In, "Identifying Quality Requirement Conflicts," *IEEE software*, March, pp.25-36, 1996.

[2] Alan M. Davis, "Software Requirements Analysis & Specification," Prentice Hall, pp.23-24, 1990.

[3] Glass, Robert L., "Requirements Tracing," *In Modern Programming Practices, Englewood Cliffs*, NJ: Prentice-Hall, pp.59-62, 1982.

[4] Genrich Saulovich Altshuller, "Innovation Algorithm," 현실과 미래, 2002.

[5] 김효준, 생각의 창의성: Theory of Inventive Problem Solving TRIZ, 도서출판 지혜, 2004.

[6] Isabel Sofia Brito, Filipe V., Ana M. and Rita A. Ribeiro, "Handling Conflicts in Aspectual Requirements Composition," *Transactions on AOSD III, LNCS 4620*, pp.144-166, 2007.

[7] Minseong Kim, Sooyong Park, Vijayan Sugumaran, Hwasil Yan, "Managing requirements conflict in software product lines: A goal and scenario based approach," *Data & Knowledge Engineering 61*,

pp.417-432, 2008.
[8] Axel van Lamsweerde, "Managiing Conflicts in Goal-Driven Requirements Engineering," *IEEE Transactions on Software Engineeriing*, vol.23, no.11, November 1998.
[9] A.van Lamsweerde, and L.Wollemet, "Inferring Declarative Requirements Specifications from Operational Scenarios," *IEEE Trans. Software Eng.*, vol.24, no.12, Dec.1998. to appear.
[10] W.N. Robinson, V.Volkove, Requirements Conflict Restructuring, Georgia State University, Atlanta, GA, 1999.
[11] Egyed,A., Barry Boehm, USC, "Analysis of Software Requirements Negotiation Behavior Patterns," USC-CSE-96-504, 1996.
[12] Kevin C. Rea: "TRIZ and Software - 40 Principle Analogies, Parts 1 and 2," TRIZ Journal, Sept. and Nov. 2001.
[13] Osamu Shigo: "Program Engineering - Implementation, Design, Analysis, and Testing," Science-Sha, Oct. 2002.
[14] Karl E. Wiegers, Software Requirements: Practical T Techniques for Gathering and Managing Requirements Throughout the P, Microsoft Press, 2003.
[15] Stephen Withall, Software Requirement Patterns, Microsoft Press, 2007.
[16] Buschmann et el, Pattern-Oriented Software Architecture, Wiley, 1996.
[17] 황인창, 이대용, 이청호, 알기쉬운 통계학, 비앤엠북스, 1998.

정 지 영
2008년 서강대학교 컴퓨터공학과 학사
2010년 서강대학교 컴퓨터공학과 석사
관심분야는 요구공학, 소프트웨어 아키텍처

김 진 태
1998년 서강대학교 전자계산학 학사. 2000년 서강대학교 컴퓨터공학 석사. 2001년 데이콤시스템 테크놀로지 소프트웨어 엔지니어. 2005년 서강대학교 컴퓨터공학 박사. 2009년 삼성전자 DMC 연구소 책임연구원. 현재 소프트웨어공학엑스퍼트그룹㈜ 대표이사. 관심분야는 Empirical software engineering, 역공학, 아키텍처, 소프트웨어 프로덕트 라인, 요구공학

박 수 용
1986년 서강대학교 전산학과, 공학학사. 1988년 Florida State University, Computer Science, M.S. 1995년 George Mason University, Information Technology, Ph. D. 2003년 3월~2004년 2월 한국 정보통신대학교 컴퓨터학과, 초빙교수. 2003년 8월~2006년 12월 카네기 멜론대학 컴퓨터학과, MSE 객원교수. 정보과학회 소사이어티 회장. 서강대학교 컴퓨터학과, 교수