디지털산업정보학회 논문지 제5권 제2호-2009년 6월

위험분석 모델을 적용한 정량적인 소프트웨어 위험관리 방법론에 관한 연구

엄 정 호* · 이 동 영** · 정 태 명***

A Study on Quantitative Software Risk Management Methodology applied Risk Analysis Model

Eom, Jung Ho · Lee, Dong Young · Chung, Tai M.

〈Abstract〉

In the paper, we proposed the systematical and quantitative software risk management methodology based on risk analysis model. A software risk management consists of the basic risk management method(BRIMM) and the detailed risk management method(DRIMM). BRIMM is applied to unimportant phases or the phase which also the risk factor does not heavily influence to project. DRIMM is used from the phase which influences highly in project success or the phase where the risk factor is many. Fulfilling risk management combined two methods, we can reduce project's budget, term and resource's usage, and prevent risk with the optimum measures obtained by the exact risk analysis.

Key Words : Software Risk management, Risk Analysis

Ⅰ. 서론

소프트웨어 프로젝트에서 위험이란 "소프트웨어 개발 프로젝트 실패를 초래할 수 있는 개발 작업이나 환경에 관련된 속성들"로 정의되고 있다. 이런 위험요소들은 소프트웨어 개발공정 단계에서 언제든지 나타날 수 있다. 위험요소를 조기에 식별하고 분석하여 적절한 대응책을 마련함으로써, 개발 기간, 비용 및 자원의 소모를 절약하고 소프트웨어의 품질을 보장하며, 프로젝트 실패라는 최악의 경우도 방지하는 과정을 위험관리라 한다[1-3]. 예전에는 위험관리를 소프트웨어 개발과정에서 배제하거나 소프트웨어 개발공정 중에 우선순위가 낮은 공정으로 취급하였다. 그러나 현재는 소프트웨어 프로젝트의 성공여부를 결정짓는 중요한 과정으로 다뤄지고 있으며, 그에 따른 필요한 인력, 예산, 기간, 자원, 위험분석 자동화 도구 개발 등에 많은 역량을 집중시키고 있다.

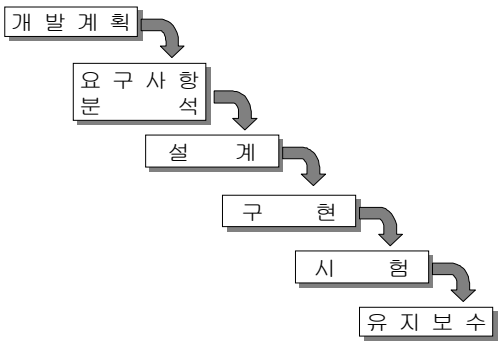
본 논문에서는 'ISO/IEC TR 13335-3'[4]를 적용한 정량적인 방법의 소프트웨어 위험관리를 수행한다. 개발 공정 중 업무 중요도가 낮은 단계에서는 기본 위험관리 방법이 적용하고 업무 중요도가 높고 프로젝트에 큰 영향을 미칠 수 있는 공정 단계에서는 상세 위험관리 방법을 적용한다.

* 대한민국 공군 장교
** 명지전문대학 컴퓨터공학과 교수
*** 성균관대학교 컴퓨터공학과 교수

본 논문의 구성은 2장에서 일반적인 소프트웨어 개발공정 절차에 대해서 간략히 언급하고, 3장에서는 정량적 소프트웨어 위험관리 방법을 제시한다. 4장에서 간단하게 적용의 예를 들고, 마지막으로 5장에서는 결론을 맺는다.

II. 소프트웨어 개발 공정

소프트웨어 시스템 개발의 일반적 공정 과정은 개발계획, 요구사항 분석, 시스템 설계, 시스템 구현, 시스템 시험 및 유지보수로 구성되고 각 단계별 위험관리가 수행된다. 개발 공정은 (그림 1)과 같다[5].



<그림 1> 소프트웨어 개발 공정 과정

① 개발계획 : 일반적으로 프로젝트의 목표와 수행 방법, 추진 일정 및 전략, 조직, 비용, 요구되는 자원 등을 기술한다.

② 요구사항 분석 : 시스템 개발 요구사항을 규명하는 과정으로 요구사항 규명, 타당성 조사, 비용과 일정에 대한 제약 설정 등 시스템이 어떤 기능을 수행하는지에 초점을 맞추어 개발 시스템의 목표를 기술한다.

③ 설계 : 요구사항 분석과정에서 수집된 요구사항을 설계도면에 옮기는 과정으로, 하부 시스템들로 이루어지는 시스템 구조를 결정하고 하부 시스템들을 하드웨어 및 소프트웨어 등의 구성요소들에게 할당한다.

④ 구현 : 설계의 결과를 사용자가 이용할 수 있는 형태로 변환시키는 과정으로 설계 명세서를 시스템의 실제 형태로 변환시키는 것이다. 이러한 과정을 통해 시스템의 기능이 수행 가능한 형태를 갖게 된다.

⑤ 시험 : 품질보증 활동의 중요한 일부분으로 사용자 요구사항, 설계, 구현의 전 과정에 대한 최종 점검을 포함하고 있으며, 제품의 오류를 발견하고 수정하는 과정이다.

⑥ 유지보수 : 소프트웨어 개발 공정 5단계까지 걸쳐서 개발된 시스템은 고객에게 인도되어 사용되는데, 사용 중 발생하는 여러 변경 사항에 대해 적응하는 활동이며 변화에 대비하는 과정이다.

일반적인 소프트웨어 개발 공정에 따른 위험관리는 각 단계별 개발자나 관리자에게 설문지 기법이나 위험 항목에 따른 점검방법을 사용하고 있다. 예를 들어 구현 단계에서는 요구사항에 맞는 프로그램 언어를 사용하였는지, 입력 데이터나 프로그램 내에 자료 구성은 적합한지 등 설문이나 위험 항목 점검표에 의해 수행된다. 이것은 위험을 정성적으로 평가함으로써 구체적인 위험의 수준을 제시하지 못하여 정확한 대책을 수립하거나 대응책을 적용하기 힘든 단점을 가지고 있다.

III. 정량적 위험관리 방법

3.1 위험관리 방법

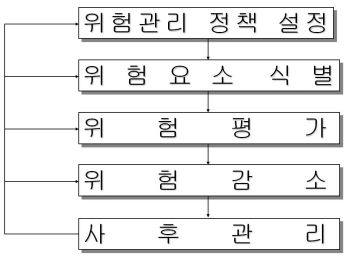
위험들은 개발 공정 중 어느 단계에서든지 나타날 수 있으며, 개발 공정 후반단계에서 발생할수록 통제하기가

점점 더 어려워진다. 예를 들면, 초기단계에 프로그래머가 코드에서 버그를 발견했을 경우에는 미소한 코드의 양만 수정하면 되지만, 후반단계에서 그 수정양이 방대하여 소프트웨어 개발 프로젝트의 전면 수정까지도 초래할 수 있다. 그래서 소프트웨어 개발과정 중에 발생되는 위험을 식별하고 평가하여 감소시키는 위험관리는 소프트웨어 개발 과정에서 매우 중요하다[6, 7].

본 논문에서는 소프트웨어 위험관리 방법을 위험분석 모델을 적용한 기본 위험관리 방법과 상세 위험관리 방법을 제시한다. 어떤 방법을 적용할 것인지는 프로젝트의 규모, 개발 목적 및 환경을 고려하여 결정한다. 프로젝트의 규모가 크면 클수록 위험요소들은 증가하고 그 영향도 크다.

3.2 기본 위험관리 방법(BRIMM)

BRIMM(Basic Risk Management Method)은 소프트웨어 개발 공정 중 중요도가 낮은 단계에서 프로젝트에 큰 영향을 주지 않는 위험요소에 대해 최소한의 자원, 비용 및 기한을 투자하여 위험관리 요구사항에 맞는 위험관리 대책을 수립한다. BRIMM은 어떤 위험요소에 대해서 상세하게 위험분석을 실시하지 않고 대책을 마련하기 때문에 차후 단계에서 위험에 노출될 수도 있다. BRIMM의 구성과 절차는 (그림 2)와 같다[4, 8-10].



<그림 2> BRIMM 절차

① 위험관리 정책 설정 : 프로젝트 전체적인 관점에서 소프트웨어 개발 목적과 요구사항, 제약조건 등을 반영한 위험관리 전략, 지침, 규약 등이 기술되어야 한다.

② 위험요소 식별 : 프로젝트에 악영향을 미칠 수 있는 잠재적인 위험요소를 파악하고 구체화하며 분류하는 과정이다.

③ 위험 평가 : 소프트웨어 개발 과정 및 소프트웨어 시스템 자체의 위험을 파악하고 발생 가능성 등을 분석하여 위험을 산출하는 과정으로 그 결과에 따라 비용효과적인 대응방법을 제시하게 된다.

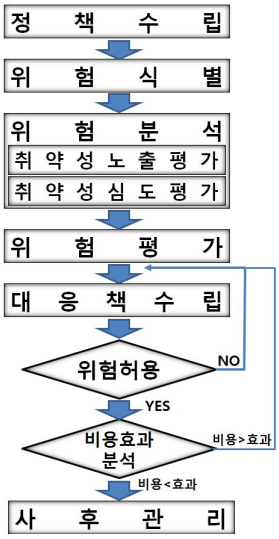
④ 위험 감소 : 위험평가 결과를 토대로 위험을 허용 위험범위 수준까지 줄일 수 있도록 회피, 대체, 축소 등의 대응방법을 설정하여 위험을 감소시키는 과정이다.

⑤ 사후 관리 : 위험관리 정책 수립에서 위험감소에 이르기까지 수행된 위험관리 단계가 적절하게 적용되었는지, 위험분석을 통한 대응방법이 허용 위험범위 수준까지 충분히 감소시켰는지 점검하는 과정이다.

3.3 상세 위험관리 방법(DRIMM)

DRIMM(Detailed Risk Management Method)는 "BRIMM의 5단계 절차"와 "ISO/IEC TR 13335-3 IT 보안관리를 위한 기술[4, 8, 10]" 의 보안관리 절차를 근거로 하여 7단계의 상세 위험관리 절차를 수립하였다. DRIMM은 일반적인 소프트웨어 개발 공정에서 단계마다 진행했던 기초적인 정성적 위험평가 방법과 달리 소프트웨어 요구사항에 따른 위험분석과 이러한 요구를 만족시키는 계획 수립, 계획의 이행조건, 위험관리 방법, 위험감소 방법 등 구체적인 절차를 반영하여 체계적인 위험관리를 가능하게 한다. 또한, 설문지 기법이나 체크 리스트를

사용하여 정성적인 결과로 위험요소마다 위험수준을 정확히 판단하지 못하고 그에 맞는 대응책도 불투명했던 기존의 방식과 다르다. 'Matrix Scaling' 기법과 위험수준 평가 방식을 활용하여 정량적인 평가가 가능하게 함으로써 위험요소를 정확히 식별하고 구체적인 위험수준을 알 수 있도록 한다. 아울러 투자 수익율을 이용하여 비용효과분석을 수행함으로써 대책별 투자 수익을 비교하여 위험요소에 대해 가장 적합한 대응책을 적용할 수 있도록 한다. 세부 위험관리 방법의 절차는 (그림 3)과 같다.



<그림 2> 세부 위험관리 절차

① 정책 수립 : 프로젝트의 목표와 수행 방법, 추진 전략, 검증 및 확인 계획, 형상관리 계획 등을 파악한 후 위험관리 목표, 정책, 전략을 수립한다. 그리고 위험수준의 적정성과 그에 맞는 적절한 위험관리 수준을 설정한다.

② 위험요소 식별 : 위험요소들을 파악하고 구체화하며 분류하는 과정이다. 위험요소 식별 방법은 주로 체크리스트, 경험적 통계, 전문가들의 회의 결과 등이 많이 사용되고 있다. 본 논문에서는 Janne Ropponen 논문에서 사용한 "소프트웨어 개발의 6개 위험요소 [11]"를 적용한다. 위험요소의 분류는 위험정도에 따른 위험요소의 우선순위 결정, 특성 파악, 대책 수립 등에 사용된다.

- 스케줄링 및 기한 준수에 대한 위험
- 시스템 기능에 대한 위험
- 계약에 따른 위험
- 요구사항에 관한 위험
- 자원 사용 및 성능에 관한 위험
- 인적요인에 의한 위험

③ 위험 분석 : 식별된 위험요소들에 대한 위험수준(RL : Risk Level)을 산출한다. 위험수준은 위험 노출정도(RE : Risk Exposure)와 발생주기(RF : Risk Frequency)의 합한 후 평균을 구한다.

- $RL = (RE + RF)/2$

④ 위험 평가 : 각 단계별 위험평가(Risk Assessment)는 해당 단계의 위험수준에 위험요소의 영향력(SRI : Stage's Risk Impact)을 곱해서 최종적으로 위험을 평가한다. SRI는 소프트웨어 개발 공정의 중요도에 의해 결정된다. 즉, 단계가 종료되기까지의 비용, 기간, 노력 등을 고려하여 높음, 보통, 낮음을 결정하고 위험평가를 산출하는데 가중치를 적용한다.

- $RA = RL \times SRI$

<표 1> SRI 값

가중치 / 개발공정	높음	보통	낮음	가중치
개발계획		●		1.5
요구사항 분석	●			2.5
설계	●			2.5
구현			●	1
시험		●		1.5
유지보수			●	1

⑤ 대응책 수립 : 위험을 감소시킬 수 있는 최적의 대응책을 수립하는 단계로 위험평가 결과에 따라 위험순위가 가장 높은 위험요소부터 대응책을 수립해야 한다. 대응책을 수립할 때 고려해야 할 점은 대응책이 가능한 위험요소들을 허용 위험수준까지 감소시킬 수 있어야 하며, 기존의 소프트웨어 개발 과정에서 적용되고 있는 대응책을 파악해서 중복되지 않게 해야 한다. 또한, 새로운 대응책으로 인해서 기존의 대응책 효과를 저하시키지 말아야 한다.

⑥ 비용효과 분석 : 수립된 대응책들의 수행여부를 비용적인 측면에서 분석하는 과정으로 대응책 설치비용과 감소되는 위험수준을 정량적으로 비교한 후 발생되는 이득을 고려하여 대응책을 설정한다.

⑦ 사후 관리 : 위험관리 정책 수립에서 대응책 수행에 이르기까지 위험관리 단계가 적절하게 적용되었는지, 위험분석을 통한 대응책이 제대로 수립되었는지, 대응책이 예산내에서 수행 가능한지, 또한 허용 위험범위 수준까지 충분히 감소시켰는지 검토하고 재분석이 필요한지도 결정하는 단계이다.

Ⅳ. 제안된 위험관리 방법론의 적용

소프트웨어 개발단계 중 요구사항 분석 단계를 제안한 위험관리 방법론에 의해 위험관리를 수행하는 것을 예를 든다.

우선 요구사항 분석 단계에서 '소프트웨어 개발의 6개 위험요소'중에 나타날 수 있는 위험을 정리하면 다음과 같다.

- 계약에 따른 위험 : 사용자와의 계약에 따른 위험
- 요구사항에 관한 위험 : 요구사항 변화에 따른 위험
- 자원 사용 및 성능에 관한 위험 : 분석도구 사용에 따른 위험
- 인적요인에 의한 위험 : 업무 분담에 따른 위험

위험분석은 위험을 발생시킬 수 있는 취약성 노출정도와 심도를 요소로 하여 'Matrix Scaling' 방법과 테이블 평가기준에 의해서 수행된다. 요구사항 분석 단계에서 위험에 대한 분석기준을 제시하면 표 2와 같다.

<표 2> 요구사항 분석 단계의 위험분석 기준

		기 준
등 급	Scale	
Very low	1	다음 단계로 진행해도 문제가 발생하지 않는 경우
Low	2	오류수정 후 다음단계로 진행이 가능한 경우
Medium	3	이전 공정과 현 공정을 재평가한 후 진행하는 경우
High	4	이전 단계로 feedback하여 공정을 다시 진행하는 경우
Very high	5	초기 단계로 Feedback하여 처음부터 공정을 다시 시작하는 경우

취약성 심도와 노출의 평가 기준은 표 3과 4와 같다.

<표 3> 취약성 심도 기준

심 도		기 준
등급	Scale	
Very low	1	각각의 위험요소들이 위험발생에 전혀 영향을 주지 않는 경우
Low	2	각각의 위험요소들이 위험발생에 영향을 주나, 공정에 영향을 끼치지 않는 경우
Medium	3	소수의 위험요소들이 위험을 발생시키나 현 공정만 재평가하는 경우
High	4	다수의 위험요소들이 활성화되어 위험을 발생시켜 위험요소와 관련된 공정을 모두 재평가해야 하는 경우
Very high	5	모든 위험요소들이 활성화되어 위험을 발생시켜 위험요소와 모든 공정을 재평가해야 하는 경우

<표 4> 취약성 노출 기준

노 출		기 준
등급	Scale	
Very low	1	위험이 발생하지 않는 경우
Low	2	위험요소가 거의 활성화되지 않는 경우
Medium	3	위험이 때때로 발생한 경우
High	4	대부분의 위험요소가 활성화되는 경우
Very high	5	모든 위험요소가 수시로 활성화되는 경우

예를 들어 요구분석 단계에서 식별된 4가지 위험요소 중에 3가지가 때때로 발생하고 위험을 발생시키는 경우라면, 취약성 노출은 'High(4)'이고 심도도 'High(4)로 평가할 수 있다. 취약성 요소로 위험을 분석하면 두 요소의 합한 평균값이므로 4가 된다.

이 값을 요구사항 분석 단계의 위험분석 기준으로 평가할 경우 위험등급이 높은 4등급(High)에 해당됨으로 테이블 기준표에 의하면 '이전 단계로 feedback하여 공정을 다시 진행하는 경우'에 해당된다. 그러면 개발자들은 이전 단계의 공정을 처음부터 검토하여 발생한 위험요소들을 재평가하고 진행했던 공정을 검토해야 한다.

전체 공정 중에 요구사항 분석 단계의 위험수준은 본 단계의 SRI 값이 '2.5'이기 때문에 위험분석 값에 곱하여 계산하면 된다. 그러면 위험수준은'10'이 되고, 위험도는 총 공정의 위험정도가 '50'이기 때문에 요구사항 분석 단계의 위험도는 총 공정의 '20%'가 된다.

대응책 수립은 각 위험요소별로 수립하는 것이 바람직하다. 일반적인 대응책은 다음과 같이 선택할 수 있다.

- 사용자와의 계약에 따른 위험 → 보험
- 요구사항 변화에 따른 위험 → 추가 장비 구입, 인건비 등 추가비용에 대한 내용을 계약서 포함
- 분석도구 사용에 따른 위험 → 백업 및 백업장비 구입
- 업무 분담에 따른 위험 → 분야별 전문가 pool 활용

본 논문에서 비용효과분석은 투자수익률로 분석한다. 투자수익률(Investment Return Rate)은 초기투자비용, 연간운영비용, 위험감소에 따른 절약비용으로 계산한다.

- *투자수익률(IR)=[(위험감소 비용-연간운영 비용)/초기투자 비용] × 100*

즉, 투자수익률은 연간 순수이익을 초기 투자비용으로 나눠 백분율로 환산한다.

- *IR=(연간 순수이익/초기투자비용)×100*

예를 들어 분석도구 사용에 따른 위험을 감소시키는 투자수익률을 계산한다. 우선 대응책은 백업 및 백업장비 구입이다. 백업시스템은 SAN(Storage Area Network)으로 여러 가지 디바이스, 설치비용 등 포함해서 구입비용이 5천만원, 연간운영비는 인건비 포함 3천만원, 대응책 수립으로 위험감소에 따른 절약비용은 계약파기 지불금액에서 대응책 수립 비용을 제외한 금액으로 프로젝트 비용이 1억원이고, 대응책 수립 비용이 2천만원이면 8천만원이 된다. 최종적인 투자수익률(IR) 값은 아래와 같다.

IR=[(80000000-30000000)/50000000]×100 =100%

소프트웨어 위험관리자와 기업의 임원들은 100% 투자수익률로 기업의 흑자경영이 가능하다고 판단되면 분석도구 사용에 따른 위험에 따른 대응책을 "백업 및 백업장비 구입"으로 선택하면 된다. 만약 불가능하다고 판단되면 위험에 따른 대응책을 다시 선택해야 한다.

Ⅳ. 결론

본 논문에서는 위험분석 모델을 적용한 실용적인 소프트웨어 위험관리 방법인 기본 위험관리 방법(BRIMM)

및 상세 위험관리 방법(DRIMM)을 제시하였다.

BRIMM은 소프트웨어 개발 공정중 중요도가 낮은 단계에서 프로젝트에 큰 영향을 미치지 않는 위험요소에 대해 최소한의 자원, 비용 및 기간을 투자하여 프로젝트의 위험관리 요구사항에 맞는 대책을 수립하는 것이다.

DRIMM은 소프트웨어 개발 공정 중 중요도가 높고 프로젝트에 큰 영향을 줄 수 있는 위험요소에 대한 위험분석, 위험관리 방법, 위험감소 방법 등 체계적인 절차에 의해 수행되는 방법이다. 중요도가 높은 단계에서 위험정도가 높은 위험요소에 대해 상세하게 분석하고 평가함으로써 정확한 대응책을 수립하여 다음 단계들에서 발생할 수 있는 보다 큰 위험을 방지한다. 예를 들어 구현단계에서 자원 사용 및 성능에 관한 위험 항목의 분석도구 사용에 따른 위험 요소에 대한 평가를 수행할 경우 기존의 방식의 위험평가 결과는 '높음', 대책은 '다른 분석도구나 백업장비를 설치해야 함'으로 나타낸다. 그러나 DRIMM은 위험등급이 4등급으로, 테이블 기준표에 의하여 이전 설계단계로 feedback하여 공정을 다시 진행해야 한다. 아울러 위험도가 총 공정의 20%에 해당된다는 결과로 위험정도를 판단할 수 있다. 또한, 대책에서도 백업장비 구입이 최종 투자수익률로 100%이기 때문에 충분히 비용효과 측면에서 효과성을 입증한다.

향후 좀 더 효율적인 위험관리 방법을 위해서는 위험관리 절차 중 위험분석 과정을 연구하여 자동화 분석이 가능하도록 해야 할 것이다.

참고문헌

[1] Ayad Ali Keshlaf and Khairuddin Hashim, "A Model and Prototype Tool to Manage Software Risks," Proceedings of the First Asia-Pacific Conference on Quality Software, 2000, pp. 297-305.

[2] Ronald P. Higuera, et al, "Software Risk Management," Technical Report CMU/SEI-96-012, SEI Carnegie Mellon Univ., 1996.

[3] Richard Fairley, "Risk management for software projects," IEEE Software, Vol. 11, No. 3, 1994, pp. 57-67.

[4] ISO/IEC SC 27/WG2, "Information Technology-Security techniques-Guidelines for the management of IT security-Part3:Techniques for the Management of IT Security" TR 13335-3, 1998.

[5] 윤청, "소프트웨어 공학," 생능출판사, 2000.

[6] Barry W. Boehm, "Software Risk Management: Principles and Practices," IEEE Software, Vol. 8, No. 1, 1991, pp. 32-41.

[7] Kalle Lyytine, et al, "A Framework For Software Risk Management," The Journal of Information Technology, Vol. 11, No. 4, 1996, pp. 275-285.

[8] ISO/IEC SC 27/WG2, "Information Technology-Security techniques- Guidelines for the management of IT security-Part2:Managing and planning IT Security" TR 13335-2, 1997.

[9] 한국정보통신기술협회, "공공정보시스템 보안을 위한 위험분석 표준-위험분석 방법론 모델", 한국정보통신기술협회(TTA), 1998.

[10] ISO/IEC SC 27/WG2, "Information Technology-Security techniques-Guidelines for the management of IT security-Part1:Concepts and models of IT Security", TR 13335-1, 1996.

[11] Janne Ropponennen "Components of Software Development Risk:How to Address Them?-A project Manager Survey", IEEE Transactions on Software Engineering, Vol. 26, No. 2, 2000, pp. 98-112.

■ 저자소개 ■

엄 정 호
Eom, Jung Ho

1994년~현재 대한민국 공군
2008년 성균관대학교 컴퓨터공학과 (박사)
2003년 성균관대학교 컴퓨터공학과 (석사)
1994년 공군사관학교 항공공학과 (학사)

관심분야 : 사이버전, 사이버보안, 접근제어, 위험분석
E-mail : eomhun@gmail.com

이 동 영
Lee, Dong Young

2003년~현재 명지전문대학교 부교수
2003년 성균관대학교 전기전자 및 컴퓨터공학과 (박사)
1998년 성균관대학교 정보공학과 (석사)
1993년 동아대학교 전자공학과 (학사)

관심분야 : 네트워크 관리, 정보보호, USN
E-mail : dylee@mail.mjc.ac.kr

정 태 명
Chung, Tai M.

1995년~현재 성균관대학교 컴퓨터공학과 교수
1995년 Purdue University W. Lafayette, IN, U. S. A. 컴퓨터공학 졸업 (박사)
1987년 University of Illinois Chicago IL, U. S. A. 컴퓨터공학과 졸업 (석사)
1984년 University of Illinois Chicago IL, U. S. A. 전자계산학과 졸업 (학사)
1981년 연세대학교 전기공학과 졸업 (학사)

관심분야 : 통합보안관리, 네트워크, 무선망
E-mail : tmchung@ece.skku.ac.kr

논문접수일 : 2009년 5월 4일
수 정 일 : 2009년 5월 26일
게재확정일 : 2009년 6월 1일