특집 05

프로세스 모델링과 시뮬레이션을 통한 소프트웨어 프로세스 개선

목 차

류호연 · 백중문
(한국정보통신대학교)

1. 서 론

성공적인 프로젝트 수행과 고품질의 소프트웨어 제품을 개발하기 위해서는 잘 정의되고 프로젝트 특성에 맞게 테일러된 소프트웨어 프로세스가 필요하다. 소프트웨어 프로세스는 소프트웨어의 개발을 위해 수행되어야 하는 작업들과 그 작업들 간의 연결, 각 작업의 입력과 출력, 작업의 책임자 그리고 작업에 할당되는 자원들의 집합으로 지속적인 개선 활동을 통하여 효율적인 프로젝트 관리가 가능하며 이를 통해 고품질의 소프트웨어 제품을 획득할 수 있게 된다. 현재 CMM(Capability Maturity Model), CMMI (Capability Maturity Model Integration), SPICE(Software Process Improvement Capability Determination, ISO 15504) 등과 같은 소프트웨어 프로세스 모델들이 개발되어 있으며 이들은 프로세스 개선의 효과 및 결과에 대한 정의, 구현 및 평가를 위한 통계적 관리 및 측정에 관한 지침을 제공한다.

프로세스 모델링과 시뮬레이션은 초기에 제조 분야에서 산업 프로세스를 이해하고 개선하기 위한 도구로 사용되었으나, 최근에는 소프트웨어 개발 프로세스 분야에서도 유용한 도구로 사용되고 있다. 이러한 소프트웨어 프로세스 모델링과 시뮬레이션은 소프트웨어 프로세스 개선과 직접적인 연관성을 가지며, 조직은 성숙도 각 수준에서 이들을 사용할 수 있다[1,2]. 즉 프로세스 모델링과 시뮬레이션은 프로젝트 수행을 위한 프로세스의 정립 및 개선, 프로세스의 변화에 따른 파급 효과, 그리고 기술의 진화 및 새로운 기술 도입에 따른 영향력 등을 파악하기 위해 적용된다. 즉, 프로세스 모델링과 시뮬레이션은 소프트웨어 개발의 전략적 관리에서부터 프로세스 개선 지원, 소프트웨어 프로젝트 관리를 위한 훈련에 이르는 다양한 분야에 적용될 수 있으며 최종적으로는 효율적인 프로젝트의 수행으로 고품질의 제품을 개발 할 수 있도록 지원한다[1,3].

프로세스 모델은 실제 또는 개념적인 복잡한 시스템에 대한 간략화된 논리적 표현 또는 추상화를 의미하는 것으로 시스템의 중요한 특성과 특징을 나타낸다. 시뮬레이션은 파악된 특징들을 가지며 동적 체계 또는 현상을 표현할 수 있는 컴퓨터화된 모델을 의미한다. 이러한 프로세

스 모델을 개발하고 시뮬레이션을 수행하는 이유는 관심의 대상이 되는 실제 시스템의 비용, 위험, 또는 계획 등을 저비용으로 초기에 알 수 있기 때문이다. 프로세스 모델링과 시뮬레이션은 소프트웨어 프로세스뿐만 아니라 제조 시스템의 분석 및 설계, 국방 무기체계들의 평가, 통신 시스템에서의 프로토콜 및 요구사항 결정, 항공, 지하철 등의 이동 시스템들의 설계 및 실험, 비즈니스 프로세스의 평가 및 재공학 등 다양한 분야에 적용되고 있다. 프로세스 모델링과 시뮬레이션은 연속 모델링(continuous modeling), 이산 모델링(discrete-event modeling), 혼합 모델링(hybrid modeling) 등의 모델링 방법을 선택적으로 사용한다[4]. 예를 들면, 실제 자원을 투입하거나 프로세스를 적용해 보지 않더라도, 모델링 결과를 시뮬레이션 함으로써 그 파급효과나 결과를 예측할 수 있게 된다. 또한 정량적인 평가를 가능하게 하여 실제 투입될 비용, 가능한 스케줄, 최종 제품의 품질 등을 판단할 수 있는 근거를 제시해 준다. 즉, 프로세스 모델링과 시뮬레이션을 수행함으로써, 위험 관리(risk management), 계획(planning), 제어 및 운영 관리(control and operational management), 프로세스 개선 및 기술 채택(process improvement and technology adoption), 프로세스 이해(process characterization and understanding), 훈련과 교육(training and learning) 등의 이점을 얻을 수 있게 된다[1].

본 연구에서는 프로세스 모델링과 시뮬레이션을 통한 소프트웨어 프로세스 개선 방안에 대해 소개한다. 이를 위해 프로세스 모델링과 시뮬레이션의 개념을 소개하고 이를 이용한 시스템 테스트 프로세스 모델링과에 대해 소개하고 적용 가능한 분야들에 대해 기술한다.

2. 프로세스 모델링과 시뮬레이션

본 장에서는 모델링 방법에 대해 살펴보고 시뮬레이션의 유용성을 설명하도록 한다. 또한 소프트웨어 프로세스 개선을 위한 소프트웨어 프로세스 시뮬레이션에 대해 기술한다.

2.1 시뮬레이션 모델링

모델링을 위한 방법에는 연속 모델링(continuous modeling), 이산 모델링(discrete-event modeling), 혼합 모델링(hybrid modeling)이 있다[4]. 이산 모델링은 상태 변수가 이벤트 시점에 변경되는 것을 순간적으로 표현한 것으로써, 예를 들면 제품계열(product line)에서 어떤 시점에서 제품의 개수가 바뀌어 이벤트가 바뀌는 것을 나타낸 것이다. 이를 위한 도구에는 Arena [8], AweSim[9], Extend[10] 등이 있다. 연속 모델링 방법은 특정 시점에서 구별되게 값이 변하는 것이 아니라, 시간에 따라 그 양이 계속 변해가는 것으로 SIMULINK[11], Dymola [12], iThink[13] 등의 도구를 이용할 수 있다. 그리고 이들을 통합해서 적용하는 방법이 혼합 모델링으로 AweSim, Extend 등의 도구가 이용된다.

〈표 1〉은 모델링 방법에서 연속 모델과 이산 모델을 비교한 것이다. 연속 모델은 프로세스와 제품 속성의 연속적인 변화에 초점을 두고 있기 때문에 프로세스와 제품 속성의 변경으로 발생할 수 있는 예상 밖의 부작용을 예측할 수 있지만, 개별 객체와 속성이 존재하지 않는 단점이 있다. 이산모델은 각 객체에 대한 유일한 속성 정의가 가능하여 각 활동에서 객체별로 변화에 따라 어떤 결과가 도출 될 수 있는지 예측할 수 있지만 오직 이벤트 시간에만 변수를 변경할 수 있는 단점을 가지고 있다. 각각의 모델들은 적용 분야에 따라 선택적으로 사용가능하며 경우에 따라 혼합 모델로 사용될 수 있다.

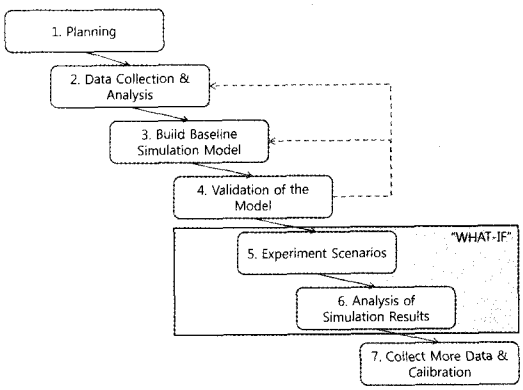
2.2 왜 시뮬레이션을 수행하는가?

소프트웨어 프로세스 모델에서 시뮬레이션을

수행하는 목적은 다양하다. 많은 경우에 시뮬레이션은 주요한 의사결정을 지원하며 위험을 줄이고 전략적, 전술적 그리고 운용적 관리가 가능하도록 한다. 소프트웨어 프로세스 모델에서 시뮬레이션을 수행하는 목적을 요약하면 다음과 같다[1,3].

- 전략적 관리(strategic management)
- 계획(planning)
- 제어 및 운용 관리(control and operational management)
- 프로세스 개선 및 기술 채택(process improvement and technology adoption)
- 이해(understanding)
- 훈련과 교육(training and learning)

(그림 1)은 시뮬레이션 모델링 단계를 나타낸 것으로 7단계로 구성된다. 1단계인 계획단계에서는 시뮬레이션의 목적과 범위를 정의하여 시뮬레이션 모델의 범위와 모델의 추상화 수준을 정한다. 일반적으로 시뮬레이션의 범위는 생명주기의 각 단계, 단일 제품을 개발하는 프로젝트, 동시에 진행되는 프로젝트들, 오랜 기간 동안 개발 및 유지보수 되는 제품 및 조직이 될 수 있다. 2단계인 데이터수집 및 분석 단계에서는 입력 인자, 출력 인자, 그리고 입출력 인자들 사이의 관계를 정의한다. 입력 인자로는 소프트웨어의 규모(LOC), 기능점수, 고용 비율, 개인의 능력



(그림 1) 시뮬레이션 모델링 단계

자원의 제약사항, 등이 포함되며, 출력 인자로는 노력 및 비용, 반복 시간, 결함 수준, 직원들의 능률, ROI 등이 있다. 3단계 베이스라인 시뮬레이션 모델 수립에서는 1단계와 2단계의 정보를 바탕으로 주요 활동과 업무, 기본 객체, 주요 자원, 활동들간의 의존성 및 각 활동에서의 객체의 흐름, 결정 시점 등의 고려하여 프로세스 추상화를 통해 기본 시뮬레이션 모델을 수립한다. 모델 검증을 수행하는 4단계 과정을 통해 피드백 2단계 또는 3단계로의 피드백을 결정하게 되며 수립된 모델을 정제해 나가게 된다. 이후 "WHAT-IF" 과정에서는 검증된 시뮬레이션 모델을 이용하여 시뮬레이션을 수행하게 된다. 5단계의 실험을 통해 시뮬레이션을 수행하게 되며 다양한 대체 모델들을 개발할 수 있게 된다. 또한 6단계의 시뮬레이션 결과 분석에서는 시뮬레이션의 결과와

〈표 10〉 연속 모델과 이산 모델의 비교[4]

	연속 모델(Continuous Model)	이산 모델(Discrete Event Model)
장점	- 프로세스&제품 속성의 연속적인 변화를 표현 - 안정적 또는 비안정적인 피드백 확인 가능 - 프로세스와 제품 속성의 변경으로 발생할 수 있는 예상 밖의 부작용 예측	- 큐(queue)를 표현하기 쉬움 - 이용 가능한 자원을 기반으로 B/W 활동 지연 - 각 객체에 대한 유일한 속성 정의 - 각 활동에서 객체별로 변화에 대한 결과를 확인할 수 있음 - 상호 의존적인 B/W 활동 파악 가능
단점	- 프로세스 단계를 기술하기 어려움 - 개별 객체와 속성이 없음 - 각 모델을 실행하는데 동일한 비율을 적용함	- 오직 이벤트 시간에만 변수를 변경할 수 있음 - 동시에 일어나는 활동을 표현하기 어려움
적용 분야	경제(economics), 시스템 다이나믹스(system dynamics), 과학적 프로세스(생물학, 화학, 물리학), 전기(electronics), 제어 시스템(control systems)	비즈니스 프로세스 재공학(business process reengineering), 네트워크(networks), 시스템 공학(systems engineering), 제조(manufacturing)

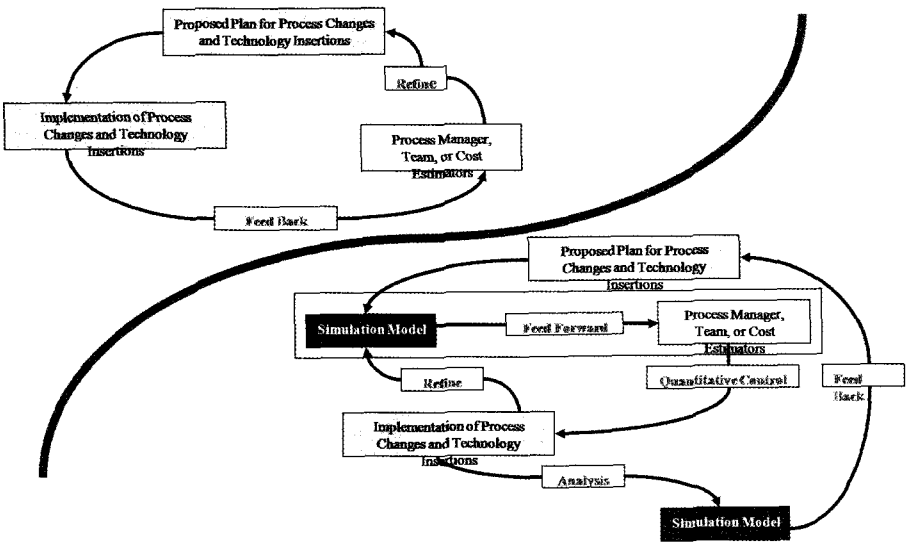
실제 시스템과의 성능을 비교할 수 있으며 그 결과를 바탕으로 더 많은 데이터를 수집하거나 교정과정을 거치게 된다.

2.3 소프트웨어 프로세스 시뮬레이션

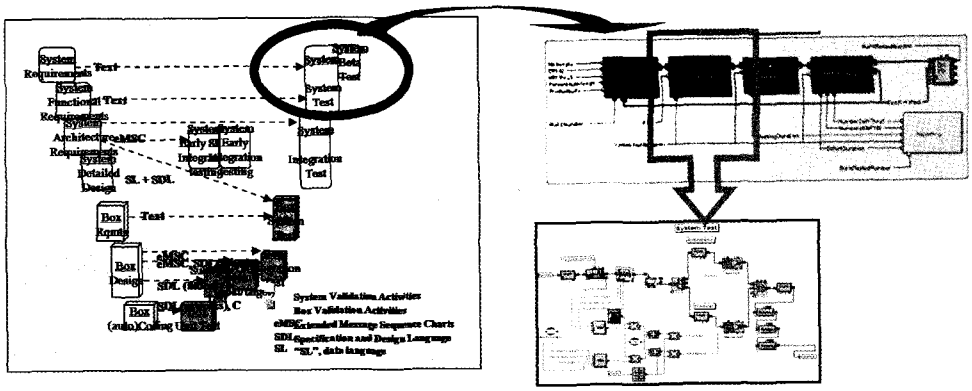
소프트웨어 개발은 매우 동적이며 복잡한 프로세스에 따라 이루어지기 때문에 시뮬레이션을 통해 소프트웨어 프로세스 개선 전략을 효과적으로 수립할 수 있게 된다. 즉 프로세스 모델링과 시뮬레이션을 소프트웨어 프로세스 분야에 적용하는 경우 전체 시스템 관점에서 프로세스 개선을 도모할 수 있으며, 프로세스 피드백 결과를 이용할 수 있게 된다. 또한 비용, 일정 및 품질에서의 우선순위 결정이 용이하게 되며 모델링된 결과를 바탕으로 일정에 따른 임계 경로 분석이 가능하다. 결과적으로 저비용으로 다양한 실험을 수행할 수 있게 된다. 이러한 시뮬레이션의 이점을 요약하면 다음과 같다.

- 복잡한 프로세스의 특성을 파악할 수 있다.
- 새롭게 제안되는 프로세스에 대한 변경이나 기술 도입에 대한 정량적 분석이 가능하여 의사결정과 위험 평가가 가능하다.
- 소프트웨어 프로세스 모델(CMM, CMMI 등)에서 더 높은 수준을 달성할 수 있다. 이는 정량적인 프로세스 관리, 소프트웨어 품질 관리, 프로세스 및 기술 변경 관리 및 지속적인 프로세스 개선을 지원하기 때문이다.
- 노력, 일정, 비용 추정을 포함한 효율적인 프로젝트 계획 수립이 가능하다.

(그림 2)는 프로세스 변경과 새로운 기술 도입 계획이 있는 경우 프로세스 모델링과 시뮬레이션을 과정을 거치는 경우와 일반적인 프로세스에 따르는 프로젝트가 어떻게 달라지는 보여주는 그림이다. 그림의 위쪽은 일반적인 프로세스를 적용하는 경우로써, 프로세스 변경과 새로운 기술 도입 계획에 대해 구현을 통해 프로세스를 관리자, 예상되는 팀원의 수, 비용 등을 추정하게 되며, 이 추정된 결과를 바탕으로 정제 과정을 통해 계획을 수정하여 적용하게 된다. 이 경우, 일단 계획하는 변경사항에 대한 구현 이후에야 관리의 효율성이나 비용 추정 등을 통해 프로세스 개선 및 품질에 대한 평가가 이루어지므로 고비용이 소요되는 것이 일반적이다. 그러나 아래 그림의 경우 변경사항에 대한 초기 시뮬레



(그림 2) 프로세스 모델링과 시뮬레이션의 적용



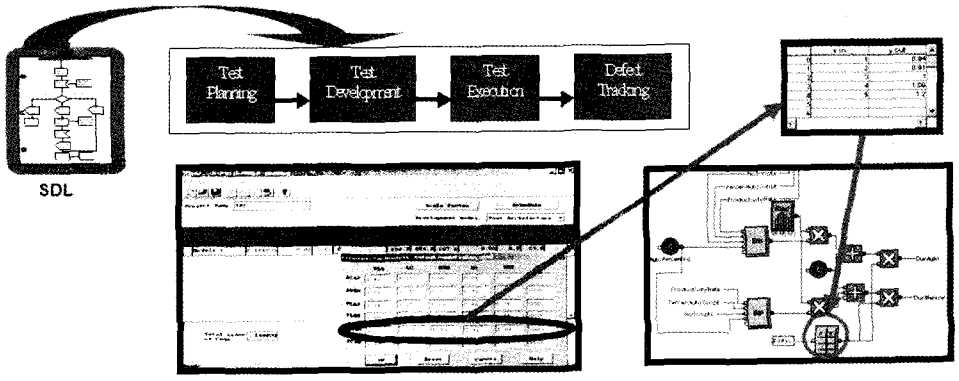
(그림 3) 시스템 테스트 프로세스 시뮬레이션 과정

이션 모델을 수립하고 시뮬레이션 과정을 통해 프로세스를 관리자, 예상되는 팀원의 수, 비용 추정 등의 데이터를 정량적으로 제어함으로써 변경사항에 대한 유효성을 예측할 수 있게 한다. 결과적으로는 시뮬레이션과 정량적 관리의 결과로 이미 평가된 프로세스 변경 내용과 새로운 기술 도입이 이루어지게 되며 피드백 과정을 통해 다양한 개선활동들이 이루어질 수 있게 되는 것이다.

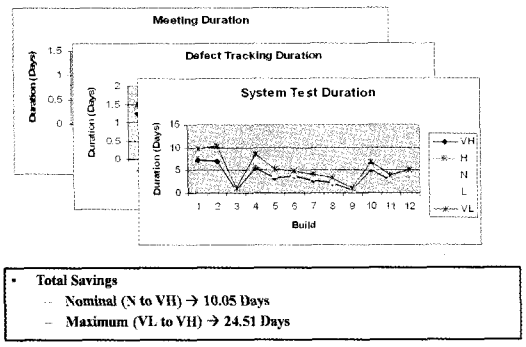
3. 예제 : 시스템 테스트 프로세스 시뮬레이션

테스트 프로세스는 제품의 행위, 성능, 견고성이 기대하는 기준에 부합하는 지를 평가하기 위한 프로세스로서, 검증 및 확증 활동을 핵심으로 하며 일반적인 소프트웨어프로세스 모델들에서 언급하고 있으나 현실적인 테스트 프로세스 개선 활동을 위해서는 부족함이 있다. 본 장에서는 예제를 통하여 효과적인 테스트 프로세스 수립과 개선을 시뮬레이션 과정을 살펴보도록 한다.

다음 (그림 3)에 나타나 있는 것은 시스템 테스트 프로세스 시뮬레이션 과정을 도식화한 것이다. 시스템 테스트 프로세스 시뮬레이션을 위해서는 우선 현재 시스템 테스트 프로세스에 대한 베이스라인을 수립하고, 이를 바탕으로 프로세스 모델링 과정을 통해 시스템 테스트 프로세스를 시뮬레이션 모델을 만든다. 시스템 테스트 프로세스 시뮬레이션 모델이 완성된 이후에는 시뮬레이션 과정을 통해 프로세스의 변경이나 새로운 기술의 도입이 일정, 품질, 노력 등에 미치는 파급 효과를 알아 낼 수 있을 것이다.



(그림 4) LTEX의 결과를 이용한 시뮬레이션의 예



(그림 5) LTEX에 의한 일정 단축

(그림 4와 5)의 경우를 살펴보자. 이 예에서는 널리 사용되고 있는 비용 모델 중의 하나인 COCOMO II[14]의 인자를 이용해서 시뮬레이션을 수행하고 그 결과를 바탕으로 일정을 추정하는 것을 보여주고 있다. 즉 시스템 테스트 프로세스에 Tau의 SDL/MSC를 도입했을 때의 효과를 평가하고자 할 때, COCOMO II의 LTEX (language & Tool Experience)을 이용하여 SDL/ MSC를 도입할 경우 SDL의 순위(1-5)가 LTEX의 순위 VL에서 VH에 순위 매김됨을 파악할 수 있으며 결과적으로는 그림 5에 나타난 것처럼 기본(N→VH) 10.05일, 최대(VL→VH) 24.51일의 일정 단축이 가능하다는 것을 알 수 있다.

4. 결 론

소프트웨어 프로젝트에서 대부분의 주요 결정사항들은 생명주기 중, 초기에 만들어지지만, 해당 솔루션의 실현 가능성을 평가하기에는 이용가능한 정보가 부족한 편이다. 이를 해결하기 위해 프로토타이핑, 성능 모델링, 비용 모델링 등의 유용한 도구들과 기술들이 존재하지만, 이들만으로는 복잡한 소프트웨어 프로젝트에서 개발 초기에 더 높은 가치를 창출하기 위한 주요 결정사항들에 대한 평가를 내리기가 어렵다. 특히, 소프트웨어 프로세스 개선 분야에 있어서는 프로젝트 수행을 위한 프로세스의 정립 및 개선, 프로세스의 변화에 따른 파급 효과, 그리고 기술의 진화 및 새로운 기술 도입에 따른 영향력 등을 파악하여야 하기 때문에 보다 정성적이고 정량적인 평가 방법이 필요하다.

다양한 분야에서 적용되고 있는 모델링은 강력한 도구로서 복잡한 시스템을 분석하거나 실제 시스템을 위한 프로세스를 평가하기 위해 사용된다. 즉 모델링은 효율적인 대화의 도구로서 구현 이전에 해당 시스템이 어떻게 운용될 것인지, 어떻게 개선될 수 있는지를 파악할 수 있게 하며 효과적인 전략을 수립할 수 있게 한다. 본 연구에서 기술한 프로세스 모델링과 시뮬레이션은 소프트웨어 개발의 전략적 관리에서부터 프로세스 개선 지원, 소프트웨어 프로젝트 관리에 이르는 다양한 분야에 적용될 수 있으며 최종적으로는 효율적인 프로젝트의 수행으로 고품질의 제품을 개발 가능하게 한다.

참고문헌

[1] David M. Raffo, "Getting the Benefits from Software Process Simulation," International Conference on Software Engineering and Knowledge Engineering(SEKE'99), June 1999.

[2] M. Ruizand I. Ramos and M. Toro, "A Dynamic Integrated Framework for Software Process Improvement," Software Quality Journal, 10, pp.181- 194, 2002.

[3] Marc I. Kellner, Raymond J. Madachy, David M. Raffo, "Software Process Simulation Modeling: Why? What? How?," Journal of Systems and Software, Vol. 46, No. 2/3, 1999.

[4] Martin, R.H. and Raffo, D.M., "A Model of the Software Development Process

Using Both Continuous and Discrete Models," International Journal of Software Process Improvement and Practice, Vol. 5, No. 2/3, June/September, 2000.

[5] Barros, M.O., Werner, C.M.L., Travassos, G.H., "Using Process Modeling and Dynamic Simulation to Support Software Process Quality Management," XIV Simp-sio Brasileiro de Engenharia de Software, Workshop de Qualidade de Software, October 2000.

[6] Burke, S., "Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization", Technical Report CMU/SEI-96-TR- 024, Software Engineering Institute, Carnegie Mellon University, Pittsburg, USA, 1996.

[7] Law, A. and Kelton, W., Simulation Modeling & Analysis, McGraw Hill, 1991.

[8] Arena, http://www.arenasimulation.com/

[9] A. Alan Pritsker, Jean J. O'Reilly, Simulation with Visual Slam and Awesim, John Wiley & Sons, Inc., New York, NY, 1999.

[10] Extend, http://www.imaginethatinc.com/

[11] SIMULINK, http://www.mathworks.com/products/ simulink/

[12] Dymola, http://www.dynasim.se/index.htm

[13] iThink, http://www.hps-inc.com

[14] Jongmoon Baik and Nancy Eickelmann, "Applying COCOMO II Multipliers to Simulation Models", 16th International COCOMO/Software Cost Modeling Forum.

저자약력

류 호 연

2000년 경상대학교 정보시스템 석사
2005년 경상대학교 정보시스템 박사
2005년 경성대학교 초빙외래교수
2005년~2006년 한국과학기술연구원 Post-Doc
2006년~현재 한국정보통신대학교 공학부 연구교수
관심분야 : 소프트웨어 테스트, 소프트웨어 신뢰성, 소프트웨어 보안, 소프트웨어 프로세스 개선, 온톨로지 공학
이 메 일 : hoyeon@icu.ac.kr

백 종 문

1993년 조선대학교 전산통계학과 학사
1996년 University of Southern California Computer Science 석사
2000년 University of Southern California Computer Science 박사
2001년~2005년 미국 모토로라 SSERL (Software and Systems Engineering Research Lab) 수석연구원
2005년~현재 한국정보통신대학교 공학부 조교수
관심분야 : 소프트웨어 메트릭스, 소프트웨어비용추정, 소프트웨어 동적모델링, 소프트웨어 프로세스개선, 소프트웨어 품질보증, 소프트웨어 6 시그마
이 메 일 : jbaik@icu.ac.kr