비기능적 요구사항을 지원하는 소프트웨어 개발 프로세스

論 文
9-1-3

A Software Development Process Supporting Non-Functional Requirements

정 효 택*, 주 상 현*
Hyo Taeg Jung and Sang Hyun Joo

Abstract

As the importance of non-functional requirements has increased, many researchers have become interested in the software development process for non-functional requirements including the notation, modeling, and assessment of non-functional requirements. However, the characteristics of non-functional requirements are so sophisticated and there are many topics which have not been solved until now. In order to address one of the unsolved problems, we propose a systematic software development process to support the management of non-functional requirements. The process consists of six steps, each of which is composed of detailed activities. Using the proposed process, the non-functional requirements can be managed and modeled more effectively and systematically than previous ones.

Keywords : software development process, non-functional requirements, performance, security, availability, Unified Modelling Language (UML), Colored Petri Net (CPN)

I. 서 론

요구공학의 발전과 함께 소프트웨어의 비기능적 요구사항(Non-Functional Requirement, NFR)의 중요성이 강조됨에 따라, 많은 공학자들은 소프트웨어의 기능적 요구사항은 물론 비기능적 요구사항을 효과적으로 지원하는 소프트웨어 개발 프로세스들을 제안하는 노력을 기울여왔다.

일반적인 소프트웨어 개발 프로세스들은 선형(linear), 비선형(non-linear), 진화형(incremental), 나선형(spiral) 등의 다양한 형태로 제안되었다. I. Sommerville은 활동(activities)의 반복(iteration)을 허용하는 개념적 선형 요구사항 프로세스 모델을 제안하였다[1]. 반면에 L. A. Macaulay는 활동들의 반복(iteration)이나 중복(overlap)을 허용하지 않는 순수 선형 요구사항 프로세스 모델을 제안하였다[2]. 이러한 선형 모델과는 달리 비선형 또는 나선형 형태의 모델도 개발되어 왔다. 예를 들면 P. Loucopoulos는 반복적이고 주기적인 프로세스 모델을 제안하고, B. W. Boehm은 일련의 활동들을 반복적으로 수행함으로써 점차적으로 프로세스를 진행시키는 나선형 모델을 제안하였다[3-4].

K. Sousa는 인터랙티브 시스템 개발에 있어서 사용성(usability)을 제공함은 물론 프로세스 내에 HCI(Human Computer Interaction) 개념을 도입한 사용자 인터페이스를 제공하기 위해 소프트웨어 공학과 HCI가 통합된 소프트웨어 개발 프로세스인 UPi를 제안하였다[5]. UPi 프로세스는 IBM RUP (Rational Unified Process)와 통합하여 개발환경에 맞게 커스터마이징(customizing)되거나 테일러링(tailoring) 될 수 있다. UPi 프로세스는 요구사항의 점검, 프로토타입의 활용, 반복적인 설계 기법 등을 허용함으로써, 사용자들에게 고수준의 사용성을 제공할 수 있으며 이로 인해 사용자 만족도를 높일 수 있다. 또한 UPi는 사용 패턴과 프로토타입의 활

접수일자 : 2010년 2월 11일
최종완료 : 2010년 3월 02일
*한국전자통신연구원 콘텐츠연구본부
교신저자, e-mail : joos@etri.re.kr

용을 통한 클래스의 재사용을 제공해주기 때문에 문제를 쉽게 발견할 수 있고 결국 저비용으로 소프트웨어 개발이 가능하게 한다. 하지만 UPi의 활동들과 가이드라인 등이 RUP의 개발 생명주기에 의존적으로 매핑 되어 있는 등 RUP와 L. Constantine이 제안한 방법론에 많은 영향을 받고 있다[6].

S. Rottger는 상이한 추상화 레벨에 따른 측정(measurements)에 기반한 소프트웨어 개발 프로세스를 제안하였다[7]. 여기서 측정은 반응시간이나 지연시간 등 시스템의 품질 인자로 간주되고 측정될 수 있는 값을 말한다. 제안한 프로세스에서는 초기 개발 단계에서 추상적인 비기능적 요구사항을 명세, 정제함으로써, 정제된 비기능적 요구사항의 제한점들을 표현할 수 있는 기능을 제공한다. 시스템의 비기능적 요구사항을 정제하기 위해서는 측정과 관련된 컨텍스트 모델이 명확하게 정의하여야 하는데, 제안한 프로세스는 규모가 크고 대량의 정보를 지닌 모델들의 복잡성을 줄이는 방법은 제시하지 않고 있다.

현재까지 공학자들이 해결하려고 노력하는 문제점 중의 하나는 비기능적 요구사항을 개발 프로세스의 어느 부분에서 처리해야 하는 문제인데, 예를 들면 운용 시스템(operational system)의 비기능적 요구사항은 프로세스 자체 혹은 프로덕트 등과 관련된 일종의 제한사항으로 간주되고 있다. V. Sivess는 의사 결정시에 Calculus of Communication Systems (CCS)를 사용하는 모델링 접근법을 제안하였다[8-9]. CCS 표기법은 상이한 다큐먼트 상태(document states)에서 상이한 역할 에이전트들(role agents)에 의해 수행되는 다양한 활동들을 표현하는데 적합하다. Pi-calculus는 활동들 사이의 새로운 추적관계나 제한(constraints)이 발생할 경우 이를 모델링하는데 사용된다. 제안된 접근법은 정형명세를 사용하기 때문에 복잡한 다큐먼트 상태에 종속적인 일련의 프로세스들을 매우 자연스럽게 표현할 수 있다. 또한 이 접근법은 소프트웨어 프로세스내에서 비기능적 요구사항을 처리함으로써 저비용으로 소프트웨어의 이주(migration)가 가능하도록 일반적인 가이드라인을 제공한다. 하지만 정형명세를 사용하기 때문에 지원도구 없이 프로세스를 사용하기가 수월하지 않은 단점이 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 성능, 보안 및 유용성과 같은 비기능적 요구사항을 지원하는 소프트웨어 개발 프로세스를 제안하고, 제3장에서는 간단한 현금자동지급기의 시뮬레이션을 통하여 제안한 프로세스의 주요 단계들의 검증을 시도하였다. 마지막 장에서는 향후 연구 방향과 결론을 내린다.

II. 비기능적 요구사항을 지원하는 소프트웨어 개발 프로세스

본 장에서 제안한 프로세스는 비기능적 요구사항을 포함하는 소프트웨어 개발을 지원하는 소프트웨어 개발 프로세스를 제안한다. 제안하는 프로세스는 크게 여섯 단계로 구성되어 있다. 즉, 비기능적 요구사항을 가공(Elaborating NFRs), 모델 개발(Developing NFRs Model), 추정(Estimating NFRs), 평가(Assessing NFRs), 정형명세(Specifying NFRs in a Formal Method), 분석(Analyzing NFRs)하는 등의 여섯 단계이다. 그림 1은 제안한 프로세스의 전체 구조를 데이터 흐름 다이어그램(Data Flow Diagram, DFD)를 이용하여 보여주고 있다.

제안한 프로세스는 비기능적 요구사항이 포함된 소프트웨어 개발에 광범위하게 적용할 수 있지만, 여러 응용분야에서 공통적으로 요구되는 성능(performance), 보안(security), 유용성(availability)과 관련한 활동들을 대표적으로 제시하였다. 시스템 특성에 따라 요구되는 비기능적 요소에 적용되기 위해서는 각 단계에서 행해야 할 세부적인 활동들이 커스터마이징 될 수 있다.

1. 1단계 - NFRs 가공

비기능적 요구사항 가공 단계에서는 비기능적 요구사항을 도출, 확정, 정제하는 활동들로 구성된다. 사용자의 요구(needs)나 개략적인 요구사항 명

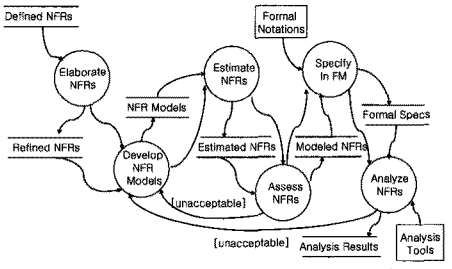


그림 1. 제안한 프로세스 전체 구조도
Fig. 1. The overall structure of proposed process

세서(Software Requirements Specification, SRS)가 입력되고, 결과로는 정제된 요구사항 명세서(Refined SRS)가 작성된다.

o 성능: 반응시간, 자원 활용도, 처리량 등을 상세하게 정제한다.
o 보안: 안전성이 확보되어야 할 자원을 식별하거나 분류한다.
o 유용성: 제공되는 서비스를 정제하거나 타입에 따라 분류하거나 서비스의 레벨을 정의한다.

2. 2단계 - NFRs 모델 개발

비기능적 요구사항 모델 개발 단계에서는 정제된 요구사항으로부터 적절한 매트릭스나 모델을 개발하는 활동들로 구성된다. 전단계에서 작성된 정제된 요구사항 명세서가 입력되거나 결과로는 모델이나 매트릭스가 개발된다. 요구사항 평가 혹은 분석 단계에서 그 결과가 수락되지 않을 경우 재실행된다.

o 성능: 성능과 관련된 요소를 식별하고 그들간의 관계를 식별하고 성능 모델 혹은 성능 메트릭스를 생성한다.
o 보안: 보안정책과 관련된 요소를 식별하고 신뢰 모델(trust model)을 만든다. 또한 위험 요소를 식별하고 위험 모델(threat model)을 생성한다.
o 유용성: 유용한 자원을 식별하고 정상적인 유용성 모델(availability model)을 만든다. 유용하지 않은 자원을 식별하고 비유용성 모델(unavailability model)을 만든다.

3. 3단계 - NFRs 추정

비기능적 요구사항 추정 단계에서는 전단계에서 생성된 모델이나 메트릭스를 측정, 추정, 테스팅하는 활동들로 구성된다. 전단계에서 생성된 모델이나 메트릭스가 입력되고, 비기능적 요구사항의 추정 값들이 생성된다.

o 성능: 성능 요구사항을 추정하고 실제적인 테스팅을 한다.
o 보안: 신뢰 모델과 위협 모델을 추정하여 탐지(detection)나 회복(recovery)기법을 디자인한다.
o 유용성: 플랫폼의 유용성을 추정하고 기능적 유용성을 설계한다.

4. 4단계 - NFRs 평가

비기능적 요구사항 평가 단계에서는 SRS에 명기되어 있는 비기능적 요구사항이나 혹은 설계 모델에 명기되어 있는 비기능적 요구사항을 평가하는 활동들로 구성된다. 전단계에서 생성된 추정값들이 입력되고, 평가결과들이 산출된다. 평가결과가 수락되지 않으면, 수락될 때 까지 2, 3 단계의 작업들이 반복된다.

o 성능: 성능 특성을 평가하고, 요구사항과 소프트웨어 아키텍처의 일관성을 점검한다.
o 보안: 위험 인자를 정의하여 그 해법을 찾으며, 관련 요구사항과 소프트웨어 아키텍처의 일관성을 점검한다.
o 유용성: 플랫폼과 기능적 유용성을 통합하고, 위험 요소를 식별하고 그 해법을 찾으며, 관련된 요구사항과 소프트웨어 아키텍처의 일관성을 점검한다.

5. 5단계 - NFRs 정형명세

비기능적 요구사항 정형명세 단계에서는 전단계에서 생성된 비기능적 요구사항을 정형기법을 사용하여 명세하는 활동으로 구성된다. 비기능적 요구사항은 객체의 행위(behavior), 구조(structure) 및 상호 인터랙션(interaction)을 표현할 수 있는 UML (Unified Modeling Language) 모델로 표현이 가능하다. Petri Nets, LOTOS, Z-notation, Calculus of Communication Systems(CCS), Communicating Sequential Process(CSP) 등의 기법들이 UML 모델을 정형명세 하는데 사용될 수 있다. 입력은 비기능적 요구사항이 표현된 소프트웨어 아키텍처이며, 결과물은 정형명세이다[8, 10-13].

6. 6단계 - NFRs 분석

마지막 단계인 비기능적 요구사항 분석 단계에서는 정형명세된 비기능적 요구사항을 분석한다. UML 모델에 명기된 비기능적 요구사항들 시뮬레이션하고 그 결과를 분석한다. 만약 분석된 자료가 수락이 되지 않으면, 수락될 때 까지 2단계 이후의

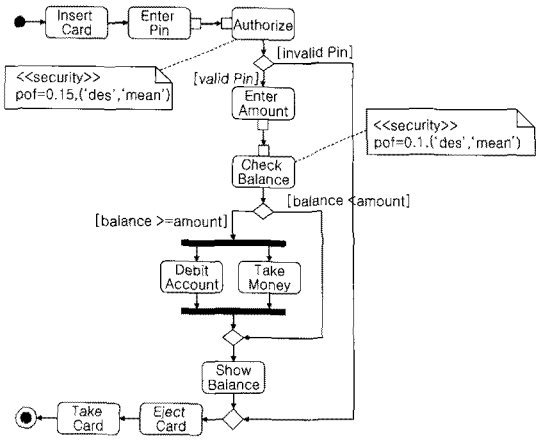


그림 2. 비기능적 요구사항(PoF)을 포함한 액티비티 모델
Fig. 2. An activity model including Non-Functional Requirements(PoF)

활동들을 반복 수행한다. 입력은 정형명세된 비기능적 요구사항이고, 결과물은 시뮬레이션 결과를 분석한 정보이다.

III. 시뮬레이션

이 장에서는 2장에서 제안한 프로세스 중 비기능적 요구사항을 정형명세하고 분석하는 5, 6단계를 간

표 1. 시뮬레이션 결과
Table 1. The result of simulation

samples	Case 1: PoF of Authorize=0.20 PoF of CB=0.10			Case 2: PoF of Authorize=0.15 PoF of CB=0.10		
	Failure of ATR	Failure of CB	No Failure	Failure of ATR	Failure of CB	No Failure
1-50	14 28.0%	3 6.0%	33 66.0%	9 18.0%	2 4.0%	39 78.0%
51-100	22(8) 22.0%	6(3) 6.0%	72(39) 72.0%	15(6) 15.0%	8(6) 8.0%	77(38) 77.0%
101-150	30(8) 20.0%	8(2) 5.3%	112(40) 74.7%	23(8) 15.3%	12(4) 8.0%	115(38) 76.7%
151-200	43(13) 21.5%	15(7) 7.5%	142(30) 71.0%	33(10) 16.5%	15(3) 7.5%	152(37) 76.0%
	29.0%		71.0%	24.0%		76.0%

samples	Case 3: PoF of Authorize=0.10 PoF of CB=0.10			Case 4: PoF of Authorize=0.05 PoF of CB=0.10		
	Failure of ATR	Failure of CB	No Failure	Failure of ATR	Failure of CB	No Failure
1-50	7 14.0%	2 4.0%	41 82.0%	2 4.0%	3 6.0%	45 90.0%
51-100	13(6) 13.0%	7(5) 7.0%	80(39) 80.0%	5(3) 5.0%	8(5) 8.0%	87(42) 87.0%
101-150	17(4) 11.3%	11(4) 7.3%	122(42) 81.4%	7(2) 4.7%	13(5) 8.7%	130(43) 86.6%
151-200	24(7) 12.0%	15(4) 7.5%	161(39) 80.5%	11(4) 5.5%	18(5) 9.0%	171(41) 85.5%
	19.5%		80.5%	14.5%		85.5%

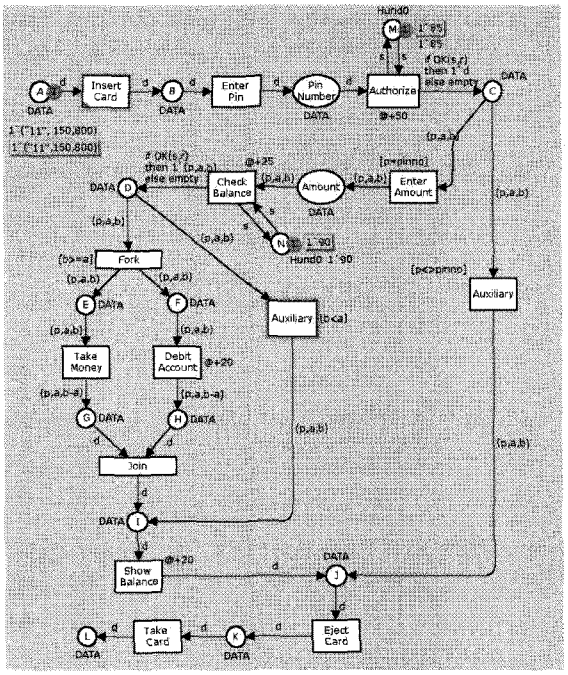


그림 3. 비기능적 요구사항(PoF)을 포함한 CPN 모델
Fig. 3. A CPN model including Non-Functional Requirements

단한 현금자동지급기(ATM) 응용 프로그램에 적용함으로써 그 타당성을 검증한다. 선행 작업으로 우리는 UML 액티비티 모델을 Colored Petri Net (CPN)으로 변환하는 매핑룰과 알고리즘을 개발하였다.

보안(security)은 접근제어(access-control), 안전(safety), 보호(protection) 등의 요소로 표현할 수 있는데, 실패 확률(Probability of Failure, PoF)은 시스템이 외부로부터 얼마나 보호될 수 있는지를 표현하는 측정 지표로 많이 활용되고 있다[14].

ATM의 기능을 모델링한 그림 2의 UML 액티비티 모델에서는 Authorize와 CheckBalance 모듈의 실패 확률(Probability of Failure, PoF)을 각각 15%와 10%로 가정한 경우를 보여주고 있다. 즉, 각각의 모듈을 성공적으로 수행할 확률은 각각 85%와 90%이다. 비기능적 요구사항인 실패 확률을 시뮬레이션하고 그 결과를 분석하기 위해서, 우리는 먼저 UML 액티비티 모델을 매핑룰과 변환 알고리즘을 기반으로 그림 3처럼 CPN 모델로 변환하였다.

CheckBalance의 실패 확률을 10%로 가정하였을 경우, Authorize의 실패 확률을 20%, 15%, 10%, 5%(성공 확률은 80%, 85%, 90%, 95%) 로 변경하면서 200번의 현금 인출을 시도하였으며, 그 결과는 표 1과 같다.

표 1의 각란에 기록된 값은 결과의 누적 횟수

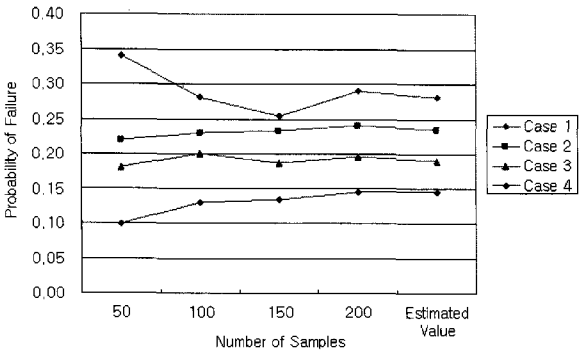


그림 4. 샘플 수에 따른 측정값

Fig. 4. Actual value depending on the number of samples

를 나타내며 괄호안의 값은 시도한 50회 중에서 해당하는 횟수만을 나타낸다. 예를 들면, Case 1, 샘플 51-100 경우 (Authorize 및 CheckBalance의 PoF가 각각 0.2, 0.1) 50회의 시뮬레이션 중에서 Authorize가 실패한 횟수가 8회, 누적횟수가 22회, CheckBalance가 실패한 횟수가 3회, 누적횟수가 6회, 두 모듈 다 실패없이 성공적으로 수행한 횟수가 39회, 누적횟수가 72회이다.

Case 1의 경우 Authorize와 CheckBalance의 PoF가 각각 0.2, 0.1 이므로, 두 모듈이 동시에 성공적으로 수행될 확률은 0.8 * 0.9 = 0.72이고, 따사서 두 모듈이 동시에 실행을 실패할 확률인 PoF의 추정값은 1 - 0.72 = 0.28이 된다. 그림 4와 그림 5에서는 Case별로 PoF의 추정 값과 실제 측정값을 보여주고 있는데, 특히 그림 4에서는 샘플의 수가 많아질수록 측정값이 추정 값에 근접함을 알 수 있다.

시뮬레이션 결과를 분석하여 제안한 프로세스의 타당성을 검증하기 위해 본 실험에서는 퍼센

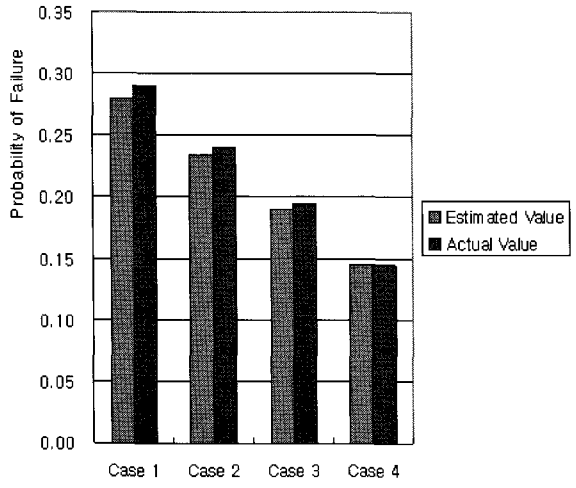


그림 5. Case별 추정값과 측정값의 비교

Fig. 5. Comparison between Estimated and actual value per case

표 2. 퍼센티지 에러

Table 2. The Percentage Errors

	Case 1	Case 2	Case 3	Case 4
Estimated Value	0.280	0.235	0.190	0.145
Actual Value	0.290	0.240	0.195	0.145
Percentage Error	3.448%	2.083%	2.564%	0%

티지 에러(Percentage Error, PE)를 이용한다. PE는 실제적으로 측정한 값이 이론적으로 추정한 값과 어느 정도 차이가 나는지를 나타내는 측정지표가 되는데, 식 (1)과 같이 계산된다.

$$Percentage\ Error(\%) = \frac{|\text{추정값} - \text{측정값}|}{\text{측정값}} * 100 \quad (1)$$

식 (1)에서 알 수 있듯이, PE는 추정값과 실제적으로 측정된 값과의 차이가 적을수록 감소한다. 본 시뮬레이션 결과 표 2에서처럼 네 가지 경우 모두 PE < 3.5% 임을 알 수 있다.

IV. 결 론

본 논문에서는 성능, 보안성, 유용성 등과 같은 비기능적 요구사항이 포함된 소프트웨어 개발을 지원하는 개발 프로세스를 제안하였다. 이 프로세스는 6단계로 구성되어 있으며, 각 단계에서 수행되어야 하는 세부적인 활동들이 정의되어 있다. 특정 비기능적 요구사항을 지원하기 위해서는 요구사항의 특성에 따라 활동들이 카스터마이징 될 수 있다. 제안한 소프트웨어 프로세스의 타당성을 검증하기 위하여 보안성의 척도가 될 수 있는 실패 확률과 같은 비기능적 요구사항을 포함한 모델을 분석하였다. 즉, UML모델을 CPN 모델로 변환하고, 그 모델을 대상으로 시뮬레이션과 결과 분석의 방법을 설명함으로써, 프로세스의 5, 6 단계의 타당성을 보여 주었다.

향후 계획으로는 제안된 프로세스의 1단계부터 4단계까지의 활동들에 대한 타당성 검증을 준비하고 있으며, CPN 모델뿐만 아니라 다른 다양한 정형명세 기법으로 표현된 모델에 대한 적용도 고려하고 있다. 또한 성능이나 유용성과 관련된 비기능적 요구사항을 포함하는 모델에 대해서도 제안된 프로세스를 적용해 볼 계획을 갖고 있다.

[참 고 문 헌]

[1] I. Sommerville and G. Kotonya, *Requirements Engineering: Processes and Techniques*, John Wiley and Sons Ltd, 1998.

[2] L. A. Macaulay, *Requirements Engineering*, Springer-Verlag, 1996.

[3] P. Loucopoulos and V. Karakostas, *System Requirements Engineering*, McGraw-Hill, 1995.

[4] B. W. Boehm, "A spiral model of software development and enhancement," in *IEEE Computer*, vol. 21, no. 5, pp. 61-72, 1988.

[5] K. Sousa, E. Furtado, and H. Mendonca, "UPi- a software development process aiming at usability, productivity and integration," *Proceedings of the 2005 Latin American Conference on Human Computer Interaction*, vol. 124, pp. 76-87, Oct. 2005.

[6] L. L. Constantine and L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley, 1999.

[7] S. Rottger and S. Zschaler, "A software development process supporting non-functional properties," *Proceedings of the IASTED International Conference on Software Engineering*, Feb. 2004.

[8] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

[9] V. Sivess, "Non-functional requirements in the software development process," *Software Quality Journal*, vol. 5, no. 4, pp. 285-294, Dec. 1996.

[10] C. A. Petri, "Kommunikation mit automaten," *Schriften des Institutes fur Instrumentelle Mathematik*, Bonn, West Germany, 1962.

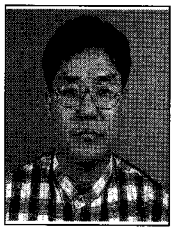
[11] T. Bolognesi and E. Brinksma, "Introduction to the ISO specification language LOTOS," *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 25-59, 1987.

[12] J. M. Spivey, *The Z notation: A Reference Manual, Second Edition*, Prentice Hall, 1992.

[13] C. A. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[14] OMG, *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms: OMG Available Specification*, OMG document number formal/06-05-02, May 2006.

Biography



정 효 택

1986년 경북대학교 전자공학과 졸업
1997년 연세대학교 컴퓨터과학과 (공학석사)
2008년 The University of Texas at Dallas Computer Science (S/W공학박사)
1988년~현재 한국전자통신연구원

<관심분야> Software Engineering, Requirement Engineering, Software Architecture, Software Quality

<e-mail> htjung@etri.re.kr

주 상 현

1989년 동국대학교 전자공학과 졸업
1994년 동국대학교 전자공학과 졸업 (공학석사)
1999년 니이가타대학 자연과학연구과 졸업(공학박사)
2001년-현재 한국전자통신연구원

<관심분야> Virtual Reality, Virtual World, User Interface, MPEG CoDec

<e-mail> joos@etri.re.kr