韓國IT서비스學會誌
第11卷 第2號
2012年 6月

시스템 요구사항 최적화를 위한 프레임워크

김철진* · 송치양** · 이숙희***

A Framework for Optimizing System Requirements

Chul-Jin Kim** · Chee-Yang Song** · Sook-Hee Lee***

■ Abstract ■

A well organized system requirements provide a solid basis of achiving a succesful project and are effective methods of communication among stakeholders. We illustrate and propose a 'System Requirement Development Framework' to develop correct requirements as a methodical approach. We first organize a 'Requirement Development Process' and then establish a 'Non-Functional Requirements Principle'. On these basis, we next propose a 'Requirement Development Guideline' from a perspective of 'Functional Requirement', 'Non-Functional Requirement' and 'Triage' in advanced stages. We also verify and evaluate the suitability of our proposed 'System Requirement Development Framework' by applying it to several projects.

Keyword : System Requirement, System Requirement Engineering, Non-Functional Requirement, Triage

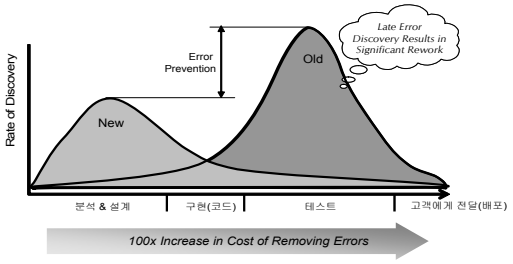
논문투고일 : 2012년 01월 20일 논문수정완료일 : 2012년 05월 20일 논문게재확정일 : 2012년 05월 24일

* 인하공전 컴퓨터시스템과, 제1저자
** 경북대학교 컴퓨터정보학부
*** 서경대학교 컴퓨터과학과, 교신저자

1. 서 론

시스템의 품질은 이해 관계자에 대한 요구사항의 만족도로서 계량화되어 평가될 수 있다. 이해 관계자로부터 정확하고 완전한 시스템 요구사항을 추출하고 이를 명세하는 활동은 시스템 개발의 목적 및 방향을 구체화하는 단계이므로 매우 중요하다. 따라서 개발초기에 요구사항을 명확히 정의하고 아키텍처, 시스템 구조와 동작에 반영하여야 한다. 이를 통해 개발의 효율성, 효과성을 높일 수 있으며 특히, 개발 후반의 재작업을 방지할 수 있다.

요구사항의 품질은 시스템 전체의 품질을 좌우할 만큼 대단히 중요하다. [그림 1]은 SDLC (Software Development Life Cycle) 상에서 결함을 조기에 발견할수록 비용이 감소함을 보여준다. 이는 개발 초기(요구분석 및 분석/설계 단계)에 명확하고 완전한 시스템 요구사항 개발의 중요성을 말해준다[3].



[그림 1] 분석/설계 품질에 의한 비용 절감 효과[3]

체계적인 시스템 요구사항 개발의 정립은 프로젝트 성공을 위한 기반 구축, 이해 관계자와의 효과적인 의사소통의 수단 제공, 요구사항의 완전성과 정확성 유지의 측면에서 중요하다. 첫째, 요구사항은 사용자 관점에서의 품질의 척도이므로 프로젝트의 성공과 실패를 가늠하는 척도이기에 제대로된 요구사항 개발이 필요하다. 둘째, 이해관계자로부터 명확한 요구사항을 도출하기 위한 수단을 제공할 수 있다. 셋째, 표준화된 요구사항 개발 절차와 세부 활동의 제공을 통해 명확한 요구사항을 개발할 수 있다.

시스템 아키텍처의 오류에 의해 시스템이 실패하는 이유는 비기능적 요구사항에 기인한다. 비기능적 요구사항은 대상 시스템이 지원해야 할 기능적 요구사항의 속성 또는 품질 요구사항을 말하며, 소프트웨어의 품질 속성으로 반영된다[2]. 비기능적 요구사항은 본질적으로 모호성과 비가시성을 가지고 있다. 본 논문에서는 기존의 연구들이 이러한 문제를 해결하기 위해 방안이 미흡하기 때문에, 체계화된 시스템 요구사항 개발 프레임워크를 제안한다.

시스템 요구사항 개발의 최종 목표는 완전한 요구사항 명세서가 아니라 명확하게 표현되는 요구사항을 개발하는 것에 맞춰져야 한다[1]. 이를 위해 본 논문의 제 2장에서는 비기능적 요구사항과 관련된 9개 국제 표준을 조사하고 제 3장에서 비기능적 요구사항에 대한 정의 원칙을 도출하여 개발 생명주기의 요구분석 및 분석/설계 단계에서 시스템 요구사항을 체계적으로 개발할 수 있도록 시스템 요구사항 개발 프레임워크를 제시한다. 제 4장에서는 제시한 시스템 요구사항 개발 프레임워크를 바탕으로 주요 도메인의 시스템 개발 프로젝트에 적용하여 실험 및 분석한다. 마지막으로 제 5장에서 결론을 맺음으로써 본 논문을 마무리한다.

2. 관련 연구

비기능적 요구사항의 모호성과 비가시성을 해결하기 위해 비기능적 요구사항과 관련된 9개의 국제 표준을 조사한다. 해당 표준들 간의 공통/유사 품질 기준을 수집하여 이를 바탕으로 제 3.1절에서 비기능적 요구사항 정의 원칙을 도출한다.

본 논문에서 조사한 요구사항 관련 국제 표준은 <표 1>과 같다.

2.1 INCOSE

INCOSE(International Council on System

〈표 1〉 비기능적 요구사항 관련 국제 표준

국제표준80	주체	주요 영역(분야)	설명
INCOSE	INCOSE	System Engineering	성공적인 시스템 실현을 위한 학제간 원칙 및 실행지침을 개발 전파하는 비영리 단체
ISO 9126 (Quality Attribute)	ISO/IEC	Software Engineering	SW 품질 측정을 위해 SW 품질 요소와 특성을 정의하고 객관적으로 정량화한 품질 모형 표준
IEEE 1233 (System Requirement)	IEEE	System Requirement Specification	요구사항 집합과 시스템 요구사항 명세서의 개발 지침서
System Eng. (Richard)	Richard	System Engineering	시스템 엔지니어링 관련 표준 제공
SoftEng.polito.it	SoftEng	Software Development Technique	진보된 소프트웨어 개발 기술의 개발 및 실험
Software Eng. (Otto P.)	Otto P.	Software Engineering	소프트웨어 엔지니어링 관련 표준 제공
IEEE 12207.1 (SW Requirement)	IEEE/EIA	ISO/IEC 12207 확장	IEEE 12207.0의 데이터 목적을 확장하기 위한 권고사항을 제공하는 지침 문서
IEEE 12207.1 (System Requirement)	IEEE/EIA	ISO/IEC 12207 확장	IEEE 12207 프로세스를 구현하는데 필요한 권고사항을 제공하는 지침 문서
MIL-STD-498 (System Requirement)	미국방성	소프트웨어 개발 및 문서화	미국방성의 소프트웨어 개발 표준

Engineering, 국제시스템 공학회)는 시스템 개발에 있어서 복잡성과 불확실성에 대해 시스템 공학적인 접근 및 해법을 연구하는 1990년에 설립된 비영리 단체이다. INCOSE에서 발간한 'Metrics Guidebook', 'Technical Measurement' 등을 바탕으로 추출한 비기능적 요구사항은 <표 2>와 같다. 특히, INCOSE에서는 시스템 설계 및 트레이드 오프와 관련된 비기능적 요구사항을 추출한다.

2.2 ISO 9126

ISO 9126은 소프트웨어 품질을 측정하기 위해 소프트웨어 품질 요소와 특성을 정의하고 이를 객관적으로 정량화 할 수 있도록 한 품질 모형 표준이다. ISO 9126은 4개의 세부항목으로 구성되어 있다[6].

ISO 9126-1(Quality Model : 6개의 품질특성, 21개의 부특성)

ISO 9126-2(External Metrics : SW 완성단계의 측정으로 Executable Code, Test Case Run을 포함)

ISO 9126-3(Internal Metrics : SW 개발단계의 측정으로 Source Code, 분석 Document, Design Spec을 포함)

ISO 9126-4(Quality in Use Metrics : 효율성, 생산성, 안정성, 만족성의 규정 목표)

2.3 IEEE 1233(System Requirement)

IEEE 1233은 IEEE에서 정의한 요구사항 집합과 시스템 요구사항 명세서(SyRS)의 개발을 위한 지침서이다[4]. IEEE 1233의 요구사항 명세 컴포넌트 중 'Requirement Type'은 비기능적 요구사항으로 추출할 수 있다.

2.4 System Eng.(Richard)

Richard Thayer는 IEEE Computer Society와 IEEE Software Engineering Standards Subcommittee의 시니어 멤버로 활동하였다. 그는 소프트웨어 프로젝트 관리(Software Project Management), 소프트웨어 공학(Software Engineering), 소프트웨어 공학 표준(Software Engineering Standard)와

관련하여 40여 개의 연구를 진행하였다. 특히, 소프트웨어 요구공학에서는 소프트웨어 시스템 및 외부 주변 환경(하드웨어, 사람 등)을 포함하는 요구공학의 베스트 사례를 제시하였다[10]. 시스템 요구공학에서 추출한 주요 비기능적 요구사항은 <표 2>와 같다.

〈표 2〉 System Eng.(Richard)의 비기능적 요구사항

비기능적 요구사항		
Functional	Performance	External Interface
Design Constrains	Correctness	Portability

2.5 SoftEng.polito.it

SoftEng.polito.it는 전달 시간(Delivery time), 신뢰성(Reliability), 비용(Cost), 변경의 용이(Ease of Change) 등의 품질 속성을 향상시키고, 연속적이고 반복적인 결과를 제공할 수 있는 산업 전반에 활용될 수 있는 진보된 소프트웨어 개발 기술을 개발하고 실험하는 것을 미션으로 삼고 있는 단체이다. SoftEng.polito.it에서 발표한 비기능적 요구사항[9]을 참조하여 <표 3>의 비기능적 요구사항을 도출한다.

〈표 3〉 SoftEng.polito.it 추출 비기능적 요구사항

분류	비기능적 요구사항	분류	비기능적 요구사항
Product	Usability	Organizational	Delivery
	Efficiency		Implementation
	Reliability		Standards
	Protability		Interoperability
			Ethical

2.6 Software Eng.(Otto P.)

[8]은 컴포넌트 기반(CBSE)의 소프트웨어 시스템의 품질 속성(Quality Attirbute)을 분류하고 품질 속성과 사용 사례 실제화(Use Case Realization)의 관계를 설명하고, 기능적 및 비기능적 요구사항의 디자인을 지원하는 소프트웨어 공학 프로세스를 제안하였다[8]. Software Eng.(Otto P.)에서 제시한 품질 속성에서 추출한 주요 비기능적 요구사항은 <표 4>과 같다.

〈표 4〉 Software Eng.(Otto P.) 비기능적 요구사항

분류	비기능적 요구사항	분류	비기능적 요구사항
Not observable at runtime (over product Life-cycle)	Testability	Observable at runtime	Useability
	Portability		Dependability
	Integrability		Secuirty
	Maintainability		Safety
	Reuseability		Performance
	Deployability		

2.7 IEEE 12207.1(SW Requirement)

IEEE 12207은 ISO/IEC 12207의 본문과 부록을 그대로 수정없이 적용하고, MIL-STD-498과 MIL-STD-498의 데이터 항목 기술서(Data Item Descriptions) 및 IEEE 소프트웨어 공학 표준으로부터 유도한 세부적인 지침을 추가로 포함한다[9]. IEEE 12207.1의 소프트웨어 관점에서 추출한 비기능적 요구사항은 <표 5>와 같다.

〈표 5〉 IEEE 12207.1의 비기능적 요구사항

분류	비기능적 요구사항	분류	비기능적 요구사항
Functionality	Function	기타	Qualification
	Performance requirements		Safety
	Physical characteristics		

2.8 IEEE 12207.1(System Requirement)

IEEE 12207.1은 특정한 문서 혹은 전자 데이터가 특정한 프로세스에 어떻게 연관되는지에 대한 지침을 원하는 사용자에게 문서의 내용에 대한 다양한 권고사항을 제공한다. IEEE 12207의 시스템

관점에서 추출한 비기능적 요구사항은 <표 6>과 같다.

〈표 6〉 IEEE 12207.1의 비기능적 요구사항

비기능적 요구사항		
Required States	Business requirements	User requirements
Required modes	Organizational requirements	

2.9 MIL-STD-498(System Requirement)

MIL-STD-498(Millitary-Standard-498)은 미국의 군사 표준규격이었으며, “소프트웨어 개발과 문서화 요건을 수립”하기 위한 목적으로 만들어졌다. 이 표준 규격은 1994년 11월 8일 공표되었으며, 기존의 표준 규격인 DOD-STD-2167A, DOD-STD-7935A, DOD-STD-1703을 대체하여 적용되었다. 이 표준 규격은 1998년 5월 27일 J-STD-016과 IEEE 12207로 대체되어 효력을 상실했다[7]. MIL-STD-498(System Requirement)에서 추출한 비기능적 요구사항은 <표 7>과 같다.

〈표 7〉 MID-STD-498의 비기능적 요구사항

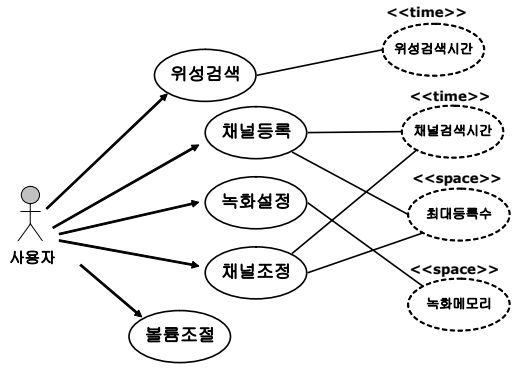
분류	비기능적 요구사항	설명
기능성	체계능력 요구사항	시스템 기능의 체계적 제공을 위한 요구사항
인터페이스	외부 인터페이스 요구사항	시스템 관점에서 외부 시스템과의 통합을 위한 인터페이스 요구사항
	내부 인터페이스 요구사항	시스템 내의 컴포넌트 간의 인터페이스 요구사항

2.10 목적-객체 패턴 접근법[9]

패턴 기반의 접근법(Pattern-based approach)은 경험과 사례를 이용하여 요구사항을 추출할 수 있다[5]. 그러나 대부분의 기능을 추출하는데 한계를 가지고 있으므로, 목적-객체 패턴 접근법(Goal-object pattern approach)에서는 UML과 목적-지향 기법(Goal-oriented method)을 이용하여 기능적/비기능적 요구사항을 도출한다. 이 기법에서는 점증적으로 작은 단위의 패턴들을 축적하여 큰 단위의 패턴을 구성하는 방식으로 요구사항을 추출한다.

2.11 비기능 요구사항 추출 방안[2]

[2]연구에서는 기능 중심의 사용사례(Use Case)를 중심으로 비기능적인 사용사례를 추출한다. [그림 2]에서와 같이 기능 사용사례에 대한 비기능 사용사례를 점선으로 된 타원에 연결하여 비기능 요구사항을 추출한다.



[그림 2] 기능/비기능 사용사례의 추출

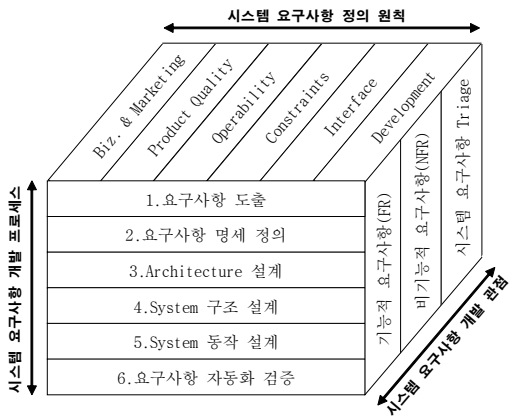
[2]연구에서는 비기능 요구사항을 추출하기 위한 구체적인 기법 및 가이드 라인이 미흡하다.

3. 시스템 요구사항 개발 프레임워크

시스템 요구사항 개발을 위한 요구사항 정의 원칙을 수립하기 위해 제 2장에서 관련된 9개의 국제 표준을 조사하였다. 본 절에서는 이를 바탕으로 새로운 시스템 요구사항 개발 방법론을 제안한다.

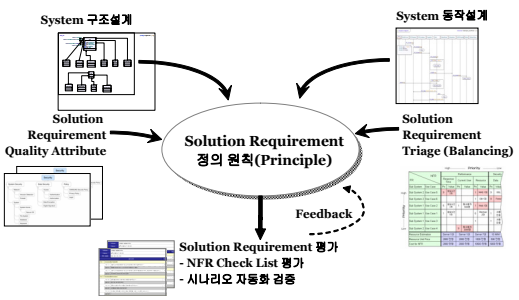
시스템 요구사항 개발 프레임워크는 시스템 요구사항 정의 원칙, 시스템 요구사항 개발 프로세스, 시스템 요구사항 개발 관점으로 구성된다. 각 구성요소는 세부적으로 복수 개의 하부 구성요소

로 나눌 수 있으며 이를 도식화하여 나타내면 [그림 3]과 같다.



[그림 3] 시스템 요구사항 개발 프레임워크 : CUBE 모델

[그림 3]의 시스템 요구사항 개발 프레임워크 규브(CUBE) 모델을 요구사항 개발 프로세스 간의 관계를 기반으로 새로 도식화한 시스템 요구사항 개발 프레임워크의 상호작용(INTERACTIVE) 모델은 [그림 4]와 같다. 즉, 시스템 요구사항(System Requirement) 정의 원칙을 기반으로 시스템 요구사항 품질속성, 시스템 구조설계, 시스템 동작설계, 시스템 요구사항 트리아제(Triage)를 개발한 후, 비기능적 요구사항에 대해서는 'NFR Check List'를 통해 평가하고, 기능적 요구사항에 대해서는 시나리오 자동화 검증을 통해 평가한다.



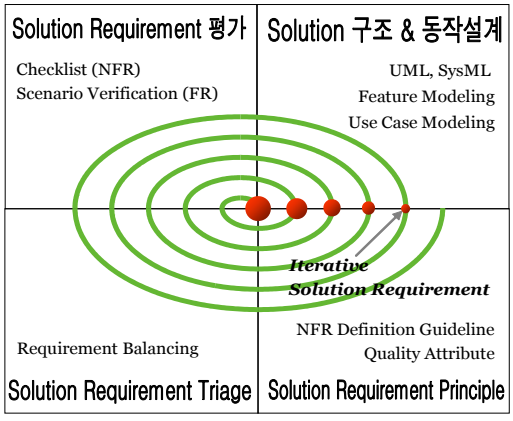
[그림 4] 시스템 요구사항 개발 프레임워크 : INTERACTIVE 모델

시스템 요구사항 개발 프레임워크의 구성요소인 시스템 요구사항 개발 프로세스, 시스템 요구사항 정의 원칙, 시스템 요구사항 개발 관점에 대한 세부내용은 다음과 같다.

3.1 시스템 요구 사항 개발 프로세스

시스템 요구사항 개발 프레임워크의 또 다른 축인 시스템 요구사항 개발 프로세스는 SDLC 표준이 ISO/IEC 12207의 개발 프로세스를 기반으로 하여 나선형/반복(Spiral/Iterative) 모델과 UP(Unified Process) 방법론을 수정하였고 궁극적으로는 시스템 요구사항을 효과적으로 개발 및 적용할 수 있도록 한다.

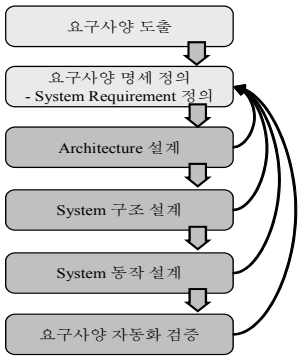
첫째, 나선형/반복 모델의 반복 기법을 통해 조기 요구사항의 개발 완성도를 높여 위험을 최소화하도록 반복적인[원칙 → 트리아제 → 평가 → 설계]의 과정을 거치도록 수정한다. 반복적 시스템 요구사항 기반의 시스템 요구사항 개발 프로세스는 [그림 5]와 같다.



[그림 5] Iterative System Requirement 기반의 시스템 요구사항 개발 최적화 프로세스

둘째, UP의 명세화, 문서화, 가시화, 구조화 기법을 수정하여 [그림 6]과 같이 적용한다. 시스템 요구사항 개발 프로세스를 요구사항 도출, 요구사항 명세 정의, 아키텍쳐 설계, 시스템 구조 설계,

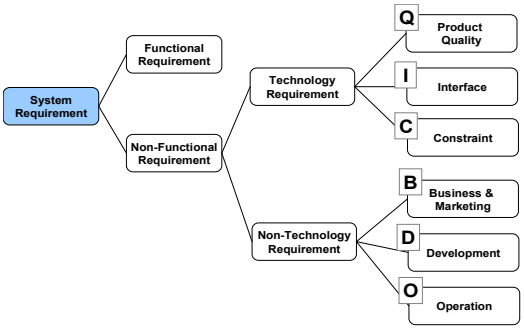
시스템 동작 설계, 요구사항 자동화 검증 단계로 구분하여 정형화한다.



[그림 6] RUP 방법론의 Tailoring을 통한 시스템 요구사항 개발 프로세스

3.2 시스템 요구 사항 정의 원칙

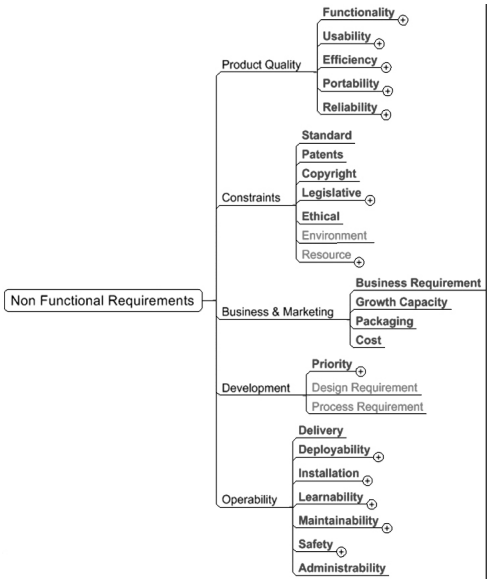
제 2장에서 조사한 9개 국제 표준에서 161개의 비기능적 요구사항을 추출한다. 추출한 비기능적 요구사항을 다시 39개로 필터링하여 이를 6개의 서브 그룹으로 카테고리화 한다. 카테고리화는 기술 관점과 비기술 관점으로 분류하고 이를 다시 세분화한다. 즉, 추출된 비기능적 요구사항은 [그림 7]과 같이 'Product Quality', 'External', 'Constraint', 'Business and Marketing', 'Development', 'Operation'으로 분류한다.



[그림 7] 비기능적 요구사항의 서브그룹

추출된 비기능적 요구사항의 전체 내역은 [그림 8]과 같으며, 이를 기준점으로 하여 시스템 요구사항 정의 원칙을 수립할 수 있다. [그림 8]의 각각의 비기능적 요구사항에 대한 분류는 <표 8>, <표 9>와 같다. <표 8>과 <표 9>에서는 'Business and Marketing' 카테고리에 대한 상세항목은 제외하였으나 실무적 적용을 위해서는 'Business and Marketing'에 대해서는 반드시 고려하여야 할 것이다.

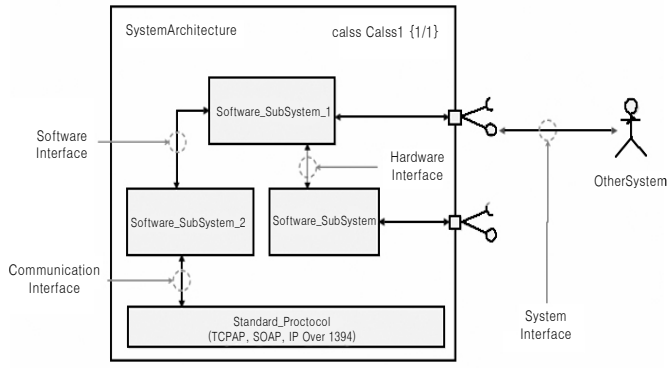
각각의 비기능적 요구사항에 대한 정의 원칙은 해당 비기능적 요구사항(세부적 분류가 가능하면 레벨 2 비기능 요구사항, 불가능하다면 레벨 1 비기능 요구사항)별로 '정의', '조건', '입력', '도구', '사례'로 구성하여 명확히 기술한다. 이것은 비기능적 요구사항에 대해 해당 조직, 프로젝트에 맞추어 수정하여 기술한다. <표 10>은 제품 품질(Product Quality) 카테고리의 기능성(Functionality) 서브그룹의 적합성(Suitability)에 대한 시스템 요구사항 정의 원칙의 사례이다.



[그림 8] 도출한 비기능적 요구사항

지금까지 설명한 시스템 요구사항 정의 원칙의 수립 절차는 다음과 같다. 먼저 9개의 국제 표준를 조사하여 161개의 비기능적 요구사항을 추출한다.

〈표 8〉 카테고리화된 비기능적 요구사항 전체 내역(1/2)

카테고리	Level 1 NFR	Level 2 NFR(세부적 분류 가능 시)
Product Quality	Functionality(기능성)	Suitability(적합성), Accuracy(정확성), Interoperability(상호운용성), Security(보안)
	Usability(사용성)	Understandability(이해성), Attractiveness(친밀성)
	Efficiency(효율성)	Performance(성능), Resource Utilization(자원 활용율), Time Behavior
	Protability	Adaptability, Mobility, Nomadicity, Replaceability, Co-existence
	Reliability(신뢰성)	Maturity(성숙도), fault Tolerance(오류 허용성), Recoverability(회복성)
Constraints	Patents(특허)	세부적 분류 없음
	Standard(표준)	세부적 분류 없음
	Copyright(저작권)	세부적 분류 없음
	Resource(자원)	Hardware, Software, Communication
Interface	System Interface	
	Hardware Interface	
	Software Interface	
	Communication Interface	

〈표 9〉 카테고리화된 비기능적 요구사항 전체 내역(2/2)

카테고리	Level 1 NFR	Level 2 NFR(세부적 분류 가능 시)
Operability	Maintainability	Analyzability(분석 가능성), Changeability(변경성), Stability(안정성), Testability(시험성)
	Installation	Installation Process, Installation Time
	Deployability	Configurability, Distributeability
	Delivery	세부적 분류 없음
	Learnability	세부적 분류 없음
	Safety	Environmental Influence, Operation and Maintenance, Operation, Personnel Injury
	Administrability	세부적 분류 없음
Development	Priority	세부적 분류 없음
	Design Requirement	세부적 분류 없음
	Process Requirement	세부적 분류 없음
Business and Marketting	본 논문에서는 제외하였으나 실무에 적용 시 최우선적으로 반드시 고려하여야 함	

〈표 10〉 Product Quality 〉 Functionality 〉 Suitability에 대한 시스템 요구사항 정의 원칙

Level 1 NFR	Level 2 NFR	Component	Description	Related Artifacts	Master
Functionality (기능성)	Suitability (적합성)	정의	사용자 요구사항과 Functional Requirement와의 적합성(일관성)을 유지하기 위한 활동		품질 보증 담당자
		조건	Suitability ≒ 100%		
		입력	User Requirement, Functional Requirement, Testcase	User Requirement Spec. SW Requirement Spec.	
		도구	3점 정합 체계		
		사례	User Requirement, Functional Requirement, Testcase의 일관성을 유지하기 위해 3점 정합 체계 이용		

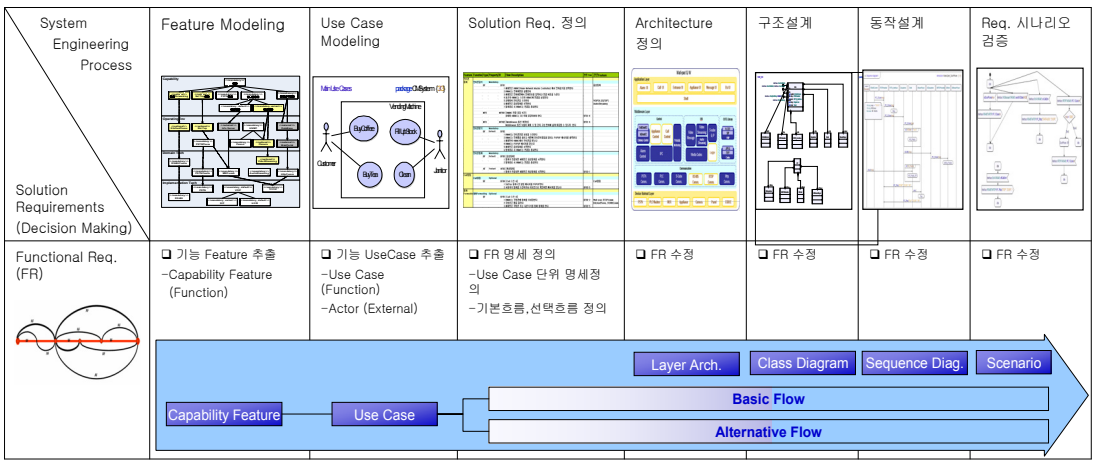
추출된 비기능적 요구사항을 그룹핑하여 6개로 카테고라이즈 한다. 카테고라이즈된 비기능적 요구사항을 39개로 압축하여 요구사항 정의 원칙을 개발한다. 이제 잘 작성된 비기능적 요구사항 정의 원칙을 기본 Baseline으로 하여 해당 시스템의 요구사항 간 트리아제기법 개발에 사용할 수 있다.

트리아제 기법(요구사항간의 균형) 개발은 비기능적 요구사항 정의 원칙의 활용 방법으로서 제 3.3절의 시스템 요구사항 개발 관점에서 설명한다.

3.3 시스템 요구 사항 개발 관점

시스템 요구사항 개발 프레임워크의 마지막 축인 시스템 요구사항 개발 관점은 기능적 요구사항 개발, 비기능적 요구사항 개발, 트리아제 기법(요구사항간의 균형) 개발로 구성된다.

첫째, 기능적 요구사항 개발은 제 3.1절의 시스템 요구사항 개발에 최적화된 시스템 요구사항 개발 프로세스의 각 단계별로 [그림 9]와 같은 활동을 수행한다. 기능적 요구사항 개발은 기능 피처(Feature)을 추출한 후, 해당 피처에 대한 복수개의 사용 사례(Use Case)를 도출한다. 각 사용 사례별로 기본 흐름(Basic Flow)과 대안 흐름(Alternative Flow)이 모두 정의된 기능적 요구사항(FR, Functional Requirement)을 명세한다. 기능적 요구사항 명세는 점진적/반복적으로 상세화한다. 아키텍쳐 정의

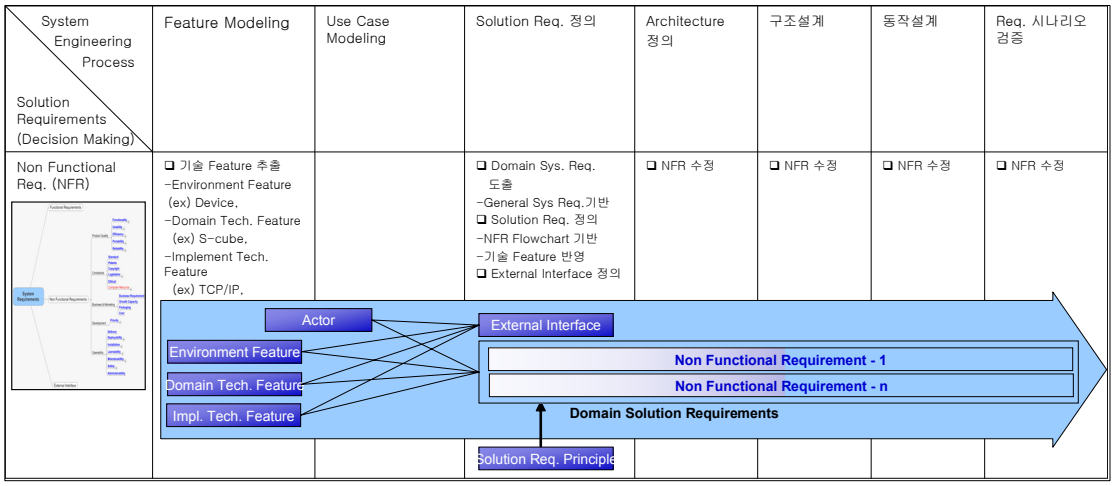


[그림 9] 시스템 요구사항 개발 관점 : 기능적 요구사항 개발의 주요 활동

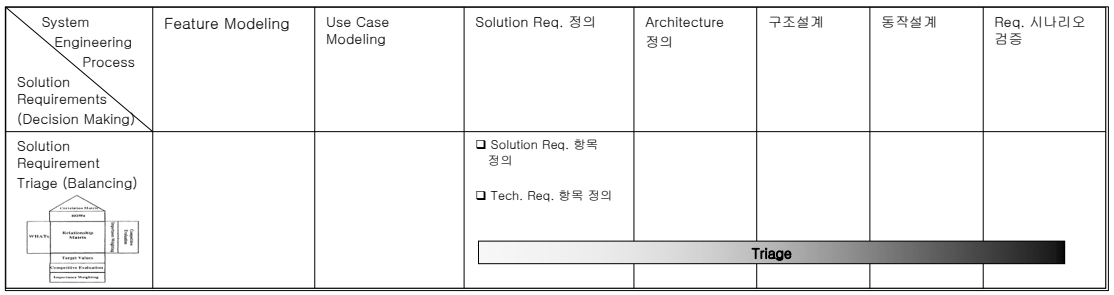
단계에서는 기능적 요구사항의 계층 아키텍처, 구조 설계에서는 클래스 다이어그램(Class Diagram), 동작 설계에서는 순차도(Sequence Diagram), 그리고 기능 검증을 위한 시나리오(Scenario)를 작성한다.

둘째, 비기능적 요구사항 개발은 제 3.1절의 시스템 요구사항 개발에 최적화된 시스템 요구사항 개발 프로세스의 각 단계별로 [그림 10]과 같은 활동을 수행한다. 피쳐 모델링(Feature Modeling) 단계에서는 환경(Environment), 도메인 기술(Domain Technique)과 구현 기술(Implementation Technique) 측면으로 세분화하여 기술 피쳐를 추출한다. 추출 완료된 피쳐들을 고려하여 액터(Actor)가 연계된 비기능적 요구사항을 다음과 같이 정의한다. 먼저 추출한 도메인 기술 피쳐에 대해 '시스템 요구사항 정의 원칙'에 기반하여 '도메인 시스템 요구사항'을 도출한다. 도출된 '도메인 시스템 요구사항'을 기준으로 환경, 기술, 구현 피쳐들을 반영하여 비기능적 요구사항을 'NFR Flowchart' 도구 등을 활용하여 명세한다. 추가로 비기능적 요구사항에는 외부 시스템 간에 신뢰성, 유연한 상호운용성, 유지보수성 향상을 위해 반드시 외부 인터페이스(External Interface)에 대한 정의를 포함하여야 한다. 비기능적 요구사항에 대해서도 기능적 요구사항과 마찬가지로 점진적, 반복적으로 상세화 한다.

셋째, 트라아제 기법 개발은 제 3.1절의 시스템 요구사항 개발에 최적화된 시스템 요구사항 개발 프로세스의 각 단계별로 [그림 11]과 같은 활동을



[그림 10] 시스템 요구사항 개발 관점 : 비기능적 요구사항 개발의 주요 활동



[그림 11] 시스템 요구사항 개발 관점 : 트리아제기법의 주요 활동

수행한다. 정의된 요구사항을 모두 만족하는 시스템을 개발하려면 비용이 대단히 많이 든다. 주어진 비용과 자원 내에서 가장 최적의 품질을 달성하기 위해서는 각 요구사항간에 균형이 필요하다. 즉, 트라아제 기법을 통해 요구사항 품질를 향상시키고, 비용과 성능을 최적화할 수 있다. 트라아제 기법은 '기능적 및 비기능적 요구사항 정의' 버전 1.0 완료된 시점부터 반복적으로 이루어진다. 트라아제 기법은 다음과 같다. 정의한 요구사항 리스트를 우선 순위화 한 후 기능적 요구사항과 비기능적 요구사항을 2차원으로 맵핑한다. 2차원 매트릭스에서 투입 리소스를 예측하고 해당 비기능적 요구사항의 비용을 산출할 수 있다. 산출된 비용과 우선순위를 고려하여 해당 비기능적 요구사항을 적용할지 여부를 결정할 수 있다. 위에서 설명한 트리아제 기법에 대한 예는 [그림 12]와 같다. 우선순위화한 사용 사례로 명세한 기능적 요구사항과 측정 가능한 비기능적 요구사항을 행과 열로 2차원 구성한다. 각 항목 별로 우선 순위(Priority)와 값(Value)을 설정하고 이를 기반으로 해당 'NFR'(열)의 자원 및 비용을 추정할 수 있다. 추정된 비용을 기준으로 요구사항 간 균형을 통해 최적의 품질을 확보할 수 있다.

4. 실험 및 평가

제 3장에서 제시한 시스템 요구사항 개발 프레임워크를 주요 도메인의 시스템 개발 프로젝트에 적용하여 실험 및 분석을 실시한다. 실험 및 평가를 실시한 대상 도메인은 <표 10>과 같다. 소프트웨어 규모는 50만 LOC(Line Of Code) 이상은 대규모, 10만~50만 LOC는 중규모, 10만 LOC 이하는 소규모로 구분한다.

대상 시스템에 대해 시스템 요구사항 개발 프레임워크의 적용은 다음 절차로 수행한다. 먼저 비

〈표 11〉 실험 및 평가 대상 시스템 내역

시스템 명	도메인	SW 응용 유형	SW 규모	핵심 품질 특성
시스템 A	유무선 통신	통신제어용	중	신뢰성, 성능
시스템 B		노드제어 (인터넷)용	소	
시스템 C		노드제어 (TV)용	소	
시스템 D	방송 통신	시스템용	대	다중 사이트, 분산처리, 성능
시스템 E		멀티미디어용	중	
시스템 F	서비스	시스템용	소	다중 사이트, 분산처리
시스템 G		지능정보용	소	

High — **Priority** — Low

FR \ NFR			Performance											
			Response Time				Current User				Resource			
Sub System	Use Case	ID	Pri	Value	Test -able	Test Method	Pri	Value	Test -able	Test Method	Pri	Value	Test -able	Test Method
Sub System 2	Use Case 5	005	9	응답시간 2초	○	단위테스트					1	WAS 1대	×	
Sub System 2	Use Case 6	006									1	DB 1대	×	
Sub System 1	Use Case 2	002	5	응답시간 2초	○	단위테스트	1	동시용자 500명	○	시스템 테스트	1	Web 1대	×	
Sub System 1	Use Case 1	001	1	응답시간 5초	×						5	Backup 2대	×	
Sub System 1	Use Case 3	003												
Sub System 2	Use Case 4	004					9	동시용자 2000명	○	시스템 테스트				
Resource Estimation			Server 1대				Server 1대				Server 7대			
Resource Unit Price			2000 만원				2000 만원				1500 만원			
Cost for NFR			2000 만원				2000 만원				10500 만원			

High — **Priority** — Low

[그림 12] 트리아제 기법 사례

기능적 요구사항 정의 원칙에 의거하여 시스템 요구사항(비기능적) 평가 리스트(시스템 요구사항 평가 메트릭)를 해당 도메인 영역에 맞추어 작성하고 이를 토대로 <표 11>에서 선정한 7개 시스템을 평가한다. 그리고 계량적으로 평가된 결과를 가지고 문제점을 분석하고 개선사항을 도출한다.

4.1 시스템 요구 사항 평가 메트릭

제 3.1절에서 정의한 시스템 요구사항 정의 원칙에 기반하여 대상 시스템에 대한 시스템 요구사항 평가 메트릭을 작성한다. 시스템 요구사항 정의 원칙은 6개의 카테고리인 'Product Quality', 'Operability', 'Constraints', 'Interface', 'Development', 'Business and Marketing'으로 구성되어 있지만, 여기서는 선정된 시스템의 특성을 고려하여 업무 및 마케팅을 제외한 5개 카테고리의 39개 비기능적 요구사항, 71개의 평가 항목으로 구성한다.

'Business and Marketing'에 대한 평가 메트릭은 시장성과 전략을 포함한 고객접점 및 마케팅 부서와 긴밀히 협력하여 작성하여야 하나 본 논문에서는 현실적으로 불가능하여 평가의 객관성을 유지하기 위해 제외하였다. 그러나 현업에서의 적용 시에는 반드시 'Business and Marketing' 영역을 고려하여 평가 메트릭을 작성하여야 할 것이다.

평가 메트릭의 작성 방법의 예는 <표 12>와 같다. <표 12>는 개발 카테고리의 레벨 1 비기능 요구사항인 'Priority', 'Design Requirement', 'Process Requirement'에 대한 시스템 요구사항 평가 메트릭 작성 방법을 보여준다. 시스템 요구사항 정의 원칙에서 적용할 비기능적 요구사항을 선정한 후, 해당 비기능적 요구사항의 달성정도를 수치화하여 계량화한다.

4.2 분석 및 평가

시스템 요구사항 개발 프레임워크를 적용하기 위해 7개 시스템을 선정하였고, 제 4.1절에서 정의한 시스템 요구사항 평가 메트릭을 기준으로 해당 시스템을 평가하였다. [그림 13]을 보면, 인터페이스 영역은 상대적으로 한계점(Threshold)에 가까워 어느정도 요구사항을 달성하였지만, 'Product Quality'와 'Operability' 영역은 극히 저조하므로 향후 시급히 개선이 필요함을 알 수 있다.

[그림 14]는 기능적 요구사항과 비기능적 요구사항 간의 트리아제를 통해 요구사항을 조정하는 과정을 나타내고 있다.

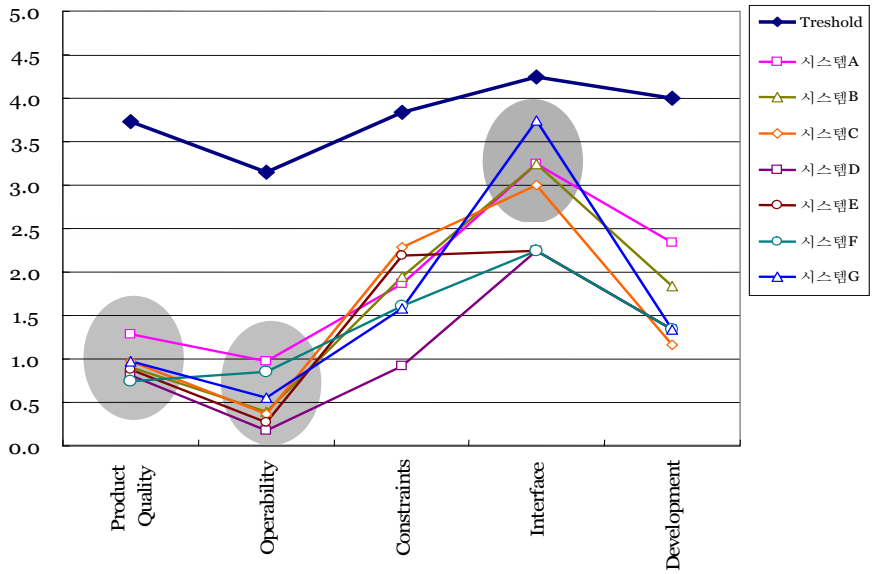
[그림 13]과 [그림 14]의 평가 결과를 분석하여 취약 영역을 찾아서 해당 영역의 개선점을 <표 13>과 같은 분석 결과를 도출할 수 있다.

5. 결 론

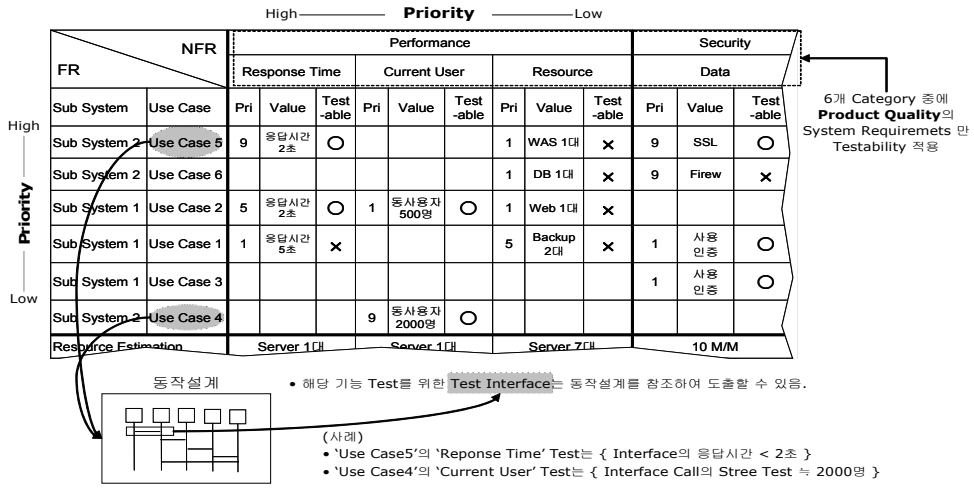
명확하게 표현되는 요구사항을 개발하고 이를 설계 단계에 적용하는 것은 시스템의 품질을 달성하는데 매우 중요한 요소이다. 본 논문에서 제안한 시스템 요구사항 개발 프레임워크를 통해 만족할 만한 수준의 품질을 보장하기 위한 가이드라인을 얻을 수 있으며 현업 적용할 수 있다. 특히, 사용자 요구사항 정의 원칙을 활용한 요구사항간의 균

〈표 12〉 시스템 요구사항 정의 원칙

카테고리	Level 1 NFR	평가항목	달성기준
Development (DV)	DV1.1 Priority	DV1.1-1(Triage) 기능 및 비기능 요구사항에 대한 우선순위화를 하였는가?	우선순위화
	DV2.1 Design Requirement	DV2.1-1 과제 상황에 따른 설계 수준을 정의 하였는가?	과제 수준
		DV2.1-2 설계 시각에 따른 설계 도구를 정의 하였는가?	도구 선정
	DV3.1 Process Requirement	DV3.1-1 개발 프로세스를 정의하였는가?	프로세스



[그림 13] 시스템 요구사항 평가 메트릭 적용 결과 : 전체



[그림 14] 시스템 요구사항 트리아제(요구사항 간 균형)

형 기법인 트리아제를 기능적 요구사항, 비기능적 요구사항의 관점과 병행하여 수행한다면 효과적이고 효율적으로 요구사항을 개발할 수 있다. 또한 요구사항의 테스트성(Testability)를 확보하여 요구사항 검증을 자동화한다면 요구사항 검증에 들어가는 비용을 획기적으로 줄일 수 있을 것이다. 마지막으로 현 소프트웨어 개발의 생산성 및 품질의 향상을 위해 본 논문의 요구사항 개발 방법론이 기반을 제공할 것이다.

참 고 문 헌

[1] 박수용, 황만수, "요구사항 관리(Requirement Management)의 체계적인 접근방법", 『IT Tr-

〈표 13〉 평가 시스템의 분석 결과

시스템 명	현수준	취약 영역	개선 포인트(품질 향상 대안)
시스템 A	3.8	◦ Product Quality : Reliability ◦ Operation : Maintainability, Safety ◦ Constraints : Copyrights	◦ Feature Modeling ◦ 구조 & 동작 설계
시스템 B	1.4	◦ Product Quality : Reliability ◦ Operation : Delivery, Safety ◦ Constraints : Copyright, Resource ◦ Interface : System Interface ◦ Development : Priority	◦ NFR 정의 원칙 ◦ 구조 & 동작 설계
시스템 C	1.4	◦ Product Quality : Functionality, Reliability ◦ Operation : Installation, Deployability, Delivery ◦ Constraints : Copyright ◦ Interface : System Interface ◦ Development : Priority	◦ FR 정의 가이드 ◦ FR 시나리오 검증(자동화) ◦ Feature Modeling
시스템 D	1.1	◦ Product Quality : Usability, Protability ◦ Operation : Maintainability, Deployability, Safety ◦ Constraints : Patents, Copyrights ◦ Interface : System Interface ◦ Development : Priority	◦ NFR 정의 원칙 ◦ 구조 & 동작 설계
시스템 E	1.7	◦ Product Quality : Usability, Reliability ◦ Operation : Installation, Safety ◦ Constraints : Copyrights	◦ Featuring Modeling
시스템 F	1.6	◦ Product Quality : Efficiency, Functionality ◦ Operation : Delivery, Learnability, Safety ◦ Constraints : Copyrights, Resource ◦ Development : Priority	◦ FR 정의 가이드 ◦ FR 시나리오 검증(자동화) ◦ Feature Modeling
시스템 G	1.6	◦ Product Quality : Prototability, Reliability ◦ Operation : Installation, Safety ◦ Constraints : Patent, Copyrights ◦ Development : Priority	◦ NFR정의 원칙 ◦ 구조 & 동작 설계

end」, 2010.

[2] 서광익, 최은만, “사용사례를 이용한 내장형 소프트웨어의 비기능 요구사항 추출방안”, 「한국컴퓨터종합학술대회 2005 논문집」, 제32권, 제1(B)호(2005), pp.385-387.

[3] 최정아, 정기원, “내장형 소프트웨어의 비기능적 요구사항 성능 중심 추적”, 「정보과학회논문지 : 소프트웨어 및 응용」, (2006), pp.615-623.

[4] Boehm, B. W. et al., “Some Experience with Automated Aids to the Design of Large-Scale Reliable Software”, *IEEE Trans. On Software Engineering*, 1975.

[5] Gamma E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison Wesley, 1994.

[6] IEEE Std 1233, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[7] IEEE/EIA 12207 : Software Life Cycle Proceses, Institute of Electrical and Electronics Engineers Electronic Industries Association.

[8] ISO/IEC 9126. Software Engineering-Product Quality, ISO/IEC, 2001.

[9] Lawrence, C. and S. Sam, “Capturing and Reusing Functional and Non-functional Requirements Knowledge : A Goal-Object Pattern Approach”, *IEEE*, (2006), pp.539-544.

[10] MIL-STD-498 : Software Development and Documentation, Department of Defense, United States of America, 1994.

[11] Otto, P., W. Jason, and W. Alain, “On Quality Attribute Based Software Engineering”, *27th Euromicro Conference 2001 : A Net Odyssey*, (2001), p.114.

[12] Softeng.polito.it, http://www.softeng.polito.it/pubs.html.

[13] Richard, H. and C. Sidney, “Software Requirements Engineerings”, *IEEE Computer Society, 2nd Edition*, 1997.

◈ 저 자 소 개 ◈

김 철 진 (cjkim@inhatc.ac.kr)

숭실대학교에서 소프트웨어공학 석사를 하고 동 대학에서 커스터마이제이션 기법으로 컴퓨터공학 박사를 받았다. 삼성전자에서 책임연구원으로 근무했으며, 현재 인하공전 컴퓨터시스템과 교수로 재직 중이다. 주요 관심분야로 컴포넌트개발 방법론, 컴포넌트 커스터마이제이션, 모바일 서비스, 클라우드 컴퓨팅 등이며, International Journal of Software Engineering and Knowledge Engineering, Information and Software Technology, 정보과학회, 정보처리학회, 산학기술학회 등에 다수 논문을 실었다. 주요저서는 객체지향설계 및 구현(2010) 등이 있다.

송 치 양 (cysong@knu.ac.kr)

중앙대학교에서 프로토콜 모델링 기법으로 시스템프로그래밍 석사를 하고, 고려대학교에서 계층적 모델링 방법으로 전산학 박사를 받았다. 현재 경북대학교 컴퓨터정보학부 교수로 재직 중이며, 주요 관심분야로 컴포넌트 기반 개발 방법, 모델 기반 아키텍쳐, 서비스 지향 비지니스 모델링, IPTV 등이며, International Journal of Software Engineering and Knowledge Engineering, Institute of Electronics Information and Communication Engineers, Journal of Computer Science and Technology, 정보과학회, 정보처리학회 등에 다수 논문을 실었다.

이 숙 희 (oleesh@skuniv.ac.kr)

동국대학교에서 정보처리전공으로 석사를 하고 성균관대학교 통계학과에서 전산통계전공으로 박사를 받았다. 현재 서경대학교 컴퓨터과학과 교수로 재직 중이며 주요 관심분야로 객체지향 소프트웨어 개발 방법론, 소프트웨어 테스트, 컴포넌트 기반 소프트웨어공학 등이며, 시뮬레이션학회, 정보처리학회 등에 다수 논문을 실었다. 주요저서는 자료구조(2010), 소프트웨어공학역서(2011) 등이 있다.