미래의 소프트웨어공학 기술†

고려대학교 김상수* · 인 호**
서울시립대학교 이병정**
서강대학교 박수용**

1. 서 론

소프트웨어공학이란 소프트웨어의 품질과 생산성을 향상시키기 위하여 사용자의 요구사항을 체계적으로 분석하여 설계 및 구현, 구현된 시스템의 시험 그리고 유지보수 및 폐기 시까지 소프트웨어 전 수명주기 간에 걸쳐 이루어지는 체계적인 접근법을 말한다[1,2]. 소프트웨어 시스템을 효율적으로 개발하고 관리하기 위해서는 체계적인 방법론 및 프로세스가 필수적이다.

전체 산업에서의 소프트웨어가 차지하는 비중과 중요성이 증대되어가고 있으며 하드웨어의 기능을 소프트웨어가 대체하여 기능을 제공하는 추세로 발전해 가고 있다. 임베디드 및 유비쿼터스 컴퓨팅으로 대별되는 미래의 IT 환경에서는 더욱더 소프트웨어의 비중과 중요성이 증대될 것이다[3]. 정보화 사회에서의 IT 기술의 발달과 함께 소프트웨어 개발을 위한 다양한 방법론과 지원도구 등에서 많은 발전을 가져왔다. 하지만 미래 환경에서의 소프트웨어 개발 수요와 사용자의 만족을 충족시키기에는 부족함이 있다.

이러한 문제를 해결하기 위해서는 미래 환경에 적합한 소프트웨어공학 기술을 체계적으로 정립하고 발전 전략을 수립해 나가야 한다. 이를 위해 그동안 국내·외적으로 다양한 연구가 있어 왔다.

국내의 소프트웨어 및 소프트웨어공학 관련 기술개발 전략 수립을 위한 연구로는 한국전자통신연구원(ETRI)의 "소프트웨어 공학 기술의 시장 동향"[4], 한국과학기술연구평가원의 국가기술지도(NTRM: National Technology Road Map)[5] 등을 예로 들 수 있다. [4]에서는 소프트웨어공학 시장 동향을 분석하기 위해 소프트웨어의 기술을 세부적으로 분류 분석하고, 소프트웨어공학 시장 현황 및 기술 동향을 분석하였으나 미래 환경에 적합한 소프트웨어공학 기술의 대안에 대하여 제시하지 않고 있다. [5]의 내용 중 소프트웨어공학과 관련된 부분은 정보·지식·지능화 사회 구현(소프트웨어 표준화 및 설계와 재사용 기술 부분)과 관련된 기술 로드맵의 일부분으로 다양한 소프트웨어공학 기술 중 특정 기술인 컴포넌트 기술 위주로 기술 로드맵을 제시하고 있다. 이 밖에도 소프트웨어공학 기술을 분류하고 연구대상을 선정하기 위한 노력이 있어 왔다. 이 경우도 정부의 선도사업 발굴에 목적을 두고 이루어졌기 때문에 소프트웨어를 체계적으로 분류하고 소프트웨어 산업의 체계적인 발전을 위한 방향을 제시하기에는 부족하다.

국외에서는 가장 대표적인 연구내용으로 2000년 ICSE를 통해 발표된 소프트웨어공학 로드맵(Software Engineering: A Road Map)[6]이 가장 대표적이다. 21세기를 대비하는 소프트웨어공학 분야의 연구 대상별 이슈(Issue)와 과제(Challenge)를 도출하고 세부기술과의 연관관계를 제시하고 있다. 이 역시 연구 분야의 식별에 초점을 맞추고 있어 전체 소프트웨어공학의 발전방향을 가늠하기에는 부족한 점이 많다.

따라서, 본 글에서는 미래 환경에서의 소프트웨어공학 관련 정책과 관련 기술 개발 전략을 수립하기 위한 차세대 소프트웨어공학 로드맵을 제시하고자 한다. 소프트웨어 공학 로드맵은 기존의 기술 분류 및 기술 로드맵과는 달리 소프트웨어 개발환경을 분석하기 위하여 미래사회의 전망에서 출발하여, 미래의 소프트웨어 전망과 소프트웨어공학의 시나리오를 분석하고 이에 따른 핵심 소요 기술들을 도출, 기간별 세부 연구 및 발전 전략을 수립하는 체계적인 접근법을 적용하였다. 이는 향후 소프트웨어공학의 정책을 수립하고 세부기술의 발전전략을 수립하기 위한 기초 자료로 활용될

† 본 글은 KIPA의 소프트웨어공학센터(이상은 소장)에서 주관하는 미래소프트웨어포럼 공학분과 위원님들(저자 포함, 한혁수 교수, 전진옥 비트컴퓨터 사장, 박복남 LG전자 책임연구원, 전태웅 교수, 최병주 교수, 서동수 교수, 이관우 교수, 민재식·고병선 KIPA 연구원)의 연구결과를 토대로 작성되었습니다. 본 글의 교신저자는 서강대학교 박수용 교수(sypark@sogang.ac.kr)와 고려대학교 인호 교수(hoh_in@korea.ac.kr), 서울시립대학교 이병정 교수(bjlee@venus.uos.ac.kr)입니다.
* 학생회원
** 종신회원

것이다.

이를 위해 2장에서는 미래사회에서의 소프트웨어와 소프트웨어의 전망에 대하여 알아보고, 3장에서는 미래의 전망에 따라 요구되는 소프트웨어 공학기술의 세부 핵심기술을 식별한다. 또한, 소프트웨어공학기술의 시기별 발전 방향을 포함하는 세부기술별 로드맵을 제시하고 4장에서 결론을 맺는다.

2. 소프트웨어공학의 미래 전망

2.1 미래사회의 전망

미래사회를 한마디로 표현한다면 유비쿼터스 컴퓨팅(Ubiquitous-Computing) 환경이 지배하는 사회라고 할 수 있다[7,8]. 유비쿼터스 환경은 다양한 종류의 망, 센스 및 칩, 인간 즉 환경계와 인간계가 끝임 없이(Seamless) 연결되어 활용되는 환경이다. 이 환경에서는 언제 어디에 있어도 인식하지 않고 스트레스 없이 사용되는 신개념 통합 환경(Digital Convergence)이며, 유비쿼터스의 본질은 컴퓨터 및 네트워크가 인간 생활공간의 "상황(Situation)"을 인식하는 것이라 할 수 있다.

이러한 유비쿼터스 환경의 일반적인 특징을 살펴보면 모든 컴퓨팅 장치는 네트워크로 연결, 즉 네트워크에 연결되지 않은 컴퓨터는 유비쿼터스 컴퓨팅이 아니라고 할 수 있다. 또한 컴퓨터는 사용자의 눈에 보이지 않아야(Invisible) 한다. 가상공간이 아닌 현실세계의 어디서나 컴퓨터의 사용(Embodied Virtuality)이 가능하다. 인간화된 인터페이스(Calm Technology)로서 사용자 상황(장소, ID, 장치, 시간, 온도, 명암, 날씨 등)에 따라 서비스가 변하는 특징을 갖는다.

유비쿼터스 컴퓨팅의 특성에 따라 환경을 구성하기 위해 무엇보다도 중요한 것은 모든 장치 및 사물에 컴퓨팅 장치가 내장(Embedded) 된다는 것을 의미한다. 여기에서 유비쿼터스 환경은 임베디드 시스템과 밀접한 관계를 가지며 임베디드 시스템에서의 소프트웨어가 차지하는 비중과 그 중요성이 증대되어 가고 있는 추세이다.

2000년대에 일본의 마쯔시다 전기에서는 임베디드 시스템에서의 소프트웨어의 중요성을 인식하지 못함으로 해서 커다란 실패를 경험하게 되며 이를 소프트웨어공학 적인 접근에서 해결하였다[3]. 이러한 사례는 임베디드 시스템에서의 소프트웨어의 중요성과 소프트웨어를 체계적으로 개발하고 관리하기 위한 소프트웨어공학의 중요성에 대하여 시사하는 바가 크다.

미래사회에서의 소프트웨어의 중요성은 비단 정보기술(IT: Information Technology) 분야의 학자나 일부 IT 기업에서 주장하는 편협 된 사고는 아니다. 사회학적인 측면에서는 미래의 사회를 후기 정보화 사회(PIS: Post Information Society)라고 말하기도 한다. 사회학자 김문조 박사는 미래사회를 고도의 자유도를 지닌 초 개방적 복잡계(Supra-Complexity system of High Degree of Freedom)가 될 것이며 이것이 한국 사회의 표본이 될 것이라고 말한다. PIS는 기술적 고도화 및 융합화, 기술사회적 구성체의 공고화, 사이버 문화를 주축으로 한 제 2의 정보혁명, 이성-감성의 조화에 기반한 고개념 사회(High Concept Society)라는 특징을 나타내며 이러한 상황에서는 소프트웨어가 사회 발전의 요체로 대두될 것이라고 말하였다[9]. 이렇듯 미래사회에서의 소프트웨어의 중요성이 강조되고 있는 가운데, 유비쿼터스(Ubiquitous) 사회는 이러한 소프트웨어의 중요성이 기술적으로 실현되는 환경이라고 할 수 있다.

이상에서 살펴본 바와 같이 우리는 미래 사회를 임베디드 유비쿼터스 환경 사회라고 가정하고 이러한 환경에서 가장 중요하게 부각되고 있는 소프트웨어를 효율적으로 개발 관리하기 위한 소프트웨어 공학의 발전 전략 및 정책수립 방향을 제시하기 위한 첫 단계인 미래의 소프트웨어 개발 환경과 소프트웨어에 대한 미래의 전망을 확인하고 그와 관련된 소요기술들을 이어지는 절에서 도출한다.

2.2 소프트웨어 미래 전망

미래의 환경에 적합한 소프트웨어를 개발하기 위한 소프트웨어 공학 기술을 도출하고 발전방향을 수립하기 위해 미래 사회에서 핵심적인 역할을 할 소프트웨어에 대한 전망을 살펴보고자 한다.

임베디드 유비쿼터스 환경으로 대별되는 미래 사회에서의 소프트웨어의 전망에 대하여 많은 전문가들이 "전체 시스템에서 소프트웨어의 비중이 증대될 것이다" 등 다양한 견해를 제시한다[10]. 다양한 시각으로 미래의 소프트웨어에 대하여 조망 할 수 있지만, 본 글에서는 소프트웨어공학 기술적인 측면에서 각 분야별 기술을 도출하기 위한 목적과 관점에서 미래 소프트웨어의 특징을 전망하였다.

"*대부분의 시스템에서의 소프트웨어의 비중이 현저하게 증가 할 것이다.*" 항공기의 경우 현재 80% 이상의 기능을 소프트웨어가 담당하고 있으며, 자동차의 경우도 40%에 이르는 기능을 소프트웨어에 의지하고 있다. 이러한 추세는 정밀한 시스템일수록 더욱더 소프트웨어의 비중이 증대되고 있음을 알 수 있으며 이러한

추세는 미래에 갈수로 더욱 더 심화될 것이다[10].

"소프트웨어를 포함한 시스템은 고객과 사용자의 요구 만족을 넘어 가치창출이 요구되어지는 추세이다." 소프트웨어와 경영(Business)과의 밀접한 관계가 심화(Tightly Coupled) 되면서 소프트웨어 개발 시에 제품의 품질과 개발의 성공뿐만 아니라 개발된 소프트웨어가 고객의 가치 창출에 어떠한 도움을 줄 수 있는지 등에 대한 분석과 이의 실현을 위한 노력이 병행되어야 한다[11-13].

"미래의 소프트웨어는 다양한 인터페이스(Interface), 상황(Sutuation)의 변화, 요구(Requirements)의 역동적인(dynamic) 변화 등을 만족 시켜야 한다." 임베디드 유비쿼터스 컴퓨팅 환경에 대응할 자가 적응 및 관리 등의 기능이 일반화 될 것이다.

"고품질의 소프트웨어에 대한 인증의 요구가 증가 되며, 유비쿼터스 환경의 복잡성을 모델링할 수 있는 기법의 활용이 요구될 것이다." 임베디드 시스템은 많은 경우가 Mission Critical한 시스템이다. 이 경우 실시간 또는 고품질의 시스템 인증에 대한 요구가 증가될 것이다. 또한 다양한 망과 칩 등이 네트워크로 연결된 유비쿼터스의 특징은 이벤트의 흐름이나 상태 폭발(States Explosion) 등의 문제를 효과적으로 다룰 수 있는 방법이 요구된다.

"소프트웨어의 변경을 빠르게 전달하라는 요구가 증가하고, 소프트웨어의 개발과 진화의 명확한 구분이 없어질 것이다." 경영과 기술의 급속한 변화는 이를 지원하는 소프트웨어의 예상치 못한 변경을 빠르게 전달할 것을 요구하며 급격한 비즈니스의 환경변화와 빠른 변경의 요구는 소프트웨어를 계속 빠르게 진화 시킨다. 이것은 임베디드 시스템의 짧은 수명주기와 맞물려 더욱 더 심화될 것 이다.

"소프트웨어의 요구는 사용자 환경을 개인의 맞춤형 서비스를 요구할 것이다." 따라서 미래의 소프트웨어 기술은 Custom-made에서 Generic으로, COTS 컴포넌트를 합성하는 방법의 개발이 필요할 것이며, 노동집약형(Labor-intensive)에서 자산 집약형(Asset-intensive)으로 변화할 것이다.

"컴포넌트 기반의 소프트웨어에서 서비스 지향 소프트웨어 환경으로 개발의 패러다임이 변화할 것이다." 기존의 컴포넌트 기반의 소프트웨어 재사용이 서비스들의 통합 구성을 통해 다양한 소프트웨어를 개발할 수 있는 방법도 유용하게 사용될 것이다.

"임베디드 유비쿼터스 환경의 특성에 따라, 짧은 수명주기, 하드웨어와 소프트웨어의 협력개발, 실행 중(Run-time) 소프트웨어의 변경, 소프트웨어 개발을 지원하기 위한 자동화된 도구의 사용이 증가되고 전체 소프트웨어 개발 프로젝트의 개발에도 자동화된 효율적인 방법이 적용이 요구될 것이다."

이상에서 제시한 미래 소프트웨어의 다양한 전망들을 바탕으로 요구공학(Requirements Engineering), 아키텍처 및 설계(Architecture & Design), 정형기법(Formal Specification & Mathematical Foundation of SE), 품질 및 테스트(Quality, Vali- dation, Verification and Accreditation), 진화 및 역공학(Evolution and Reverse Engineering), 소프트웨어 재사용(Software Reuse), 소프트웨어 프로세스(Software Process), 도구 및 환경(Tools and Environments), 소프트웨어 프로젝트 관리(Software Project Manage- ments), 임베디드 유비쿼터스 환경에서의 소프트웨어공학(Software Engineering for Embedded Ubiqui- tous) 등 총 10개 분야의 소프트웨어공학과 관련된 기술들이 나아가야 할 방향을 제시하고자 한다.

2.3 소프트웨어 공학기술의 미래 시나리오

미래의 소프트웨어 전망을 통해서 도출된 소프트웨어공학 세부 기술별 미래 시나리오에 대하여 살펴보면 다음과 같다.

2.3.1 요구공학(Requirements Engineering)

미래의 요구공학 분야에서는, 고객 및 사용자의 기대 만족을 넘어 가치창출이 요구되어진다. 즉, 고객의 요구 사항을 파악 하고 만족시키기 위한 요구공학의 기본 틀을 넘어 요구사항의 가치를 평가하고 새로운 고객가치를 창출하는 요구공학 기법이 필요하게 될 것이다. 이때, 가치분석(Value Analysis), 창의적 요구공학 프로세스(Creativity Requirements Engineering Process), 비즈니스 기반 요구사항 분석(Business-based Requirements Analysis), 비즈니스 전략(Business Strategy) 등과 접목하여 고객 가치를 창출하게 된다. 이것은 생산성과 품질에 대한 경쟁이 심화됨에 따른 영향으로부터 기인한다. 따라서 가치창출은 소프트웨어 개발의 새로운 목표가 된다.

유비쿼터스 환경에서 다양한 인터페이스(Interface), 상황의 변화, 요구의 역동적인(Dynamic) 변화 등을 만족 시킬 수 있어야 한다. 개발 시의 요구사항 분석을 넘어 실행 중에 요구사항 변화에 대처 할 수 있는 기능이 필요하다. 실행중 감시(Run-time Monitoring), 상황인지(Context-awareness), 런타임 소프트웨어공학(Run-Time Software Engineering)이 기반 기술로서 성장하게 될 것이다.

2.3.2 아키텍처 및 설계(Architecture & Design)

어플리케이션 영역과 실행 환경이 갈수록 복잡하고, 다양해짐에 따라 소프트웨어 제품들을 개별적으로 개발하는 것이 점점 어려워지고 있다. 이러한 문제를 해소하기 위해, 여러 어플리케이션 영역에 걸쳐서, 또는 특정 영역에서 공통적인 부분들을 추상화, 계층화, 모듈화 하여 핵심 소프트웨어 자산으로 구축하고 이를 조직적으로 재사용함으로써 특정 어플리케이션과 플랫폼에 맞는 소프트웨어 제품을 효율적으로 개발할 수 있다. 컴포넌트 기반 개발(CBD: Component-Based Development), 모델 기반 아키텍처(MDA: Model-Driven Architecture), 제품라인공학(PLE: Product Line Engineering) 등의 기술은 이러한 문제를 해결하기 위해 필요한 기술들이다.

또한, 소프트웨어 개발이 핵심 소프트웨어 자산을 구축하고 이를 재사용하는 자산 집약형 소프트웨어 기술로 발전하면서 아키텍처를 잘 정의하고 이를 중심으로 소프트웨어를 분석, 구현, 유지 및 관리하는 것이 더욱 중요함에 따라 아키텍처 기반 개발(ABD: Architecture-Centric Development) 방법론이 적용되어야 한다. ACD는 상위 추상화 수준의 시스템 구조를 정의한 아키텍처 모델들이 소프트웨어의 개발과 진화의 방향을 이끌어가는 개발 방식이다. 지금까지 소프트웨어 아키텍처의 다양한 뷰, 기술 언어, 모델링 방법, 분석 및 변환 기법, 재사용 및 복원 기법 등이 많이 연구되어 왔다. ACD는 개별 시스템의 아키텍처를 넘어서서 제품 계열 아키텍처, 도메인 아키텍처, 아키텍처 스타일 등과 같이 여러 어플리케이션 시스템들에 공통적인 아키텍처를 구축하고 활용하는 기술로 발전하게 될 것이다.

2.3.3 정형기법(Formal Specification & Mathematical Foundation of SE)

단기적으로는 고품질 소프트웨어에 대한 인증 요구가 증가할 것이다. 국내 보안시스템의 경우 EAL 5~7 의 고 등급을 인증 사례가 나올 것으로 보인다{현재는 EAL3~4수준, 국외의 경우 스마트카드(Smart Card) 관련 부분은 이미 EAL 5 이상의 인증을 받고 있음}. 항공 분야의 Safety-critical System은 RTCA/DO-178B level A, B,C의 인증서가 발급 된다. 또한, 유비쿼터스 환경하의 복잡성을 모델링할 수 있는 기법이 활용된다. 이벤트 흐름과 상태 폭발의 문제를 효과적으로 다룰 수 있는 정형이론과 도구가 개발될 것이고, Safety-critical Ubiquitous System개발에 적용될 것이다.

장기적으로는 소프트웨어 컴포넌트의 활용이 늘어나며 품질관련 문제가 상당부분 해결될 것이다. Proof-carrying Code, Context-aware Component 기술이 안정될 것이다. 또한, 기능과 품질 요구를 통합하여 지원하는 정형기법이 활용될 것이다. 정형명세로부터 자동 생성되는 코드의 품질 문제(효율성, 유보수성, 재사용성)가 상용화 수준으로 개선될 것이다.

2.3.4 품질 및 테스트(Quality, Validation, Verification and Accreditation)

테스팅(Testing)과 관련된 기술은 이미 70년대에 모두 정리가 완료되었다. 이후부터 현재까지는 물론 미래에도 테스팅 기술은 소프트웨어의 발전 추세에 맞게 테일러링 되어 사용될 것이다. 임베디드 유비쿼터스 환경에 적합한 소프트웨어를 검증하기 위한 기술들이 필요하게 된다.

2.3.5 진화 및 역공학(Evolution and Reverse Engineering)

미리 예상하지 못한 소프트웨어 변경을 빠르게 전달하라는 요구가 급속하게 증가할 것이다. 따라서 가정과 추정을 많이 하지 않는 Reactive Evolution 방법론이 요구되며, Concept Location, Change Propagation 기술을 포함한 Program Comprehension 핵심 기술이 필요하게 될 것이다.

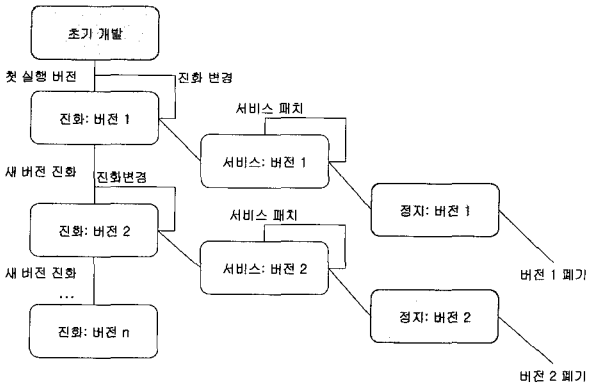

그림 1 버전 단계 모델[14]

그림 1의 버전 단계 모델[14]에서 보는 바와 같이 소프트웨어 개발과 진화의 명확한 구분이 없어질 것이다. 매우 복잡한 조직의 진화를 보여주는 생물학적인 모델에 기반한 학습에 의하여 소프트웨어를 진화시키기 위한 새로운 모델이 요구되며, 제품 기반이 아니고 서비스에 기반한 소프트웨어 진화 모델이 필요할 것이다.

초기 설계 단계부터 레거시 소프트웨어(Legacy Software) 단계까지 전 수명 주기(Life Cycle) 전체에 걸쳐 소프트웨어 변경을 지원하기 위한 역공학(Reverse Engineering) 기법과 도구가 사용될 것이다. 연속적인 프로그램 이해(Continuous Program Understanding)를 위한 기법과 도구가 사용될 것이며, 이해되지 않는 시스템 복구를 시도하기보다는 순공학의 작은 루프들

에서 역공학이 점증적으로 적용될 것이다.

2.3.6 소프트웨어 재사용(Software Reuse)

경쟁적인 시장에서 우위를 선점하기 위해서는 소프트웨어 개발 기업은 소프트웨어의 생산성 및 품질 향상과 납기 단축에 총력을 다 할 것이다. 소프트웨어 개발 시 개발 비용 및 기간을 줄이고, 고 품질의 소프트웨어를 생산하기 위해서 기존에 개발되고 검증된 소프트웨어를 재사용하여 새로운 소프트웨어를 개발할 수 있어야 한다.

고객 혹은 시장의 다양한 요구에 따라 다양한 소프트웨어를 쉽게 생산할 수 있는 Mass Customization이 가능해야 할 것이다. 소프트웨어의 Mass Customization을 위해서는 공통된 소프트웨어 플랫폼을 바탕으로 가변점과 가변치를 정의하여 사용자 요구의 변화에 따라 해당되는 가변치를 가변점에 결합함으로써 다양한 소프트웨어를 체계적으로 생산할 수 있어야 한다.

다양한 COTS 컴포넌트 및 레거시 소프트웨어와의 상호운영성이 필요할 것이다. 기존에 존재하는 COTS 컴포넌트나 레거시 소프트웨어를 재사용하여 새로운 소프트웨어를 개발하거나, 새로이 개발된 소프트웨어가 기존 레거시 소프트웨어와 연동될 필요가 있을 것이다.

공개 소프트웨어의 활용을 통한 소프트웨어의 개발과 확산이 활성화될 것이다. 소프트웨어를 개발 시에 모든 사용자가 사용가능한 공개 소프트웨어를 바탕으로, 필요한 만큼 수정 및 확장을 하여 새로운 종류의 소프트웨어를 개발할 수 있을 것이다.

컴포넌트 기반의 소프트웨어에서 서비스 지향 소프트웨어 환경으로 소프트웨어의 개발 패러다임이 변경될 것이다. 기존의 컴포넌트 기반의 소프트웨어 재사용은 특정 아키텍처나 플랫폼에 의존적인 단점이 있는데, 특정 플랫폼에 의존하지 않고 표준 상호작용에만 의존하는 독립적인 서비스들의 통합 구성을 통해 다양한 소프트웨어를 개발할 수 있을 것이다.

2.3.7 소프트웨어 프로세스(Software Process)

소프트웨어의 프로세스 기술 관련 시나리오는 "소프트웨어 개발의 많은 부분이 요구사항으로부터 자동 생성 될 것인가?", "소프트웨어 개발에 참여하는 이해관계자(Stakeholder)들의 협업 정도가 얼마나 될 것인가?" 따라 달라진다.

소프트웨어 프로세스는 개발 기술과 밀접한 관계를 가지고 있다. 학계에서는 프로세스의 자동화 및 정형화 연구를, 업체에서는 프로세스 맵과 흐름도로 프로세스를 표현하고 적용을 통해 개선을 추구하는 방향으로 진행될 것이며, 자동화가 가능하기 위해서는 COTS 컴포넌트 합성이 이루어져야 할 것이다.

2.3.8 도구 및 환경(Tools and Environments)

일반적으로 소프트웨어 공학에서 사용되는 도구들은 다음과 같이 다양하게 분포되어 있다. 요구공학, 아키텍처, 소프트웨어 재사용, 역공학, 소프트웨어 테스팅과 품질보증을 지원하는 소프트웨어 테스팅 및 품질관리 도구, V&V, 소프트웨어 프로세스 효과적 지원 및 관리 도구, 정형기법을 적용한 요구공학 지원 도구, 소프트웨어공학 환경과 통합된 도구 등 다양한 도구들의 궁극적인 목표는 소프트웨어 개발 및 관리의 자동화라고 할 수 있다. 지금의 자동화 수준은 10~15% 수준 정도이며, 향후 자동화 정도는 50% 이상으로 올리기 위한 지속적인 노력이 이루어져야 한다. 이러한 자동화는 개발 및 지원 도구들의 통합화와 함께 소프트웨어 엔지니어들의 이슈가 될 것이다.

2.3.9 소프트웨어 프로젝트 관리(Software Project managements)

향후 소프트웨어 프로젝트 관리의 최대 이슈는 수동적에서 능동적으로 변화한다는 것이다. 다양한 형태의 위험(Risk)를 관리하기 위한 위험관리 기법이 도입 될 것이다. 소프트웨어 비용의 개념이 고객의 가치창출에 의한 산정 방법으로 변화될 것이다. 이것은 투입된 자원을 토대로 산정된 비용의 개념에서 고객의 만족에 따른 가치산정 모형 형태로 변환됨을 의미한다. 또한 투자대비 효과 분석이 무형 자산(Invisible Asset)으로 전환 될 것이다. 생산 소비자(Prosumer) 등의 신조어가 등장하며, 디지털 경제에서는 기존의 파레토 법칙(Pareto law) 대신에 Google 사례에서 보듯이 Long Tail의 개념으로 대체 되며 고객 관계, 브랜드 등의 무형자산이 투자 효과의 핵심으로 부각될 것이다.

2.3.10 임베디드 유비쿼터스 소프트웨어공학(Software Engineering for Embedded & Ubiquitous)

미래사회의 IT 기술 환경은 임베디드 유비쿼터스 컴퓨팅환경이라고 할 수 있다. 임베디드 유비쿼터스 환경의 기술적인 특징과 이슈들로는 통합(Convergence), 복잡성(Complexity), 상태 폭발(State Explosion), 상호운용성(Interoperability), 개인화(Personalization), 프라이버시(Privacy), 보안(Security), 정황인지(Situation Awareness), 자가 적응(Self-Adaptive · Configuration · Optimization · Healing · Protection · Planning), 실시간(Real-time), 신뢰성(Reliability) 등과 같은 유비쿼터스 환경화되면서 필요한 소프트웨어의 특징으로부터 기인한 것과, 일반적인 임베디드 시스템의 특징인 초소형화, 제한된 메모리, 저비용, 저에너지 소모, 시장 적시성(Time to Market) 문제 등이 핵심적인 이슈가 될 것이다.

3. 미래 소프트웨어 공학 로드맵

미래 사회의 전망에서 출발하여 소프트웨어공학 기술의 시나리오를 바탕으로 미래 소프트웨어공학의 로드맵을 제시하고자 한다. 로드맵 작성은 시나리오 기반(Scenario Based)으로 하였다. 먼저, 미래 시나리오에 따라 필요한 세부기술을 도출하고 핵심 기술을 식별한다. 다음으로, 세부 기술 시기별로 로드맵을 제시하고자 한다.

3.1 미래 소요 핵심기술

미래의 소프트웨어 공학 기술의 시나리오에 따라 다음과 같은 핵심 기술들을 도출하였다. 그림 2는 미래의 소프트웨어공학 기술의 시나리오에 따라 소요되는 분야별 세부 기술의 현황을 보여주고 있다.

요구공학(Requirements Engineering) 미래의 시나리오를 뒷받침하기 위해 소요되는 요구공학 관련 핵심 기술로는 전통적인 요구공학 기법과 함께 가치기반 요구공학 기법, 런타임(Run-time)을 지원하는 요구공학 기법 등이 요구된다.

아키텍처 및 설계(Architecture & Design) 분야에서는 소프트웨어의 설계기술은 컴포넌트 기반에서 아키텍처 기반 개발의 패러다임으로 변화하면서 설계결정과 품질속성 사이의 관계 분석 기술, 뷰 별 표현 및 다중 뷰 통합을 지원하는 아키텍처 기술 언어기술, 아키텍처와 코드 사이의 일치성 보장, 동적 환경 적응형 시스템 개발을 지원하는 아키텍처 설계, 아키텍처 지식의 자산화와 관련된 기술들이 요구된다.

정형기법(Formal Specification & Mathematical Foundation of SE) 분야에서는 정형기법 및 수학을 기반으로 한 소프트웨어공학 기술로는 전통적인 정형기법 분야의 기수인 정형 명세기술, 정형 검증기술 그리고 환경구축기술 등의 기술이 임베디드 유비쿼터스 환경에 맞게 테일러링되어 사용될 것이다.

품질 및 테스트(Quality, Validation, Verification and Accreditation) 관련된 기술은 기존의 기술을 중심으로 임베디드 환경의 특징에 맞게 적합한 방법으로 테일러링된 테스트 프로세스(Test Process), 테스트 기술(Test Technique), 테스트 기반(Test Infrastructure) 기술이 요구 된다.

진화 및 역공학(Evolution and Reverse Engineering) 분야에서는 시스템 동역학(Systems Dynamics)을 포함한 타 분야의 학문을 적용한 기법과 기존의 분석기법 및 프로세스 기술이 미래 환경에서 사용될 것이다. 시간에 따라 소프트웨어 변경을 모델링하는 시스템 동역학(System Dynamics), 프로그램 이해(Program Comprehension), 변경 연구(Change Study), 역공학(Reverse Engineering), 공학 프로세스(Engineering Process) 그리고 검증 등의 기술이 요구된다.

소프트웨어 재사용(Software Reuse) 분야에서는 아키텍처 및 서비스 기반의 소프트웨어의 재사용을 위한 자산 명세기술, 자산개발기술, 자산관리기술을 포함하여 다양한 환경에 적응할 수 있는 소프트웨어 적응기술 합성 기술 등이 소요 된다.

소프트웨어 프로세스(Software Process) 분야에서는 정량적인 프로세스관리 기법과 프로세스의 최적화 기술 자동화 기술 등이 필요하다.

도구 및 환경(Tools and Environments) 분야에서는 개발 프로세스의 통합관리 및 자동화를 위한 도구 및 환경 기술이 요구되며, 특히 요구공학, 아키텍처 지원도구 및 환경, V&V 지원도구 및 환경, 소프트웨어 재사용, 역공학 지원도구 및 환경, 소프트웨어 프로세스 효과적 지원 및 관리 도구, 정형기법 지원도구 그리고 소프트웨어 공학 지원도구들의 통합화와 관련된 기술들이 요구된다.

소프트웨어 프로젝트 관리(Software Project managements) 부문에서는 제품 및 프로세스 재원 등을 통합적으로 관리할 수 있는 환경이 구축되어야 하며 요구되는 요소 기술들은 기본 기술들로는 제품개발관리 기술, 프로세스 관리 기술, 자원 관리 기술 그리고 정량적인 관리를 위한 측정 기술 등이다.

임베디드 유비쿼터스 소프트웨어공학(Software Engineering for Embedded & Ubiquitous) 기술은 임베디드 유비쿼터스 환경에 요구되는 소프트웨어를 효율적으로 개발하기 위한 기술들로 대부분 기술들이 기본의 소프트웨어공학의 범주에 포함된다. 하지만 기존의 기술 분야와는 차별되는 패러다임을 통한 접근이 필요하며 임베디드 유비쿼터스라는 환경에 특수하게 적용되는 핵심기술들은 서비스 충돌 해결(Service Conflict Resolution), 정황인지(Situation Awareness), 소프트웨어와 하드웨어 Co-design, Incremental Model Checking, 실생 중 검증(Run-time Verification), 보안공학(Security Engineering), 자가 치료(Self-healing), Agile Process등이 요구되며, 이러한 기술들은 새로운 독립된 연구 분야로 취급 된다.

3.2 소프트웨어공학 기술 로드맵

소프트웨어공학 기술의 미래에 대한 로드맵은 단기(5년), 중기(10년) 및 장기(15년)로 구분하여 전개하여 그림 3과 같이 나타낼 수 있다. 그림에서 보는 바와 같이 각 기술을 시기별 성숙도를 고려하여 어떠한 전략으로 발전시킬 것인지를 제시하였다.

소프트웨어공학 기술	소요 핵심 기술
요구공학 **(Requirements Engineering)**	❑ 요건분석/협상/충돌해결: Requirements Analysis, Negotiation, Service Conflict Solving ❑ 가치기반 요구공학: Value analysis, Business Process ❑ 실행 중 요구공학: Run-time Monitoring, Self-managed, Context-Awareness
아키텍처 및 디자인 **(Architecture & Design)**	❑ 설계결정, 품질속성 분석: Design Trade-off Analysis, Quality-driven Design Decisions, Defining Special-purpose Language for Architecture Description ❑ 다중 뷰 통합기술, 동적환경 지원 아키텍처: Context-, QoS-, and Self-aware Systems Architecture, Runtime Architecture ❑ 지식의 자산화 자산 관리 기술: Reference Materials for Domain-Specific Architectures, Analysis Techniques
정형기법 **(Formal Specification & Mathematical Foundation of SE)**	❑ 명세기법: State-based Specification, Process-based Specification Technology ❑ 검증기법: Formal Verification, Model Checking ❑ 자동화 기법: Code Generation, Supporting Tool Construction
테스트 **(Testing)**	❑ 시스템 테스트, V&V, 품질관리: System Testing, Validation Verification and Accreditation, Quantitative Quality Management ❑ 자동화된 결함감지: Automatic Detecting
진화 **(Evolution & Reverse Engineering)**	❑ 형상/변경관리: System dynamics, Change Model, Change Management, Change Impact Analysis, Concept location, Change Propagation, Configuration Management ❑ 진화 프로세스: Versioned-Staged Model, Mini-cycle Process ❑ 역공학: Continuous Program Understanding, Reverse Engineering Process, Data Analysis, Concept Abstraction, Legacy system reengineering,
재사용 **(Reuse)**	❑ 자산 설계: Component Specification, Service Specification, Variance Modeling, Architecture Design, Component Design, Service Implementation ❑ 분석/관리: Architecture Analysis and Asses, Inconsistency Analysis, Domain Analysis, SW Asset Management, Searching, Asset Mining, Inconsistency Resolve, Parameterized Programming, Macro Processing ❑ 자산 적응 및 합성: Architecture Component Convergence, Service Binding, Dynamic Conjunction
프로세스 **(Process)**	❑ 정량화/표준화: Quantitative Management Method, Process Standardization, Process Asset Library, Project DB, Measurement Repository ❑ 최적화: software Process Optimization, Process Change Management, Technology Change Management ❑ 자동화: Software Process Automation
도구 및 환경 **(Tools and Environments)**	❑ 소프트웨어개발지원: SW analysis & Design, requirements, Architecture Supporting Tools, Component Development Tools, V&V, Testing, Quality Management, Configuration Supporting Tools ❑ 프로세스, 프로젝트관리 지원: SW Project Management Tools, SW Process Management Tools, Integrated Management Tools ❑ 통합된 환경 자동화 지원: Run-time, Self-adaptive Environment Support tools
프로젝트 관리 **(Project Management)**	❑ 제품 개발 관리: Document Management, Code Management, Data Management ❑ 프로세스 관리: Schedule/Risk/Configuration Management, Tailoring ❑ 자원 관리: Resource Allocation, Productivity Analysis, Cost-Benefit Analysis, Measurement Metrics
임베디드 유비쿼터스 소프트웨어공학 **(Software Engineering for Embedded & Ubiquitous)**	❑ 상황인지: Situation Aware Logic, Situation Aware Middleware, V&V ❑ 자가적응: Self-Adaptive/Reconfigurable Architecture, fault tolerance ❑ 임베디드 유비쿼터스 시스템 개발: HW/SW Co-design, Agile Process, Interoperability, Wrapping Technology, Resource management ,Incremental Model Checking, Run-time, Dynamic Verification

그림 2 미래 소요 소프트웨어공학 핵심기술

먼저 **요구공학(Requirements Engineering)**의 경우 단기적으로는 요구공학의 정착화, 대중화를 위하여 기본적인 요구사항의 중요성 인식시키기 위한 지속적인 노력을 기울여야 한다. 이때 기업이 경우 요구사항 관리자(Requirements Manager)와 요구사항을 전문적으로 관리할 수 있는 조직(Requirements Team)이 구성되어야 한다. 요구공학 기법 및 프로세스에 대한 연구가 성숙되어야 한다. 중·장기적으로 가치 창출, 창의적 요구공학활성화 및 가치분석, 비즈니스와의 접목을 통해 가치 중심의 요구공학기법을 발전시킨다. 이는 장기적인 목표로 실행중 요구공학(Run-time Requirements Engineering), 상용화를 위한 실행중 요구사항 감시(Run-time Requirements Monitoring) 기술, 자가 관리(Self-managed) 기술 등과 병행하여 이루어져야 할 것이다.

아키텍처 및 설계(Architecture & Design) 분야의 기술을 살펴보면 단기적으로는 CBD 방법론의 정착화와 ABD 방법론에 대한 연구가 이루어지고, 중기적으로는 아키텍처 기반 개발 방법론을 적용하기 위한 노력이 있어야 한다. 이러한 노력은 장기적으로 축적된 자산화 된 아키텍처를 적용할 수 있도록 할 것이며, 아키텍처개발 방법론의 정착에 힘을 기울여야 한다.

정형기법(Formal Specification & Mathematical Foundation of SE) 분야에서는 단기적으로 전문가 POOL의 구성하되 B, Z, CSP, Petri-net 등 기법별 정형기법 전문가 그룹을 구성하고 활동을 지원한다. 정형기법 교육 컨텐츠 구축, 컨설팅 전문가 육성을 위한 교육 자료 및 대학 교육을 위한 교재를 편찬한다. 실무 적용의 시작 단계로 해당분야의 IT 구축에 정형기법을 활용할 경우 국외 전문가 집단과 연계하여 기술도입을 추진한다. 중기적으로는 기술성장을 위하여 고등급 인증 대상의 확산을 위한 노력이 필요하다. 정형기법이 성공적으로 적용될 수 있는 통신, 반도체, 기타 정밀 분석이 요구되는 분야로 적용을 확산 한다. 또한 대학 교육을 통해 배출된 고급 인력이 산업체에 투입되어 정형기법을 전문으로 하는 기업들이 배출되도록 한다. 이렇게 될 경우에 기술 안정기간인 장기적으로는 이 분야의 선도 기술을 가진 국가와 동등한 수준에서 프로젝트를 수행할 수 있게 될 것이다.

품질 및 테스트(Quality, Testing, V&VA) 분야는 단기적으로 테스팅에 대한 중요성을 인식시키고, 시스템 테스팅에 집중 하되 개발자가 테스트 하는 것으로 앞당기는 노력이 지속되어야 한다. 결함 수명주기(Defect Life cycle)가 테스팅과 연계되었을 때 가능하다. 중장기적으로 미래 환경에 맞는 테스팅 기술을 적용해야 한다. 이때에는 자가 탐지(Self Detecting) 하는 테스팅 기술의 개발을 목표로 한다.

진화 및 역공학(Evolution and Reverse Engineering) 분야에서는 단기적으로는 진화 방법론과 프로그램 분석 기술 및 역공학 방법론에 대한 연구와, 급격한 비즈니스 환경 변화와 그에 따른 시장 적시성(Time to Market)으로 인한 미리 예상치 못한 소프트웨어의 빠른 변경 요구됨에 따른 기술 및 지원도구의 개발이 필요하다. 장기적으로는 에는 소프트웨어가 더욱 복잡해짐에 따라 생물학적인 유사성을 가진 모델로부터 학습에 의한 새로운 소프트웨어 진화 모델을 적용한 기술을 개발한다.

소프트웨어 재사용(Software Reuse)은 단기적으로 컴포넌트 및 아키텍처로 구성된 자산의 명세·개발·분석·관리 기술과 관련된 컴포넌트 명세, 서비스 명세, 가변성 모델링, 아키텍처 설계 기술, 서비스 구현 기술, 일치성 분석 기술, 소프트웨어 자산의 관리 및 검색기술, 자산 마이닝 기술 등의 연구개발에 주력 한다. 중·장기적으로는 임베디드 유비쿼터스 환경에 적합한 소프트웨어를 개발하기 위한 소프트웨어 적응기술 및 소프트웨어 합성기술의 개발이 필요할 것이다.

소프트웨어 프로세스(Software Process)는 단기적으로 소프트웨어 프로세스의 정량적 관리가 가능하도록 하고 표준 프로세스가 정립되어야 한다. 이것은 프로세스 자산 라이브러리(Process Asset Library), 프로젝트 데이터베이스(Project DB), 측정 저장소(Measurement Repository) 등의 구축과 함께 이루어져야 한다. 중기적으로 소프트웨어 프로세스의 최적화 작업과 프로세스 변경관리(Process Change Management), 기술 변경관리(Technology Change Management) 등의 기술을 성숙시키는 것을 목표로 하고 장기적으로는 소프트웨어 프로세스의 자동화 및 정형기법(Formal Method)과의 결합 등의 기술을 성숙시켜야 한다.

도구 및 환경(Tools and Environments)과 관련해서는 단기적으로는 소프트웨어 단독 개발 효율성, 효과성 증대, 개발 도구의 사용성 증대를 위한 노력이 필요하며, 개발 관리 도구의 효율성 증대, 재사용 객체 생성 및 관리 도구가 요구된다. 중기적으로는 소프트웨어 군별 개발 효율성, 효과성 증대, 전략적 소프트웨어 재사용 객체 생성 및 관리 지원, 소프트웨어공학과 비즈니스 및 마케팅의 접목을 위한 노력이, 장기적으로는 환경 적응형 통합 개발 환경 및 관리 효율성, 효과성 증대, 자가 적응(Self-adapted) 개발 지원 및 관리 환경과 도구 기술, 자가 관리(Self-managed) 개발 지원 환경과 도구 기술의 성숙에 초점을 맞추어야 한다.

그림 3 소프트웨어공학 로드맵

소프트웨어 프로젝트 관리(Software Project managements) 기술은 단기적으로는 소프트웨어 프로젝트 관리를 위한 위험관리 기술, 프로세스 관리 기술 등 기본적인 기술의 개발과 함께 소프트웨어 개발에서 체계적인 프로젝트 관리 기법의 적용에 중점을 두어야 한다. 중·장기적으로는 프로젝트 관리의 통합화 자동화에 필요한 기술의 개발과 적용, 유비쿼터스 컴퓨팅 환경을 이용한 자동화된 관리 자료의 수집 체계를 이용한 프로젝트 관리 기술의 적용이 필요하다.

임베디드 유비쿼터스 소프트웨어공학(Software Engineering for Embedded & Ubiquitous)에 포함된 기술은 대부분 성숙되지 않은 기술들로 구성되어 있다. 하지만 이러한 기술을 Driven 하는 임베디드 유비쿼터스 기술은 이미 실용화되어 IT 분야의 핵심으로 또한 기술 환경과 패러다임의 변화 일으키고 있다. 따라서, 각 분야기술의 개발과는 별도로 영역을 구축하고 연구개발 되어야 한다. 단기적으로는 개발 수명주기 간에 적용되는 각각의 기술들에 대하여 연구개발을 시작하여 중·장기적으로는 이러한 기술이 성숙될 수 있도록 지속적인 투자와 지원이 필요할 것이다.

4. 결 론

지금까지 미래 환경에서의 소프트웨어공학 기술의 발전 전략을 수립하기 위한 로드맵을 전개하였다. 로드맵의 작성은 미래 사회에 대한 소프트웨어의 전망과 소프트웨어공학기술의 시나리오에 따라, 소요되는 핵심 기술을 중심으로 시기별 발전 전략 수립을 위한 개략적인 방향을 제시하였다.

미래의 소프트웨어공학 기술은 기존의 전통적인 소프트웨어 수명주기를 중심으로 구별된 기술들을 소프트웨어 개발 시 체계적으로 적용하는 소프트웨어공학에 대한 중요성 인식과 정착화가 우선시 되어야 한다. 또한 미래사회의 가장 주목할 만한 특징인 임베디드 유비쿼터스 컴퓨팅 환경의 추세에 맞는 '임베디드 유비쿼터스 소프트웨어공학'의 정립과 함께 이에 따른 관련 세부기술에 대한 연구개발에 노력하여야 할 것이다. 임베디드 유비쿼터스 소프트웨어공학 기술은 기존의 기술 영역에 포함되지만 소프트웨어 공학 전문가들의 패러다임의 변화가 이루어질 때까지는 독자적인 연구영역으로 구축되어야 할 것이다.

소프트웨어공학 기술은 기존의 다양한 분야 기술과 전문가들을 포함하는 다학제 간의 통합적 접근이 요구되는 기술이다. 현재에도 IT 기술의 발전과 함께 소프트웨어공학에 적용할 수 있는 다양한 기술들이 개발되어 발전하고 있으며, 소프트웨어공학에서의 수요만을 기다리고 있다. 실질적으로 소프트웨어 개발을 위한 공학기술로의 접목을 위해서는 체계적인 연구와 지원이 따라야 한다. 소프트웨어공학 기술 연구를 위한 연구소가 산·학·연을 중심으로 설립되고 운영되어야 한다. 임베디드 유비쿼터스 환경에서의 명실상부한 세계최고의 IT 강국으로 거듭나기 위해서는 소프트웨어의 중요성과 이를 체계적으로 개발·관리할 수 있는 소프트웨어공학에 대한 아낌없는 투자가 있어야 할 것이다.

참고문헌

[1] I. Sommerville, "Software Engineering - Seventh Edition," Addition Wesley, Boston, 2004.

[2] IEEE Computer Society, "SWE BOK: Software Engineering Body of Knowledge," IEEE Computer Society, 2005.

[3] 한국정보산업연합회, "해외 기업의 임베디드 소프트웨어 개발역량 강화 사례," 한국정보산업연합회, 임베디드소프트웨어산업협의회, 2006.

[4] ETRI, "소프트웨어공학 기술 및 시장동향," 한국전자통신연구원, 소프트웨어 연구소, 2001.

[5] 과학기술부 외 , "국가기술지도-총론," 한국과학기술연구평가원, 2002.

[6] A. Finkelstein and J. Kramer, "Software Engineering: A Roadmap," Proc. of the Conference on The Future of Software Engineering, 2000.

[7] Weiser M. (1995) Ubiquitous computing: (invited talk). *Proc. USENIX Conference*, 1995.

[8] Mark Weiser, "The Computer for the 21 Century," PERVASIVE Computing, Reaching for Weiser's Vision, January-March 2002.

[9] Rolf Jensen, "The Dream Society," McGraw-Hill, New York, 1999.

[10] 이호수, "유비쿼터스 컴퓨팅의 핵심 기술 임베디드 시스템," 신기술신경영, Spring 2004.

[11] Berry W. Boehm, "Value Based Software Engineering," ACM SIGSOFT Software Engineering Notes, vol 28 no 2 March 2003.

[12] W. Chan Kim, Renee' Mauborgne, "Blue Ocean Strategy," Harvard Business School Press, 2005.

[13] 김상수, 임상원, 박용식, 인호 "블루오션 전략을

적용한 요구공학 프로세스," 한국시스템엔지니어링학술지, 제2권 1호, pp. 11-17, 2006.
[14] Rajlich V. T. and Bennett, K. H., "A Staged Model for the Software Life Cycle," IEEE Computer, Vol. 33, Iss. 7, pp. 66 - 71, July 2000.

김 상 수

2004 국방대학교 무기체계 M&S(석사)
1993~1997 공군군수사령부 번역장교
1997~2002 전투비행단 중대장 및 참모
2004~2005 공군전투발전단 무기체계처
2005~현재 고려대학교 컴퓨터학(박사과정)
1993~현재 공군소령
관심분야 : 요구공학, 임베디드 소프트웨어 공학, 모델링 시뮬레이션(M&S)
E-mail: sookim@korea.ac.kr

인 호

1990 고려대학교 전산학과(학사)
1992 고려대학교 전산학과(석사)
1998 USC Computer Science(박사)
1993~2003 Texas A&M University 조교수
2003~2006 고려대학교 컴퓨터학과 조교수
2006~현재 고려대학교 컴퓨터·정보통신학부 부교수
관심분야 : 요구공학, 임베디드 소프트웨어 공학, 소프트웨어 보안, 정황인지 미들웨어
E-mail : hoh_in@korea.ac.kr

이 병 정

1990 서울대학교 계산통계학과(학사)
1998 서울대학교 전산과학과(석사)
2002 서울대학교 컴퓨터공학부(박사)
1990~1998 현대전자 S/W연구소 연구원
2002~2004 서울시립대학교 컴퓨터과학부 전임강사
2004~현재 현재 서울시립대학교 컴퓨터과학부 조교수
관심분야 : 소프트웨어 진화, 재공학, 웹 공학
E-mail : bjlee@uos.ac.kr

박 수 용

1986 서강대학교 전자계산학과(공학사)
1988 플로리다 주립대 전산학(석사)
1995 George Mason University 정보기술학(박사)
1995년 George Mason University 연구 조교수
1996~98 TRW Senior Software Engineer
1998~2002 서강대학교 컴퓨터학과 조교수
2002~현재 서강대학교 컴퓨터학과 부교수
관심분야 : 요구공학, Adaptable Components, Web Services
E-mail : sypark@sogang.ac.kr

• 제43차 전산관련학과 교수 세미나 •

• 일 자 : 2007년 1월 4~5일
• 장 소 : 유성호텔
• 내 용 : 논문발표 등
• 주 최 : 전문대학전산교육연구회
• 문 의 처 : 한양여자대학 성해경 교수
Tel. 02-2290-2207