극한프로그래밍 방법론 도입을 위한 가치와 실천에 대한 연구

A Study on the Values and Practices of the Extreme Programming for Its Adoption

저자 (Authors)
이상현, 이상준
Sang-Hyun Lee, Sang-Joon Lee

출처 (Source)
한국컴퓨터정보학회논문지 13(7), 2008.12, 269-280 (12 pages)
Journal of the Korea Society of Computer and Information 13(7), 2008.12, 269-280 (12 pages)

발행처 (Publisher)
한국컴퓨터정보학회

URL
http://www.dbpia.co.kr/Article/NODE06527794

APA Style
이상현, 이상준 (2008). 극한프로그래밍 방법론 도입을 위한 가치와 실천에 대한 연구. 한국컴퓨터정보학회논문지 , 13(7), 269-280.

韓國 컴퓨터情報學會 論文誌
第13卷 第7號, 2008. 12.

2008-13-7-5-5

극한프로그래밍 방법론 도입을 위한 가치와 실천에 대한 연구

이 상 현*, 이 상 준**

A Study on the Values and Practices of the Extreme Programming for Its Adoption

Sang-Hyun Lee *, Sang-Joon Lee **

요 약

소프트웨어 개발방법론의 선택은 항상 고민거리가 되어왔다. 극한 프로그래밍(XP) 방법론 도입의 문제점을 소프트웨어 개발 현장에서도 찾아보는 노력이 필요한 시점이다. 본 논문에서는 점점 대중화되고 있는 XP 방법론의 도입 현황을 살펴본 후, XP의 구성요소인 "가치"와 "실천"이 개발자들에게 어떻게 받아들여지고 있는지를 유용성과 이용용이성을 기준으로 조사하였다. 연구 결과 XP는 소프트웨어 개발 과정의 품질과 적기를 개선시키는데 매우 성공적이라고 받아들여지고 있었다. 또한, XP의 애자일 철학과 적용의 기반이 되는 "가치"와 "실천"은 성취하기가 비교적 수월했지만, 이중에서 기술에는 거의 종속되지 않은 의사소통, 피드백, 공동 소유권, 짝 프로그래밍, 단순설계, 고객 참여와 같은 요소는 성취하기 어렵다고 파악되었다. 창의적이고 노동 집약적인 소프트웨어 개발 현장에서 XP가 더욱 잘 수용되기 위해서는, 본 연구결과가 시사하는 바와 같이 관리자나 교육자들이 기술적 요소 외에도 인간적인 특성에 기반한 가치와 실천의 활용에 보다 많은 노력을 해야 한다.

Abstract

The choice of appropriate methodologies has always been a challenge. It is time to find obstacles when Extreme programming(XP) is adopted. In this paper, we invest status of adoption of XP methodology which is more popularized. We survey how the Values and the Practices of XP can be adopted to developer in terms of the usefulness and the easy of use of XP. As a result, it is regarded as a successful one in improving quality and timeliness through software development process. The Values and Practices, a basis of XP philosophy and its adoption, could be easily accomplished, but other elements - communication, feedback, collective ownership, pair programming, simple design and customer participation - which do not depend on technology could not be accomplished easily. To introduce XP easily in the creative and labor intensive software development fields, the managers and the educators have to make efforts to apply these Values and Practices based on humane characteristics as well as technical elements.

▸ Keyword : 극한프로그래밍(Extreme Programming), 방법론(Methodology), 가치와 실천(Values and Practices), 정보기술수용(Acceptance of IT)

• 제1저자 : 이상현 교신저자 : 이상준
• 접수일 : 2008. 10. 1, 심사일 : 2008. 10. 20, 심사완료일 : 2008. 12. 24.
* 호남대학교 정보통신학과 시간강사 **전남대학교 경영학부 교수

Ⅰ. 서 론

정보기술(IT)에 의해 디지털 경제는 많은 발전을 거듭해왔다[1]. 다른 산업 분야의 발전을 도모했던 IT는 자신의 가장 핵심 요소인 소프트웨어 개발 영역에서 향상되지 못하고 있다. 소프트웨어 개발을 성공적으로 완수하기 위해서는 일정, 비용, 품질, 범위가 만족되어야 한다. 품질은 기대보다 더 느리게 향상되고, 생산성은 최근 13% 떨어지고, 개발 프로젝트의 75%는 실패한 것으로 보고되고 있다[2,3]. 이와 같은 소프트웨어 위기 현상[4]에서 부터 출발한 체계적인 소프트웨어 개발의 필요성은 학문적으로 소프트웨어 개발 방법론 및 프로젝트 관리론 등의 발전을 가져왔다. 소프트웨어 개발 방법론은 IT 관리에서 가장 중요한 관심사이다[5,6,7,8,9].

소프트웨어 개발자는 구현된 소프트웨어를 빨리 만들어야 한다는 강한 압박을 받고 있다[10,11]. 애자일(Agile) 소프트웨어 개발 방법론은 "인터넷 시간"에 소프트웨어를 개발할 수 있는 더 가벼우며 빠르고 민첩한 소프트웨어 개발 프로세스에 대한 해답을 제공하기 위해 개발되었다[12,13,14,15]. 소프트웨어 개발 프로세스의 앞으로의 동향을 논의하는 2001년 모임에서 "애자일 소프트웨어 개발"의 가치와 목표를 정의하고, "애자일 연합 헌장"으로 12가지 공통된 원리를 제시하였다[16,17]. 애자일 연합에서는 애자일을 환경의 변화와 환경에 의해 부과된 요구에 신속하게, 적당하게, 채택하게, 반응하게 하는 능력으로 정의하였다[18].

애자일 방법론은 2008년 조사에 의하면 소프트웨어 개발 조직의 69%가 채택하여 사용하고 있는 방법론이다[19].

애자일 방법론 중에서 가장 처음 등장해서 가장 많이 사용되고 있는 방법론이 극한 프로그래밍(Extreme Programming:XP)이다[20,21,22]. XP는 켄트 백(Kent Back), 워드 커닝험(Ward Cunningham), 론 제프리(Ron Jeffries)가 1996년 다임러 크라이슬러의 급여시스템 개발 프로젝트를 수행하던 중에 새로운 소프트웨어 개발 방법을 체계적으로 정리하면서 탄생하게 되었다[23]. XP는 고객이 원하는 소프트웨어를 고객이 원하는 시기에 제공하는 것을 목표로 하며, 프로젝트 막바지에도 나올 수 있는 요구사항 변경에 더욱 잘 대처할 수 있는 특징이 있다[24,25].

소프트웨어 개발방법론은 조직 차원에서 도입되어, 개발 프로세스와 표기법 등을 표준화함으로서 ISO/IEC 12119을 기반으로 하는 ISO/IEC 15504(SPICE) 와 CMMI에서 정의한 것과 같은 조직의 성숙도를 향상시킬 수 있으며, ISO/IEC 9126과 ISO/IEC 14598 인증을 취득할 수 있는 고품질 소프트웨어를 획득하는 바탕이 된다.

서비스 기반 방법론, 컴포넌트 기반 방법론, 객체지향 방법론, 정보공학 방법론, 구조적 방법론과 같이 소프트웨어 실행 단위나 설계 단위 또는 문제 해결 기법과 같은 기술 요인에 따라 소프트웨어 방법론이 등장하였다. 소프트웨어 개발방법론의 채택을 위해서는 이와 같은 기술적인 특성과 사회심리학적 특성을 고려할 수 있으나, 소프트웨어 공학에서는 주로 기술 특성을 기준으로 채택해 왔다.

소프트웨어 개발방법론의 채택된다는 긍정적인 측면 이면에는 저항 요인이 있다. 개발자 입장에서는 익숙하지 않은 방법론, 관리자들 입장에서는 계획 및 일정 관리가 원활하게 제공되지 못하는 방법론, 고객 입장에서는 개발자가 유용성이 입증되지 않은 방법론을 사용하는 경우에 저항이 발생된다.

XP 방법론의 기술적인 유용성은 소프트웨어 개발 현장에서 채택하는 비율을 통해 짐작할 수 있지만, 개발자 입장에서는 XP를 어떻게 받아들이고 있으며, 교육자나 관리자는 어떤 점을 보완해야 하는가에 대한 연구가 필요하다.

본 논문에서는 점점 대중화되고 있는 XP 방법론의 도입 현황을 살펴본 후, XP의 "가치"와 "실천" 요소가 개발자들에게 어떻게 받아들여지고 있는지를 유용성과 이용용이성을 기준으로 조사하여, XP가 산업 현장에서 채택되기 위해 해결해야 할 이슈를 제시한다.

본 논문의 제2장에서는 소프트웨어 개발 방법론, XP의 "가치"와 "실천", XP의 채택에 대한 소개를 하고, 제3장에서는 XP의 현황과 "가치"와 "실천" 요소에 대한 실증 자료 분석을 하고, 제4장에서 결론을 내린다.

Ⅱ. 관련 연구

2.1 소프트웨어 개발 방법론

소프트웨어 개발 프로젝트에서 적절한 개발방법론의 선택은 항상 고민거리가 되어왔다[26]. 소프트웨어 개발 방법론은 13개 요소로 구성된다. 이들은 〈그림 1〉과 같이 역할, 기술, 팀, 기법, 활동, 프로세스, 작업 산출물, 이정표, 표준, 품질, 팀 가치, 개인성, 툴이다[16]. 방법론의 유형은 규범적인(Normative) 방법론, 이성적인(Rational) 방법론, 참여적인(Participative) 방법론, 휴리스틱(Heuristic) 방법론의 범주가 있다. 지식이 증가됨에 따라 방법론 영역은 휴리스틱 방법론에서 규범적인 방법론으로 옮겨지는데, XP를 포함한 대부분의 소프트웨어 개발은 휴리스틱 방법론 수준에 있다.

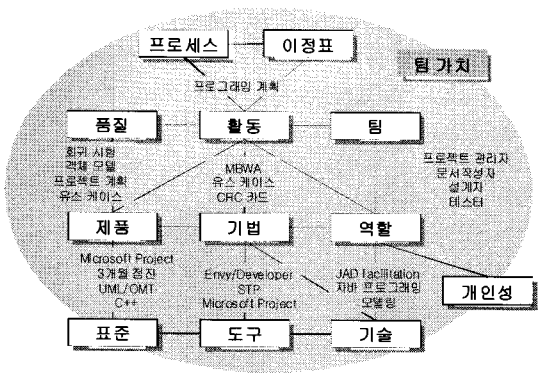

그림 1. 방법론 구성요소
Fig. 1. Elements of a methodology

2006년, 2007년에 이어 2008년 2월에 Dr. Dobb's Journal 메일링 리스트를 통해 자일 채택에 관한 조사된 바에 의하면, 애자일 방법론이 생산성에 도움이 됐다는 응답이 82%, 비용이 적게 들었다가 37%, 이해관계자의 만족도가 향상되었다가 78%가 되었다[19].

애자일 방법론으로는 Extreme Programming[22], Scrum[27], Crystal Mehods[28], Feature-Driven Development[29], Rational Unified Process[30], Dynamic Systems Development Method[31], Adaptive Software Development[32,33], Open Source Software Development[34], Agile Modeling[35], Pragmatic Programming[36] 등이 있다. 설문 조사에 의하면 응답자의 59%가 10여개의 애자일 개발 방법론 중에서 극한 프로그래밍(XP)을 가장 많이 사용하였다[37]. XP는 다양한 애자일 소프트웨어 개발 방법론의 최초 연구로 널리 알려져 있다.

2.2 XP

극한 프로그래밍(XP)는 의사소통(communication), 단순성(simplicity), 피드백(feedback), 용기(courage)의 4가지 "가치"를 기반으로 고객, 관리자, 프로그래머에 대한 역할과 권한을 중시하고, '고객에게 최고의 가치를 가장 빨리' 전달하도록 하는 경량 소프트웨어 개발 방법론이다[22,38]. 요구사항 등의 변화가 자주, 많이 있거나 개발자가 소규모(10명 내외)이고 같은 공간을 사용하는 경우에 높은 효과를 볼 수 있다고 알려져 있고, 다른 규모나 원거리에서 XP 적용을 꾸준히 시도하고 있다[39,40,41].

XP에서는 12개의 "실천"으로서 계획 게임(Planning Game), 작은 릴리즈(Small releases), 메타포(Metaphor), 단순 설계(Simple design), 테스팅(Testing), 리팩토링(Refactoring), 짝 프로그래밍(Pair programming), 공동 소유권(Collective ownership), 지속적 통합(Continuous integration), 주당 40시간(40 hour week), 고객 참여(On-site customer), 코딩 표준(Coding standards)을 사용하고 있다[23]

XP의 "실천" 기법은 팀 정신으로서의 "가치"를 중시하고 이를 기반으로 하고 있다. 즉, XP의 4대 "가치"는 12대 "실천"에 내재되어 있으며 "실천"이 활동하고 작용하기 위한 원동력이 된다. 이것은 마치 기업체의 각 부서가 근면, 성실, 솔선, 협동이라는 기업정신 또는 "가치"에 의해 유기적으로 움직이는 것과 비교할 수 있다. 경리부, 생산부, 자재구매부, 영업부, 총무부, 기술개발부 등 기업의 다양한 부서들이 기본적인 원동력 및 기업의 행동강령으로서의 기업정신 또는 "가치"를 바탕으로 상호 유기적으로 협동하면서 기업을 움직이기 위해 "실천" 기법을 적용한다.

XP는 이와 같은 상식차원의 "가치"와 "실천"을 원리로 하고, 극한(extreme)까지 실천하는 방법론이다. 본 논문에서 실증 대상이 되는 XP의 4대 "가치"와 12대 "실천"은 연구자들마다 다소간 차이가 있으나 공통적인 요소들을 간략히 정리하면 다음과 같다.

2.2.1 가치

① 의사소통(communication)
: 정확하고 구조적이며 적시의 의사소통은 구성원 누구나 프로젝트가 무엇에 관한 것이며, 무엇이 결과로 기대되고, 이러한 결과들을 얻기 위해 어떻게 진행되어야 할 것인지를 이해할 수 있도록 한다. 소프트웨어 개발은 인간 협동의 팀으로 실행되는 노동 집약적인 과정이며, 의사소통은 이들 팀이 효과적이면서 효율적으로 작동될 수 있도록 한다.

② 용기(courage to embrace change)
: 요구사항, 우선순위 및 디자인 등 무엇이건 간에 변경될 수 있고, 단일 프로젝트 개발 중에 많은 것이 변경될 수 있다. 따라서 개발 과정은 변화를 수용할 수 있어야 하며, 이를 통해 최적 가능한 결과들을 얻을 수 있다.

③ 피드백(feedback)
: 요구사항, 우선순위, 디자인 및 행위 과정에 영향을 미칠 수 있는 다른 조건들의 변화에 대응하여 커뮤니케이션이 전체 잠재적인 것들을 표출할 수 있도록 피드백 되어야 한다.

④ 단순성(simplicity)
: 소프트웨어 개발의 목표는 가능하게 작동될 수 있는 가장 간단한 해를 찾는 것이다. 변화들에 대한 충격 및 그에 대응하는데 필요한 노력이 최소화되도록 단순성을 추구해야 한다. 단순성은 효율성을 제공하게 된다.

2.2.2 실천

① **공동 소유권(collective ownership)**

: 팀의 모든 멤버들이 전체 생산물 및 테스트 코드를 소유할 수 있어야 한다는 것이다. 이러한 흐름은 팀 내에 모든 쌍이 언제든지 코드 부분을 테스트, 변경 혹은 갱신할 수 있도록 함으로써 조직원의 역할을 미리 정의된 범위로만 국한시키지 않도록 한다.

② **코딩 표준(coding standards)**

: 팀의 모든 멤버들이 둘러붙어서 총체적인 코드 소유권을 지지하고 이해성 및 유지보수성을 촉진시키도록 공통 코딩 표준을 사용한다.

③ **지속적 통합(continuous integration)**

: 지속적 통합을 통해 시스템이 항상 이용 가능함을 뜻한다. 이는 코드 테스트를 향상시키고 에러를 빨리 포착할 수 있도록 하며 더 나아가서 하나의 작업 버전이 검사 혹은 고객 양도를 위해 항상 이용 가능하도록 한다.

④ **고객 참여(on-site customer)**

: 이상적으로 고객에 의해 작성되거나 혹은 작성에 고객이 직접 참여되는 수락 테스트를 의미한다. 다른 테스트는 프로그래밍 팀에 의해 작성된다. 수락 테스트는 직접적으로 요구사항과 관계되며, 궁극적으로 프로젝트의 성공 및 실패를 결정한다.

⑤ **메타포(metaphor)**

: 관념의 일방적 고정이 아닌 를 통한 새로운 관념의 메타포 생성은 애플리케이션의 공통 비전이며 그것이 작동하는 방식을 나타낸다. 이러한 비전은 진화되며, 프로젝트 이해를 증진시키고 팀 생산성을 높이기 위해 전체 팀이 공동 노력해야 한다.

⑥ **짝 프로그래밍(pair programming)**

: 같은 컴퓨터에서 두 명의 프로그래머가 팀을 이뤄 작업하는 것이다. 두 사람 중에 한 사람은 키보드에서 실제 타이핑하고 다른 한 사람은 피드백을 한다. 이러한 방식에서 생성 코드는 연속적인 검사를 받게 되어 질과 생산성이 높아지게 된다.

⑦ **계획 게임(planning game)**

: 고객 참여의 비중을 상당히 높여서 고객에게 가치 제공이라는 기본적인 목표를 향해 프로젝트의 지속적인 관리가 이루어지도록 하는 단기 계획 생성의 절차이다. 절차 그 자체는 신중하게 단순화시켜서 요구사항의 변경 및 작업 우선순위의 투명하고 유연성이 있는 처리가 이루어지도록 한다.

⑧ **리팩토링(refactoring)**

: 외부적으로 관측 가능한 기능들은 변경하지 않고 빈번하게 사용되는 생성 코드를 재배열 및 재구성하여 추후에 설계의 이해 및 유지보수 가능성을 높일 수 있다[25].

⑨ **단순 설계(simple design)**

: 프로그래밍 팀은 기능요소를 빠른 시간에 제공할 수 있으며, 요구사항의 변경을 충분히 들어줄 수 있을 정도로 유연한 설계를 유지할 수 있다.

⑩ **작은 릴리즈(small release)**

: 요구사항들과 우선순위들의 변경을 보다 용이하게 하면서 가치가 증가된 소프트웨어를 고객에게 양도할 수 있도록 하며, 프로젝트가 지속적인 관리 상태에 있도록 하는데 도움이 된다.

⑪ **지속가능한 속도(sustained pace)**

: 개별 팀 구성원들이 장기적으로 항상 최적 상태를 유지할 수 있도록 주당 40시간 이내로 근무하게 한다. 가끔 초과 근무를 피할 수 없지만, 전체 작업 만족 및 생산성이 나빠질 수 있으므로 자주 일어나지 않도록 한다.

⑫ **테스팅(testing)**

: 코드 생성에 앞서 테스트 계획과 테스트 케이스가 작성되도록 하며, 소프트웨어 개발 활동을 테스트 주도로 반복되도록 한다.

2.3 XP 방법론의 채택

현재의 소프트웨어들은 아주 복잡해서 공동으로 작업하기 위한 조직이 필요하기 때문에, 소프트웨어 개발 방법론의 선택 및 도입은 개인 단위가 아닌 조직 단위에서 우선 검토된다[42]. 예를 들어, 구조적 방법론, 정보공학 방법론, 객체지향 방법론, 컴포넌트 기반 방법론, 애자일 방법론과 같은 대분류 중에서 애자일 방법론을 결정하고 여기서 XP를 선택한다. 방법론을 선택할 때는 방법론을 평가하게 되는데, 방법론의 기술적인 특성을 평가 기준으로 많이 사용해 왔다. 대부분의 논문들에 의하면 프로그래밍 언어 및 모델링 기술과 같은 진보에 초점을 맞추고 있다[43,44,45]. 방법론을 사용할 인적 요소의 중요성도 언급되고 있다[3,46]. 조직 단위에서 얼마나 빨리 인력을 교체하고 훈련시킬 수 있는지, 인력들에게 얼마나 많은 제약을 요구하는지, 상황 변화에 얼마나 빨리 응답할 수 있게 해주는지, 소송이나 다른 피해로부터 자체 조직을 얼마나 잘 방어할 수 있는지와 같은 5가지 방법론의 평가 요소를 사용하기도 한다[28].

조직 차원에서 방법론 도입을 결정할 때 조직원의 저항 및 부담을 최소화 하지 않으면, 방법론이 실무 현장에서 형식적으로 사용되어 생산성 저하와 같은 부작용이 발생할 수 있다. 사용자나 조직원이 방법론을 어떻게 받아들이는지에 관해서는 사회심리학적 바탕에서 연구 되었다[47,48]. 믿음, 태도, 의도 및 행위 이론을 기반으로[49], 계획 행동 이론[50], 지각된 유용성과 이용용이성[51], 사용자 인식 및 행동 충격[52]에 대한 연구, 이들의 통합 연구[25]가 있다. 정보기술을 수용에 관한 TAM(Technology Acceptance Model),

TAM2, PCI(Perceived Characteristics of Innovating), TPB(Theory of Planned Behavior), MPCU(Model of Personal Computer Utilization) 모델 각각을 현장에서 조사하여 중요 평가 요인을 밝혀내는 연구도 있었다[53].

전통적인 소프트웨어 개발방식은 효과 및 효율성 측면에서 점점 떨어지고 있다. 현재 비즈니스 환경에 대한 수요와 새로운 기술의 증가 속도는 프로젝트 진행에 있어서 요구사항들의 변경이 빈번해지고 있으며 변화에 부응하는 애자일 접근방식의 급진적인 문제의 해결 능력은 XP 가 많이 보급되는 원인이라 할 수 있다.

XP는 개발 초기에 문서를 이용하여 개념적으로 설계하고 차츰 다듬어가는 하향식 방법이 아니라, 실무적인 프로그래밍을 중심으로 진행하며 최소한의 결과를 문서화하는 상향식 개발을 한다[38]. 따라서 XP 방법론은 개발자, 관리자, 고객들로 이어지는 실무자 측면에서 선호하고 있다. 실무를 중요시 하다 보니 방법론으로서의 적절한 체계를 지키는 일도 중요하다[46].

소프트웨어 개발은 어떠한 훈련이 필요하며, XP 방법론 또한 예외가 아니다. XP는 엄밀한 규정 단계와 활동을 명시하지 않는다. 대신에 항상 궁극적인 "가치"를 유념하면서 많은 일상적인 업무들이 유연하게 "실천"될 수 있도록 한다.

XP 방법론은 〈그림 2〉와 같이 관련 이해 당사자 그룹들인 개발자, 관리자 및 고객들의 상호작용에 의해 방법론으로 채택된다. 개발자 입장에서는 익숙하지 않은 방법론, 관리자들 입장에서는 계획 및 일정 관리가 원활하게 제공되지 못하는 방법론, 고객 입장에서는 개발자가 유용성이 입증되지 않은 방법론을 사용하는 경우에 저항이 발생된다.

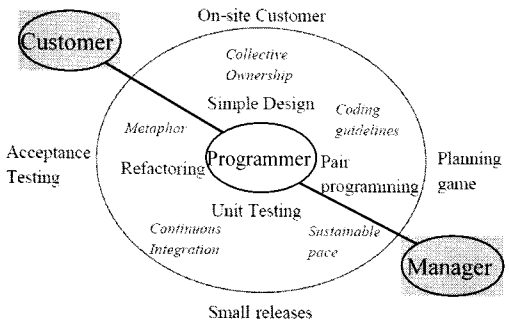

그림 2. XP 구성요소
Fig. 2. XP Elements

본 논문에서는 소프트웨어 개발자가 XP 방법론을 어떻게 받아들이고 있는지 XP의 "가치"와 "실천" 요소에 대한 유용성과 이용용이성을 조사한다.

III. 실증 연구

3.1 실험설계

기존의 방법론 채택에 관한 연구에 비해, 애자일 방법론, 특히 XP 방법론 자체의 "가치"와 "실천"을 중심으로 XP 방법론 도입에 중요한 사항을 조사하기 위한 문항과 대상자를 선정하였다. 온라인 환경에서 25 문항을 준비하여, 대학의 컴퓨터공학 전공의 졸업 예정자, 소프트웨어 개발업체에서 종사하고 있는 개발자, 관련 웹사이트 이용자 들을 대상으로 하였다. 온라인 문항에서 소개 문장들을 붙여서 이해하기 쉽도록 하였으며, 응답자 중에서 XP의 핵심적인 가치 및 소프트웨어 개발 실천 사항들에 대한 기본 지식을 갖추고 있는 것으로 가정된 사람들의 응답을 분석하였다. 분석 대상자는 대학 졸업 예정자 44명, 지역 소프트웨어 개발업체 종사자 27명, 웹사이트 링크를 통한 27명의 총 82명이 선정되었다. XP 방법론을 교육받은 대학생들을 참여시켜서, XP의 학습 과정에서의 어려운 점을 확인할 수 있도록 하였다.

응답자의 특성은 〈그림 3〉과 같다. 소프트웨어 개발 경험은 다소간 적은 쪽으로 치우친 경향이 있지만, 대략 10년 정도의 개발 경험자들이 40% 정도를 차지하고 있었다. 소프트웨어 개발 현장에서 응답자 역할 및 활동에 관한 다중 응답 결과를 살펴보면, 응답자의 73.2%가 프로그래밍에 치우쳐있지만, 그들의 역할이 팀 리더, 관리자의 역할을 중복하고 있고, 다른 아웃소싱업체에 고객으로도 참여하고 있었다.

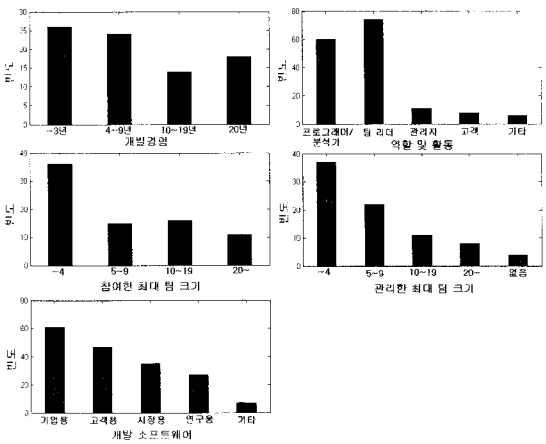

그림 3. 응답자 특성
Fig. 3. Observer's Characteristics

응답자들이 소프트웨어 개발 업무에 참여했던 최대 팀 크기에 관련하여 거의 절반이 5명 이내의 소규모 팀 운영에 참여한 경험을 가지고 있는 것으로 나타났다. 또한 응답자들이 관리해 본 최대 팀의 크기로는 응답자의 대략 1/4 정도가 10명 이내의 팀들을 관리해 본 경험이 있는 것으로 나타났다. 다중 응답으로 응답자들이 개발해본 소프트웨어 애플리케이션의 범위를 조사한 경과 다양하게 나타났으며, 대체적으로 조직 내에 사용되는 것, 알고 있는 고객을 위한 응용 프로그램, 시장 판매를 위한 애플리케이션뿐만 아니라 연구 및 학술적인 것들이 주류를 이루었다. 고객으로 참여한 더 많은 응답자들이 필요하지만, 대체적으로 표본들이 대표성을 가지는 것으로 확인되었다. 여러 가지 상황에서 소규모 팀 및 역할 중복은 소규모 소프트웨어 회사의 특성들이라 할 수 있다.

3.2 XP에 대한 경험

〈그림 4〉는 XP의 경험에 관한 것으로 XP에 대한 경험이 약간인 경우, 다양한 경우, 기타 애자일 방법에 대한 경험이 있는 경우, 경험이 전혀 없는 경우에 대한 조사 결과이다. 응답자의 3/4이상이 XP 관련 경험이 있는 것으로 나타났지만, 〈표 1〉과 같은 적합도 검정에 따르면, 이들 간에 편차는 대단히 심한 것으로 나타났다.

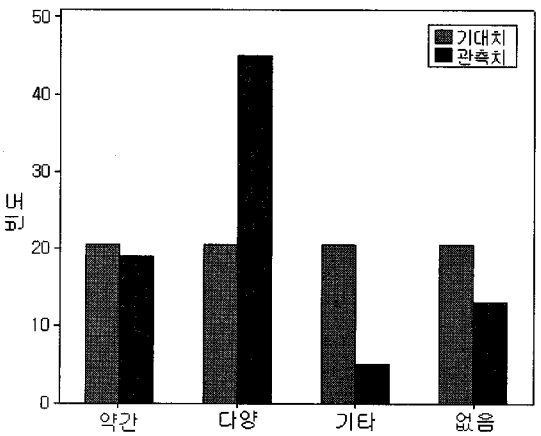

그림 4. XP에 대한 경험
Fig. 4. XP Experience

표 1. XP 경험에 대한 적합도 검정
Table 1 Chi-Square Goodness-of-Fit Test of XP Experience

Category	Observed	Test Proportion	Expected	Contribution to Chi-Square
약간	19	0.25	20.5	0.1098
다양	45	0.25	20.5	29.2805
기타	5	0.25	20.5	11.7195
없음	13	0.25	20.5	2.7439
N	DF	Chi-Square	P-Value	
82	3	43.8537	0.000	

XP가 어떤 프로젝트에 적절할 것인지에 관한 견해에 대해, 소규모의 탐사적인 프로젝트에 적절하다는 응답 비율이 대략 65%, 소규모의 루틴적인 프로젝트에 적절하다는 의견이 34%, 보통 규모의 프로젝트에 적절하다는 것이 38%, 큰 규모의 임무 중심 프로젝트에 적절하다는 견해가 61%, 유지보수에 적절하다는 것이 32%, 전혀 적절하지 않거나 관심이 없다는 경우가 대략 1/6 정도를 차지하고 있다. 〈표 2〉의 적합도 검정에 따르면, 이들 견해 간에 유의한 차이가 있는 것으로 나타나고 있다.

표 2. XP에 적합한 어플리케이션 규모에 대한 적합도 검정
Table 2 Chi-Square Goodness-of-Fit Test of XP Application Scale

Category	Observed	Test Proportion	Expected	Contribution to Chi-Square
관심없음	53	0.17	33.5	11.3507
소규모/탐사적	28	0.17	33.5	0.9030
큰규모/업무중심	31	0.17	33.5	0.1866
유지보수	50	0.17	33.5	8.1269
소규모/탐사적	26	0.17	33.5	1.6791
보통규모	13	0.17	33.5	12.5448
N	DF	Chi-Square	P-Value	
201	5	34.791	0.000	

〈그림 5〉는 각 범주의 적합도 검정의 카이제곱 값에 대한 기여도를 측정한 것으로 XP에 대한 무관심이 높고, XP에 대해 긍정적인 경우에는 소규모 탐사적인 프로젝트에 적절하다는 결과를 보이고 있다.

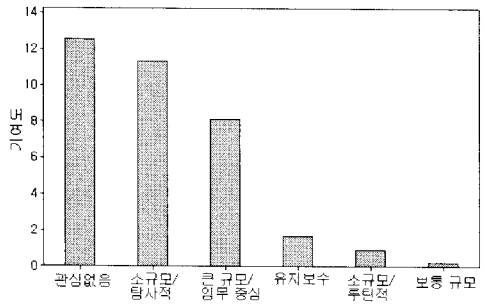

그림 5. XP의 기여도
Fig. 5. Appropriate Scale of XP Application

〈그림 6〉은 응답자가 소속된 각 조직의 XP 적용 여부에 관한 조사 결과로 이에 관한 응답은 적용과 계획 없음이 거의 비슷하게 나타났다.

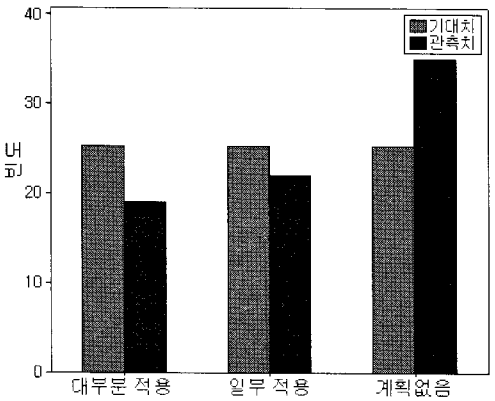

그림 6. XP의 적용 여부
Fig. 6. Status of XP Application

3.3 XP의 핵심 가치와 실천

XP의 4가지 "가치"와 12가지 "실천"을 구분하여 유용성과 이용용이성을 분석하였다.

〈그림 7〉은 XP 4가지 가치의 유용성 혹은 적절성에 관한 95% 신뢰구간이고, 〈그림 8〉은 XP 4가지 가치의 난이도에 관한 95% 신뢰구간이다. 0~4 까지의 5점 척도를 사용했으며 4점이 가장 유용 혹은 가장 어렵다는 의미이다.

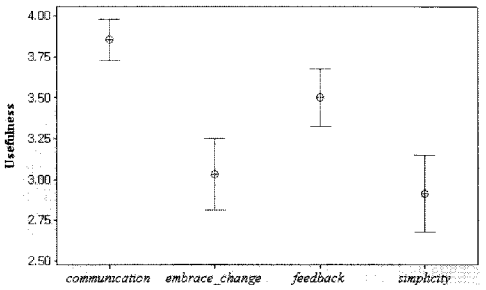

그림 7. 4개의 XP 가치에 대한 유용성
Fig. 7. Usefulness of 4 XP Values

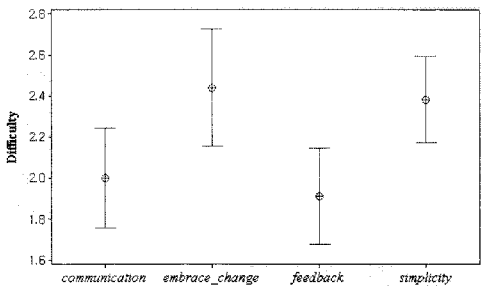

그림 8. 4개의 XP 가치에 대한 난이도
Fig. 8. Difficulty of 4 XP Values

4가지 변인들 간에 상호작용을 고려하지 않는 경우에 가장 적절한 유용한 원리는 "의사소통"인 것으로 나타났다. 모든 4가지 원리들은 주성분 인자분석 결과 〈표 3〉과 같이 모두 유의하며 76.8%의 설명력을 가지고 있었다. 4가지 요소들 간에 상호작용을 고려할 경우에 XP는 "용기"가 가장 유용한 원리라고 분석되었다.

표 3. XP 가치에 대한 주성분 분석
Table 3 Principal Component Factor Analysis of XP Values

Core Value	Communality of usefulness	Communality of difficulty
의사소통	0.474	0.884
용기	0.857	0.847
피드백	0.647	0.881
단순성	0.881	0.675
Variance : 6.1454		
% Var : 0.768		

XP "가치"의 달성 가능성은 "의사소통"과 "피드백"의 communality가 비교적 높게 나타나 이 부문의 달성이 보다 수월한 것으로 나타난다. 이들은 다른 두 가지 가치와는 달리 기술적인 종속성이 거의 없기 때문에, 심리학적인 원소들을 갖는 사회적 메커니즘에 의해 성취가능 하다고 할 수 있다. 이러한 관측은 성공적인 소프트웨어 개발을 위해 사회적 및 커뮤니케이션 스킬들을 강조함으로써 보다 더 많은 교육적/관리적 이득을 얻을수 있음을 뜻한다. 그렇지만, 기술에 더 많은 역점을 두고 있는 현행의 소프트웨어 공학 관점에서 매우 흥미로운 것이라 할 수 있다.

XP의 12개 실천에 대한 유용성을 분석하였다. 0~4 까지의 5점 척도를 사용한 결과 〈그림 9〉 와 같이 단순 설계, 테스팅 및 리팩토링이 유용한 것으로 나타났다. 특히, XP의 실천에 관한 12가지 변인들은 주성분 인자분석 결과 〈표 4〉와 같이 모두 유의하며, 80%의 높은 설명력을 가지고 있다.

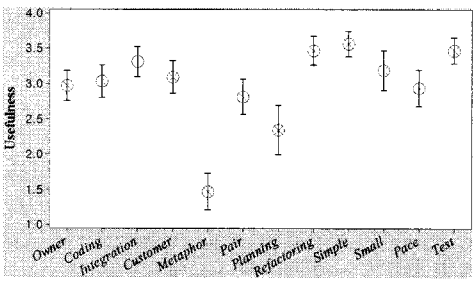

그림 9. 12개의 XP 실천에 대한 유용성
Fig. 9. Usefulness of 4 XP Practices

표 4. XP 실천에 대한 주성분 분석
Table 4 Principal Component Factor Analysis of XP Practices

Practices	Communality of usefulness
Collective Ownership	0.739
Coding Standards	0.801
Continuous Integration	0.758
On-site Customer	0.778
Metaphor	0.830
Pair Programming	0.807
Planning Game	0.905
Refactoring	0.782
Simple Design	0.724
Small Release	0.871
Sustained Pace	0.837
Testing	0.763
Variance : 9.5953	
% Var : 0.800	

"메타포"는 유용한 실천요소이지만, 〈그림 10〉에 의하면 메타포의 사용의 난이도는 어려운 편에 속한다. 이는 설계 범주의 대부분이 메타포 작업이어서 메타포 관련 의사결정은 보다 고급 프로그래머 혹은 인터페이스 설계자들에 의해 이루어질 가능성이 높기 때문이다. 메타포에 관한 많은 선택들은 윈도우, 맥 운영체제 혹은 여러 가지 리눅스 GUI와 같은 그래픽 사용자 인터페이스의 선호에 따라서 효과적으로 감소될 수 있다. 보다 적은 개수의 선택이 이루어질 수 있기 때문에 메타포는 용이하고 덜 중요한 업무로 비추어질 수 있다.

XP 실천에 관한 주성분 분석에 의하면 계획 게임이 매우 유용한 것으로 나타난다. "계획 게임" 은 고객과의 다양한 스토리 카드를 통한 계획 혹은 기획을 작성하는 것이다. XP 방법론에서는 고객을 개발팀의 일부로 배치하여 고객과 개발자 간의 끊임없는 의사소통을 중요시하며, 이러한 기획 단계의 효과가 프로젝트를 성공으로 이끈다고 볼 수 있다.

XP의 12개 실천 요소는 〈그림 10〉과 같이 대부분 달성하기가 용이한 편에 속한다. 계획 게임은 성취하기가 가장 용이한 것으로 나타나고 있다. 이는 연습의 단순성과 단기적인 짧은 것이라는 영향이 크다. 두 번째로 가장 용이한 것은 코딩 표준화 이며, 이것은 자동화된 툴을 사용함으로서 자연스럽게 성취할 수 있는 것이 주요 원인이라 할 수 있다.

4가지 가장 난해한 실천 요소인, 공동 소유권, 단순 설계, 고객 참여, 지속가능한 속도(주당 40시간)는 기술 보다는 사회적/관리적 이슈를 제기한다.

"공동 소유권"이 가장 어려운 실천 요소인 것으로 나타났다. 공동 소유권은 팀의 누구라도 코드를 개선하기 위해 변경할 권한을 가지지만, 누구나 모든 코드를 소유한다는 것은 다시 말하면 책임도 져야한다는 것을 의미하기 때문에 가장 어려운 것일 수 있다. 모든 코드는 모든 프로그래머들이 숙지하여 변경이 필요할 때에는 누구나 지체 없이 변경할 수 있어야

하며, 이는 짝 프로그래밍의 결과라고 할 수 있기 때문에 짝 프로그래밍도 다소간 어려운 편에 속한다.

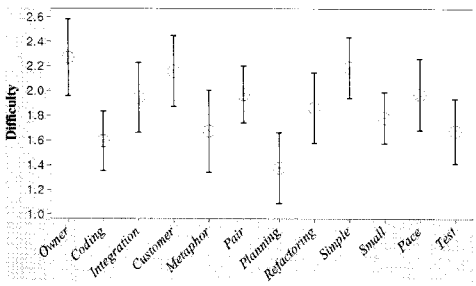

그림 10. 12개의 XP 실천에 대한 난이도
Fig. 10. Difficulty of 12 XP Practices

다음으로 달성이 어려운 것은 "단순 설계"로 이는 단순함의 가치를 나타낸다. 요구에 충족되는 가장 간단한 설계로 미래를 위해 많은 것을 투자하지 않으며, 현재의 비즈니스 가치에 집중하는 것이다. 내일 필요하게 될지도 모를 기능에 대해서 오늘 코딩하지 않는다는 것이 쉬운 것 같으나 프레임워크, 재사용가능한 컴포넌트, 유연한 디자인과 같은 것들과 연계될 때 더욱 필요하다. XP에서는 단순 설계를 보다 강조함으로써 설계 관련 이슈들을 개선할 수 있다[8].

"고객 참여"는 시스템 기능보다는 고객 자체에 집중하는 것이다. 인수 테스트 과정에서 고객과 개발자 간의 많은 문제점들이 발생한다. 기대치에 못 미치는 성능, 중요 기능의 미 구현, 기존 시스템과의 충돌, 만족스럽지 못한 문서화 등이다. 이런 문제점은 고객 참여와 체계적인 인수 테스트를 통하여 해결할 수 있다.

IV. 결론 및 시사점

본 연구에서는 XP 및 기타 애자일 방법론의 소프트웨어 개발 현황, 소프트웨어 개발자의 XP에 대한 인식을 조사함으로서 다음과 같은 결과를 획득하였다. 우선, 응답자들 가운데 극소수 개발자들만이 실제 XP를 사용하고 있지만 XP의 "가치"와 "실천"의 철학이 유용하다고 느끼고 있었다. 방법론적인 관점에서 XP는 소프트웨어 개발 과정의 품질과 적기를 개선시키는데 매우 성공적으로 받아들여지고 있었다. 두 번째, XP의 "가치"와 "실천" 요소들은 비교적 성취하기가 쉽지만 기술 부분에는 거의 종속되지 않으면서도 대단히 어려운 요소도 존재한다는 것이다. 이러한 것들은 사회적이고 관리 중심적인 혹은 설계 관련 논제들에 초점을 맞추고 있다. 이런 결과는 창의적이고 노동 집약적인 과정에 해당되는 소프트웨어 개발에 있어서 관리자 혹은 교육자들이 인간적인 특성에 보다 많은 비중을 두어야 한다는 것을 의미한다.

계속되는 연구를 통해 XP의 가치와 실천이 어떻게 교육되고 있으며 개발 조직에서 어떤 효과를 내는지에 대한 실증적인 연구와, XP를 도입한 개발조직의 성숙도의 변화에 대한 연구를 수행할 계획이다.

참고문헌

[1] Digital Economy 2000, U.S. Department of Commerce, Economics and Statistics Administration, http://www.ecommerce.gov, 2000.

[2] Glass, R. "The Realities of Software Technology Payoffs," Commmunication of the ACM, Vol. 42, No. 2, Feb. 1999. pp. 74-79.

[3] Whiting, R., "Development in Disarray," Software Magazine, Vol. 18, No. 12, p. 20, Sept. 1998., pp. 20.

[4] Gibbs, W., "Software's Chronic Crisis," Scientific American., pp. 86-95, Sept. 1994.

[5] Brancheau, J. Janz, B. and Wetherbe, J., "Key Issues in Information Systems Management: 1994-1995 SIM Delphi Results," MIS Quarterly, Vol. 20, No. 2, June 1996, pp. 225- 242.

[6] Dickson, G. Leitheiser, R. Wetherbe, J. and Nechis, M., "Key Information Systems Issues for the 1980's," MIS Quarterly, Vol. 8, No. 3, Sept. 1984, pp. 135-159.

[7] Hartog, C. and Herbert, M., "1985 Opinion Survey of MIS Managers: Key Issues," MIS Quarterly, Vol. 10, No. 4, Dec. 1986, pp. 351-361.

[8] Niederman, F. Brancheau, J. and Wetherbe, J., "Information System Management Issues for the 1990s," MIS Quarterly, Vol. 15, No. 4, Dec. 1991. pp. 475-500.

[9] Papazoglou, M. P. and Heuvel, W. V. D., "Business Process Development Life Cycle Methodology", Commnunications of the ACM, Vol. 50, No. 10, 2007. pp79-85.

[10] Aoyama, M., "Web-Based Agile software Development", IEEE Software, November/December, 1998.
[11] Cusumano, M. and Yoffie, D., "Software Development on Internet Time", IEEE Computer, Octorber, 1999.
[12] Fowler, M., The New Methodology, http://www.martinfowler.com/articles/newMethodology.html.
[13] Fowler, M., Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, Menlo Park, CA, 1999.
[14] Giga Information Group Inc.http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,69182,00. html.
[15] Shore, J. and Warden, S., The Art of Agile Development, O'Reilly, 2007.
[16] Agile Alliance Web Site : Manifesto for Agile software Development. http://agilemanifesto.org/.
[17] Martin R. C., Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002.
[18] Kruchten, P., "Agility with the RUP", Cutter IT Journal, Vol.14, No.12, 2001.
[19] Agile Adoption Rate Survey. February 2008, http://www.ambysoft.com/surveys/agileFebruary2008.html
[20] Abrahamsson, P. and et al., Agile Software Development Methods-Review and Analysis, VTT Publication 478, VTT, 2002.
[21] Abrahamsson, P. and et al., "New Directions on Agile Methods : A Comparative Analysis", ICSE '03, IEEE, 2003.
[22] Beck, K., "Embracing Change With Extreme Programming", IEEE Computer, Vol.32, No.10, 1999.
[23] Beck, K. and Andres, C., Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Boston, MA, 2nd edition, 2004.
[24] Larman, C., Agile and Iterative Development: A Manager"s Guide. Addison-Wesley Professional, Menlo Park, CA, 2003.
[25] XP-Extreme Programming : A gentle introduction, http://www.extremeprogramming.org.
[26] Mnkandla, E. and Dwolatzky, B., Agile Methodologies Selection Toolbox, ICSEA 2007, pp. 25-31.
[27] Schwaber, K. and Beedle, M., Agile Software Development with Scrum, Prentice-Hall, 2002,
[28] Cockburn, A., Agile Software Development. Addison-Wesley, Boston, MA, 2002.
[29] Palmer, S. R. and Felsing, J. M., A Practical Guide to Feature-Driven Development, Prentice-Hall, 2002.
[30] Jacobson, I., Booch, G., Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.
[31] Stapleton, J., Dynamic Systems Development Method -The Method in Practice, Addison Wesley, 1997.
[32] Highsmith, J. A., Adaptive Software Development : A Collaborative Approach to Managing Complex Systems. Dorset House Publishing, 2000.
[33] Highsmith, J. and Cockburn, A., "Agile software Development : The Business of Innovation", Computer, Vol.40, No.1, Sept. 2001. pp. 120-122.
[34] Hightower, R. and Lesiecki, N., Java Tools for Extreme Programming, Wiley Computer Publishing, 2002.
[35] Ambler, S., Agile Modeling : Effective Practices for Extreme Programming and Unified Process, John Wiley & Sons, New York, NY, 2002.
[36] Hunt, A. and Thomas, D., The Pragmatic Programmer, Addison Wesley, 2000.
[37] Agile Methodologies Survey Results. http://www.shinetech.com/agile_survey_results.jsp.
[38] Beck, K. and Fowler, M., Planning Extreme Programming, Addison-Wesley, 2000.
[39] Elssamadisy, A., Agile Adoption Patterns: A Roadmap to Organizational Success, Addison-Wesley, 2008.

[40] Jeffries, R., Anderson, A. and Hendrickson, C., Extreme Programming Installed, Addison-Wesley, 2000.

[41] Paulk, M. C., "Extreme Programming from a CMM Perspective". IEEE Software, Vol.18, No.6, Nov./Dec., 2001.

[42] Rattanasampan, W. and Kim, S. Y., "A framework to study technology use: Alternatives to technology acceptance model". In Proceedings of the Eighth Americas Conference on Information Systems AMCIS 2002, pages 883 - 889, Dallas, TX, Aug. 2002.

[43] Agarwal, R. and Prasad, J., "A field study of the adoption of software process innovations by information systems professionals", IEEE Transactions on Engineering Management, Vol. 47, No. 3, Aug. 2000. pp. 295 - 308.

[44] Fichman, R. G. and Kemerer, C. F., "Object technology and reuse: lessons from early adopters", Computer, Vol. 30, No. 10, Oct. 1997. pp. 47 - 59.

[45] Sultan, F. and Chan. L., "The adoption of new technology: The case of object-oriented computing in software companies". IEEE Transactions on Engineering Management, Vol. 47, No. 1, Feb. 2000, pp. 106 - 126.

[46] Boehm, B. and Turner, R., Balancing Agility and Discipline:A Guide for the Perplexed, Addison-Wesley, Boston, MA, 2004.

[47] Turner, R., "People Factors in Software Management :Lessons from Comparing Agile and Plan-Driven Methods", Journal of Defense Software Engineering, Dec. 2003, pp.4-8.

[48] Venkatesh, V. Morris, M. G. Davis, G. B. and Davis. F. D., "User acceptance of information technology: Toward a unified view", MIS Quarterly, Vol. 27, No. 3, Sept. 2003, pp. 425 - 478.

[49] Fishbein, M. and Ajzen. I., Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research. Addison-Wesley, Reading, MA, 1975.

[50] Ajzen, I., "The theory of planned behavior", Organizational Behavior and Human Decision Processes, Vol. 50, 1991. pp. 179 - 211.

[51] Davis, F. D., "Perceived usefulness, perceived ease of use, and user acceptance of information technology", MIS Quarterly, Vol. 13, Sept. 1989, 319 - 340.

[52] Davis, F. D., "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts", International Journal of Man-Machine Studies, Vol. 8, 1993. pp. 475 - 487.

[53] Riemenschneider, C. K. Hardgrave, B. C., and Davis, F. D., "Explaining software developer acceptance of methodologies: A comparison of five theoretical models", IEEE Transactions on Software Engineering, Vol. 28, No. 12, Dec. 2002, pp. 1135 - 1145.

저 자 소 개

이 상 현

2002년 호남대학교 컴퓨터공학과 공학사
2004년 호남대학교 컴퓨터공학과 공학석사
2007년 전남대학교 전산학과 박사수료
2004년~2005년 광주여자대학교 시간강사
2006년 ~ 현재 호남대학교 시간강사
〈관심분야〉 인공지능, 데이터마이닝, 소프트웨어공학

이 상 준

1991년 전남대학교 전산통계학과 이학사
1993년 전남대학교 전산통계학과 이학석사
1999년 전남대학교 전산통계학과 이학박사
1995년~2005년 서남대학교 경영전산정보학과 조교수
2005년~2007년 신경대학교 인터넷정보통신학과 조교수
2007년 ~ 현재 : 전남대학교 경영학과 조교수
〈관심분야〉 소프트웨어공학, 경영정보시스템, 서비스 사이언스