사용성 향상을 위한 극한 프로그래밍 프로세스 개선

이상준*

Enhancement of Extreme Programming Process for Usability Improvement

Sang-jun Lee*

이 논문은 2003년도 정보통신기초기술연구지원사업(1-2003-A1-2000-0249) 지원으로 수행되었음

요 약

극한 프로그래밍은 빠르게 변화되는 사업 환경에 기민하게 대응하는 애자일 소프트웨어 개발 방법론 중에서 가장 대표적인 방법론이다. 소프트웨어 산업이 성숙됨에 따라 소프트웨어 품질 요소중 사용성이 점차 강조되고 있으나, 극한 프로그래밍에서 사용성을 확보하기 위한 노력이 부족한 실정이다. 극한 프로그래밍의 사용성 향상을 위하여 본 논문에서는 소프트웨어 생명주기를 조사, 계획, 릴리즈 반복, 제품화, 유지보수, 사망의 6단계로 정의한다. 사용성 평가를 위한 통합시험과 인수시험을 포함한 11개의 활동을 정의하여 사용성을 향상시킬수 있는 극한 프로그래밍 프로세스를 제안한다. 제안한 프로세스의 타당성을 4가지 측면에서 확인하였다. 제안한 프로세스는 극한 프로그래밍 프로세스가 인수시험 활동을 지원하도록 개선하였으며, 극한 프로그래밍 프로세스가 CMM 레벨4를 지원하도록 하며, CMM 품질보증 실무를 지원하도록 하였다.

ABSTRACT

Extreme programming is the most representative methodology among agile software development methodologies which is agile in business environment which change fast. As software industry is matured, usability of software quality characteristics is emphasized gradually, but effort to obtain usability in extreme programming is insufficient. To improve usability of extreme programming, in this thesis, a software life cycle is defined by six phases as exploration, planning, iterations to release, productionizing, maintenance, and death phase. In this thesis, an extreme programming process which can improve usability is proposed by definition of 11 activities including integration testing and acceptance testing for usability testing, The proposed process is validated with four aspects. The proposed process is enhanced to support acceptance testing activity. The proposed extreme programming process is able to support CMM level 4 and quality assurance practice of CMM.

키워드

소프트웨어 개발 프로세스(Software Development Process), 소프트웨어 사용성(Software Usability), 애자일 소프트웨어 개발(Agile Software Development), 극한 프로그래밍(Extreme Programming)

Ⅰ. 서 론

소프트웨어 개발에 관련된 사용자들은 자신들이 무엇을 원하는지 정확히 모를 뿐만 아니라 요구사항을 한 번에 모두 이야기하지도 않고 자주 마음을 바꾸기도 한다. 이런 문제를 해결하고자

* 서남대학교 컴퓨터정보통신학과

접수일자 : 2003. 9. 18

애자일(Agile) 소프트웨어 개발 방법론이 등장하게 되었다[8,11].

극한 프로그램(XP: eXtreme Programming)은 반복형 모델의 개발주기를 극단적으로 짧게 함으로써, 프로그래머가 설계, 구현, 시험 활동을 전체 소프트웨어 개발 기간에 걸쳐 조금씩 자주 실시하도록 하고 소프트웨어 변경의 비용을 최소화하는 애자일 기법이다[22].

소프트웨어 산업이 성숙될수록 단순한 개발보다는 제품의 품질을 강조하는 것과 같이, 소프트웨어 품질 특성 관점에서는 기능성 중심에서 신뢰성과 사용성을 점차 강조하고 있고, 점점 더 많은 조직이 사용성을 중요하게 받아들이기 시작했다[17]. 보통 개발자는 사용성 개념을 채택하지 않아서, 소프트웨어 제품의 사용성 레벨이 향상되지 못하고 있다[10].

소프트웨어 사용성 향상을 위하여 사후처리 대신에 개발 프로세스의 핵심 컴포넌트로 사용자 중심의 설계 및 평가를 도입할 수 있다. 사용성 중심의 개발이란 초기 설계 향상과 재작업을 줄이도록 가능한 개발 단계의 시작 부분에 사용성을 확보하도록 하는 것이다. 보호활동 프로세스로 사용성을 제공하는 것보다는 사용성 평가을 소프트웨어 개발 프로세스에 완벽하게 통합해야 좋다[4].

본 논문은 XP가 갖는 사용성 관련 문제를 식별하고, 소프트웨어 개발 프로세스를 보강하여 그 문제를 해결하도록 연구하였다. 기존의 XP 연구에서는 사용성에 대한 연구사례가 없고, XP 프로젝트의 활동에서 사용성에 대한 고려가 없다.

XP의 사용성 향상을 위하여 본 논문에서는 소프트웨어 생명주기를 조사, 계획, 릴리즈 반복, 제품화, 유지보수, 사망의 6단계로 정의한다. 사용성 평가를 위한 통합시험, 인수시험을 포함한 11개의 활동을 정의하여 사용성을 향상시킬수 있는 극한 프로그래밍 프로세스를 제안한다.

논문의 2장에서는 소프트웨어 프로세스, 극한 프로그래밍, 사용성 평가의 관련 연구를 소개하고, 3장에서는 사용성 향상을 위한 극한 프로그래밍 프로세스를 제시한다. 4장에서는 제안한 프로세스의 타당성을 5장에서는 결론을 맺는다.

II. 관련연구

1. 소프트웨어 프로세스

소프트웨어 프로세스란 고품질의 소프트웨어를 구축하는데 요구되는 태스크에 대한 프레임워크로 정의한다. 그림 1과 같이 공통 프로세스 프레임워크는 그것들의 크기 혹은 복잡도에 관계없이 모든 소프트웨어 프로젝트에 적용할 수 있는 수많은 작은 프레임워크 활동들을 정의해서 성립된다. 프레임워크 활동들이 소프트웨어 특성들과 프로젝트팀의 요구 사항들에 적용될 수 있도록 태스크와 프로젝트 이정표, 소프트웨어 작업 제품과 산출물, 품질 보증 포인터들을 태스크 집합으로 구성한다[21].

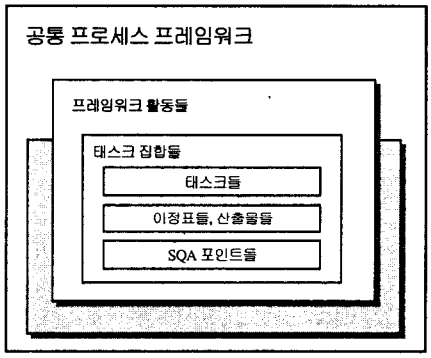

그림 1. 소프트웨어 프로세스
Fig. 1 Software Process

소프트웨어 프로세스에 대한 일반적 모델로 폭포 모델, 프로토타이핑 모델, RAD 모델, 단계적 소프트웨어 프로세스 모델(점진적 모델, 나선형 모델, 컴포넌트 어셈블리 모델, 컨커런트 개발 모델) 등이 있다[21]. 본 논문에서는 극한 프로그래밍에 적합한 프로세스를 개발하고자 한다.

2. 극한 프로그래밍

소프트웨어 개발 프로세스의 앞으로의 동향을 논의하는 2001년 모임에서 "애자일 소프트웨어 개발"의 가치와 목표를 정의하고, "애자일 연합헌장"으로 12가지 공통된 원리를 제시하였다[2].

2002년 11월부터 2003년 1월 동안에 Shine Technology에서 애자일 방법론에 대한 시장 관심을 측정하기 위해 웹기반 조사를 실시하였다. 10개의 질문에 정확하게 응답한 131개의 타당한 결과를 분석한 결과 애자일 방법론이 생산성에 도움이 됐다는 응답이 93%, 품질이 향상되었다는 응답이 88%, 비용이 적게들었다가 49%, 사업적 만족도 83%, 2003년에 적용할 것인가에 94.7%로 응답하였다[3].

지난 2002년 3월 Giga Information Group Inc.는 응용 소프트웨어 개발 컨퍼런스에서 18개월 이내에 전세계 IT 기업의 2/3 이상이 일종의 애자일 방법론을 사용하게 될 것이라고 예측했다[12].

애자일 방법론으로는 Extreme Programming, Scrum, Crystal Mehods, Feature-Driven Development, Rational Unified Process, Dynamic Systems Development Method, Adaptive Software Development, Open Source Software Development, Agile Modeling, Pragmatic Programming 등이 있다.

설문 조사에 의하면 응답자의 59%가 10여개의 애자일 개발 방법론중에서 극한 프로그래밍(XP)을 가장 많이 사용하였다[2]. XP는 다양한 애자일 소프트웨어 개발 방법론의 최초 연구로 널리 알려져 있다.

XP는 의사소통, 피드백, 단순성, 용기의 4가지 가치를 기반으로 고객, 관리자, 프로그래머에 대한 역할과 권한을 중시하고, "고객에게 최고의 가치를 가장 빨리" 전달하도록 하는 경량 소프트웨어 개발 방법론이다. XP에서는 12개의 실무로서 계획, 작은 릴리즈, 메타포, 단순 설계, 리팩토링, 테스팅, 짝 프로그래밍, 공동 소유권, 지속적 통합, 주당 40시간, 작업에 고객 동참, 코딩 표준을 사용하고 있다[5,6].

3. 소프트웨어 사용성

정보화 사회로의 전환이 급속하게 이루어지면서 사회적으로 컴퓨터 및 소프트웨어에 대한 의존도도 크게 증가하고 있다. 인터넷과 정보화 사회의 발전에 따라 소프트웨어 개발 분야에서는 높은 생산성과 품질의 향상을 가장 중요한 과제로 여기고 있다.

소프트웨어 제품 자체의 품질은 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성의 6가지 특성을 바탕으로 평가할 수 있다[15]. ISO 9241-11에 따르면 사용성은 제품이 특정 사용자에 의해 특정 사용 문맥에서 효과성, 효율성, 만족도의 특정 목표에 성취하는 정도이다[14]. 인간 공학의 관점에서는 효율성과 효과성도 사용성의 구성 요소로 간주한다. 사용성은 학습 용이성, 사용 효율성, 기억 용이성, 최소한의 에러를 포함하며, 넓게는 미학적 구성에 의해 상당 부분 영향을 받는 주관적 즐거움 까지도 포함한다[9,19]. 좋은 소프트웨어가 되기 위해서는 익히기 쉬워야 하고, 한번 익혀진 것은 높은 사용 효율을 발휘해야 하며, 시간이 흐른 뒤에도 기억하기 쉽고, 사용자의 실수를 유발하지 않아야 하는 사용성이 강조된다.

사용성을 평가하는 방법으로는 사용자 관찰, 성능 측정, 질문지, 생각 크게 말하기, 모델기반 방법, 전문가 평가등 여러 방법이 있다. 적용단계, 스타일, 객관적 유무, 측정값 유형, 정보 수준, 응답의 긴박, 사용자 행동, 시간 소비, 장비 필요, 전문성 수준에 따라 사용성 평가 방법을 선택하여 사용할 수 있다[9].

ISO/TR 16982에서의 사용성 평가 방법은[16] 어떤 생명주기 프로세스에서 사용하는 평가 방법인지, 프로젝트 환경 제약사항으로는 어떤 것이 있는가를 제1 선택 기준으로 삼는다. 제1 선택기준을 우선 고려한 후 사용자 특성, 태스크 특성, 제품 특성에 따른 제2 선택기준에 따라 사용성 평가 방법을 선택한다.

사용성 평가를 소프트웨어 개발 프로세스에 완벽하게 통합함으로서 자연스럽게 사용성 향상을 가져올 수 있다.

III. 사용성 향상을 위한 극한 프로그래밍 프로세스

1. 극한 프로그래밍의 문제점

기존의 XP 연구에서는 소프트웨어 사용성 관점에서 다음 네 가지 문제점을 갖고 있다.

- 방법론을 구성하는 역할 영역 중 사용자 인터페이스 설계자의 역할이 정의되지 않았다.
- XP에서의 사용성에 대한 연구사례가 없다.
- CMM의 관리 및 하부구조 지원 프로세스를 지원하지 못하고 있다.
- XP 프로젝트의 활동에서 사용성에 대한 고려가 없다.

XP는 방법론의 범위로 볼 때 사용자 인터페이스 설계자의 역할 영역이 정의되어 있지 않음으로 사용성에 대한 고려가 없는 실정이다[8].

XP와 애자일 개발 방법론을 제창했던 Beck은 소프트웨어 개발의 4가지 변인으로 비용, 시간, 품질, 범위를 소개하였다[6]. 소프트웨어 개발 현장의 선임 관리자는 고객 만족도 향상의 필요성을 인식하고, 사용성 실무를 주요한 수단으로 도입하고 있으나[4], XP에서는 소프트웨어 품질특성중 기능성 중심으로 품질을 확보하는 노력을 할뿐, 신뢰성, 사용성, 효율성, 유지보수성, 이식성에 대한 고려가 구체적이지 못하고 미비하다. 소프트웨어 개발을 개발자만이 수행하는 것이 아니라 프로젝트 초기부터 사용자를 참여시키자는 실무기법의 특성을 갖는 XP에서 사용성은 중요하게 다뤄야 할 부분임에도 불구하고 XP에서의 사용성에 대해 연구된 사례가 없다.

XP가 단순한 프로그래밍 기법 이상의 소프트웨어 개발 패러다임으로 존재하기 위해서는 프로세스에 대한 정의가 있어야 하지만, XP의 많은 연구자들이 합의한 프로세스는 없고, 4개의 가치와 12개의 실무기법만을 정의하여 이를 바탕으로 프로세스를 정의하여 사용하도록 하고 있다. 소프트웨어 프로세스 품질을 성숙도로 나타내는 CMM의 핵심 프로세스 영역(KPA:Key Process Area)들을 XP가 만족하기 위해서는 관리 및 하부구조의 지원이 필요하다. XP와 관련된 많은 책과 논문과 코스, 웹사이트에는 부분적으로 CMM의 조직 프로세스 정의와 훈련 프로그램등의 KPA를 삽입하여 보완하는 연구를 수행하고 있다[20].

2. 사용성 향상을 위한 극한 프로그래밍 프로세스 제안

그림 1과 같이 소프트웨어 프로세스는 여러개의 활동, 태스크, 산출물 등으로 구성된다. 본 논문에서는 사용성 향상을 위한 극한 프로그래밍 프로세스를 그림 2와 같이 제안한다. 소프트웨어 개발 단계, 단계별 활동, 단계별 산출물을 정의하였다.

전체 생명주기는 조사단계, 계획단계, 릴리즈 반복 단계, 제품화 단계, 유지보수 단계, 사망 단계의 6 단계로 구성되었다. 조사단계에서는 개발 준비, 사용성 요구분석, 사용자 스토리 작성의 활동을 수행하여 스토리 산출물을 생성한다. 계획단계에서는 계획수립 활동을 수행하여 계획서 산출물을 생성한다. 릴리즈 반복 단계에서는 테스팅 계획 작성, 분석, 설계, 짝 프로그래밍, 단위테스팅, 통합시험, 인수시험을 수행하며, 분석서, 설계서, 코드, 테스팅 계획서, 테스팅 결과서 산출물을 생성한다.

그림 2에 음영으로 표시된 사용성 향상을 위한 활동은 사용성 전문가가 특별히 다루는 부분이며, 본 논문에서는 이 부분은 자세히 정의하고, XP에 대한 참고문헌를 통해 파악할 수 있는 보편적인 개발 활동들은 간단히 소개하고자 한다.

조사 단계에서 개발자는 소프트웨어 개발에 사용될 도구, 기술, 실무에 친숙하도록 개발 준비한다. 고객은 자신이 원하는 소프트웨어에 대한 요구사항을 사용자 스토리라는 카드에 작성한다. 사용성 전문가는 희망하는 학습시간, 실행시간, 사용성 정도를 스토리 카드에 명시하도록 한다.

계획 단계에서는 개발자, 고객, 사용성 전문가가 함께 모여 사용자 스토리의 우선순위를 결정하고, 각 릴리즈에 포함될 내용을 선정한다. 개발자는 독립적으로 각각의 사용자 스토리를 소프트웨어로 구현하기 위해 얼마나 많은 노력이 필요한가 예상하고, 개발 비용과 일정을 계획서에 명시한다.

릴리즈를 위한 반복 단계에서는 계획서에 근거하여 계획을 나누고 각각이 1주에서 4주내에 구현될 수 있도록 한다. 일반적인 소프트웨어 개발 과정과 마찬가지로, 분석, 설계, 구현, 테스팅 활동이 필요하고, 특히 우선적으로 테스팅 계획을 수립하는 일이 선행되도록 한다. 특정 버전의 릴리즈를 생성하기 위해서는 몇 번의 반복이 있게되

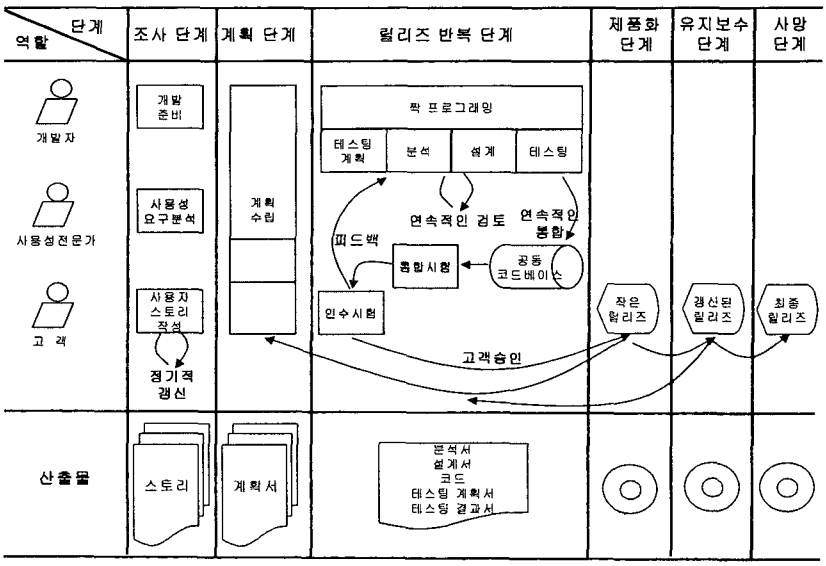

그림 2. 제안한 극한 프로그래밍 프로세스

Fig. 2 Proposed Process of Extreme Programming

며, 각 반복의 끝에는 기능시험을 수행한다. 본 논문에서는 기능시험외에 사용성 시험의 필요성을 인식하고, 통합시험에서 GOMS 평가, 인수시험에서 설문지 평가와 휴리스틱 평가를 수행하여, 계획서상의 사용성이 확보되도록 한다.

많은 종류의 사용성 평가 방법은 특정 방법 1개만을 사용해도 되고, 각 평가 방법의 단점을 보완하기 위해 몇 개의 평가 방법을 같이 사용할 수 있다. 참고문헌 [9,16]에 소개된 사용성 평가 방법 중에서 XP에 적용하기 부적합한 방법들이 있다. XP가 소프트웨어 개발 생명주기에 사용가능해야 함으로, 치명적인 사건분석 방법, 창조적 방법은 사용할 수 없고, 프로젝트 환경 제약사항중 전문적인 사용성 평가 실험실을 마련하기 위한 비용과 시간이 많이 드는 사용자 관찰 방법, 성능 측정 방법, 생각 크게 말하기 방법은 제외될 수 있으며, XP는 가능한한 문서화를 하지 않기 때문에 문서기반 방법을 사용할 수 없다.

본 논문에서는 사용성 평가 방법중 학문적으로 가장 정리가 잘 되어있으며 사용성 평가 결과를 이용하여 설계를 수정할 수 있는 가이드라인까지 제공하는 GOMS를 이용한 모델 기반 방법을 통합시험에 도입하고, 사용자의 느낌을 가장 쉽게 평가할 수 있는 질문지 방법과 전문가에 의해 숨겨진 문제점까지 찾아내올 수 있는 휴리스틱 평가 방법을 인수시험에 도입하고자 한다.

통합시험에 적용하는 GOMS(Goals, Operators, Methods, Selection rules)란 목표, 조작자, 방법 및 선택 규칙에 기초를 두고 컴퓨터 시스템과 함께 업무를 수행할 때의 사용자 지식과 작업을 기술하는 것이다. 이 방법은 전문가를 참가자 대상으로 하며 수행능력과 수행시간을 정밀하게 검사한다. 논리적인 수행 과정을 검증하고자 하거나 반복적인 사용 시간을 단축시키는 것이 목적일 경우에 매우 유용한 방법이다[18].

GOMS 모델을 구축 절차는 다음과 같다.

Step A : 최상위 레벨 사용자 목표 선택
Step B : 다음 반복 절차를 수행 :
B1. 각각의 목표를 수행하기 위한 메소드 초안 작성
B2. 초안을 완성한후, 일관성과 가이드라인에 대한 일치에 대한 요구를 이용하여 점검하고 검토
B3. 필요하다면, 고수준 연산자를 수행-목표 연산자로 변경함으로서 분석의 낮은 수준으로 이동하고, 적합한 목표에 대한 메소드를 제공
Step C : 분석을 문서화하고 점검
Step D : 판정과 가정에 대한 민감성 점검

GOMS 모델을 구축한 간단한 부분적인 예제는 다음과 같다.

```
Method for goal : Input 우편번호
        Step 1. Find Location
        Step 2. Move Cursor to 우편번호
        Step 3. Wait for System Response
                우편번호 자동 검색
        Step 4. Home-hand-to Keyboard
        Step 5. Type Keyboard 동 이름
        Step 6. Move Cursor to 동 선택
        Step 7. Home-hand-to mouse
        Step 8. RGA
Method for goal : Input 상세주소
        Step 1. Home-hand-to Keyboard
        Step 2. Retrieve Long-term Memory
        Step 3. Type keyboard 45 character
        Step 4. Forget working memory
        Step 5. Home-hand-to mouse
        Step 6. RGA
```

GOMS 방법에 인지복잡도 이론을 근거로 한 NGOMSL(Natural GOMS Language)이 있다. NGOMSL은 모형 구축에 필요한 규칙과 용어를 단순화하고, 모형 구축을 용이하게 하며, 사용성 관련 자료를 수치적으로 예측할 수 있도록 하였다. NGOMSL에 의한 예측 결과를 이용하여 설계를 수정할 수 있고, 수정한 후 성능예측을 다시 계산할 수 있다. GOMS 방법은 다른 사용성 평가 방법에 비해 사용성을 향상시킬수 있는 7개의 가이드라인과 설계의 정성적 평가를 제공한다. 원시 연산자를 이용하여 예상되는 실행시간과 사용자 학습 시간을 간단한 공식을 이용하여 계산 가능하다[18].

인수시험에 적용하는 **질문지 이용 평가 방법**은 사람들이 가지고 있는 일반적인 생각과 그 변화를 이해하는데 효율적이어서 시스템이나 서비스에 대한 사용자들의 만족도, 불편사항, 건의사항 등을 파악할 때 가장 많이 사용되는 방법이다. 질문조사의 결과는 중요한 요인 항목을 찾아내고, 그 항목들간의 상관관계를 밝히는 것 이상의 세부적인 사용성 정보를 주지 못하는 문제점도 있다. 하지만 개발에 참여하는 사용자에게 사용하기에 편리한가라는 개괄적이고 단편적인 질의 응답보다는, 척도가 있어야 상태를 알 수 있는 품질 평가의 기본 특성을 구체적인 질문으로 정량적으로 사용성을 평가할 수 있기에 질문지를 이용하고자 한다. 특히, 개발과정의 요구분석 단계와 자격시험, 고품질 레벨이 요구되는 평가에 우수한 방법이다.

질문지 이용 평가 방법의 간단한 부분적인 예는 다음과 같다.

```
[문항1] 사용하기가 쉬운가?
[문항2] 사용법을 숙달하기가 쉬운지?
[문항3] 화면 메뉴의 구성이 논리적인가?
[문항4] 화면 순서들이 명확한가?
[문항5] 다음에 나올 화면이 예측 가능한가?
[문항6] 메뉴, 지시문의 위치가 일관적인가?
[문항7] 문서의 내용 및 구조가 간결, 명확한가?
[문항8] 숙달할 때까지 시간이 적게 걸리는가?
[문항9] 조작법 또는 명령어를 기억하기가 쉬운지?
[문항10] 과제수행에 걸리는 단계수가 최대한 적은지?
[문항11] 적절한 도움말을 찾기가 쉬운가?
[문항12] 에러 메시지가 도움을 주는가?
[문항13] 실수를 교정하기가 간단한가?
```

인수시험에 적용하는 **휴리스틱 평가**는 주로 4-5명 정도의 소수의 평가 전문가가 사용성 평가 대상 시스템(또는 시제품)에 대한 사용평가를 수행하여 그 인터페이스가 사용성 원칙에 따르는지 등의 문제점을 도출한 후, 모여서 수합하는 방식으로 진행된다. 일반적으로 평가자가 많을수록 더 많은 사용성 문제를 탐지하지만 그 부가적 효율성은 5명 이상이 되면 급감한다.

휴리스틱 평가는 UI 전문가들이 UI의 각 요소가 확립된 사용성 원칙을 따르고 있는지 판단할 수 있는 사용성 검증법 중 하나이다. 각 프로젝트에 맞게 구성된 전문가 집단은 휴리스틱 평가 항목에 의해 가시적인 평가를 하고 이것을 수치적으로 나타날 수 있다. 전문가의 검증이 필요한 기초분석단계 또는 프로젝트 수행 마지막 단계에 효과적이다. 휴리스틱 평가 방법을 제안한 Nielson의 10개 평가 항목은 참고문헌 [19]에 제시되어 있다.

휴리스틱 평가 방법의 간단한 부분적인 예는 다음과 같다.

```
[문항1] 사용자 통제와 자유: 원하는 기능을 선택하기
        곤란했거나, 잘못 선택했을 때 뒤로가기를 하
        기 불편한 점이 있었나요? 백(back), 홈
        (home)버튼 등이 필요한 곳에 있나요?
[문항2] 일관성과 표준: 디자인이 일반적 관행과 표준
```

을 따르지 않고 있는 점이 있나요? 문서의 배경, 구조, 아이콘, 메뉴 등이 일관적으로 구성되지 못한 점이 있나요? 전체적으로 모든 문서들이 통일된 느낌을 주고 있나요?

[문항3] 오류방지: 당신이 실수를 범하기 쉬웠나요? 실수하지 않도록 하는 디자인의 노력이 보이나요? 양식을 잘못 입력하지 않도록 하는 기능이 들어있나요? 마우스를 잘못 누르지 않도록 세심하게 디자인 되어있나요?

제품화 단계에서는 릴리즈된 소프트웨어가 실제 사용 현장에서 문제 없는지 추가적인 테스팅을 수행하고 문제가 없을시 수행가능한 형태의 릴리즈가 가능하다. 지연된 아이디어나 제안을 추후 구현을 위해 문서화한다.

유지보수 단계에서는 아직 반영되지 않은 사용자 스토리 및 요구사항을 새로운 릴리즈로 개발하기 위하여, 현재의 릴리즈를 고객이 계속 사용할 수 있도록 지원하는 과정이다. 조직이나 조직구성 요원이 변화될 가능성이 있다.

사망 단계는 어떤 스토리도 사용하지 않으며, 사용자의 기대와 같은 결과를 만들지 못할 때 더 이상 해당 소프트웨어를 사용하지 않는 상태이다.

IV. 제안 프로세스의 타당성

프로세스의 우수성을 확인하기 위한 가장 좋은 방법은 프로세스를 다양한 도메인과 다양한 소프트웨어 개발에 적용하고 사례 데이터를 모아서 통계적으로 분석하는 것이다. 그러나 이것은 많은 시간과 예산이 필요하며 IT 업계의 현실에 맞춰볼때 사례를 모으기가 거의 불가능하다. 본 논문에서는 제안한 프로세스의 특성과 XP가 갖는 문제점의 개선 측면에서 다음과 같은 4가지 정성적인 평가를 할 수 있다.

- XP 프로세스의 사용성 향상을 위한 최초의 연구이다.
- XP 프로세스가 인수시험 활동을 지원하도록 한다.
- XP가 CMM 레벨4를 만족하도록 지원하도록 한다.
- XP가 CMM 품질보증 실무를 지원하도록 한다.

표 1. XP의 소프트웨어 개발 생명 주기 지원
Table. 1 Support of Software Development Life Cycle of XP

지원유형 생명주기	프로젝트 관리 유무	프로세스 지원 유무	실무/활동/산출물 정의 유무
개념 생성	×	×	×
요구사항 명세	×	○	○
설계	×	○	○
코딩	×	○	○
단위 시험	×	○	○
통합시험	×	○	○
시스템 시험	×	○	○
인수 시험	×	×	×
동작중 시스템	×	×	×

표 2. XP의 CMM 핵심 프로세스 영역 만족도
Table. 2 Satisfaction of CMM KPA of XP

레벨	핵심 프로세스 영역	만족도
2	요구사항 관리	++
2	소프트웨어 프로젝트 계획	++
2	소프트웨어 프로젝트 추적 및 감독	++
2	소프트웨어 하청 관리	--
2	소프트웨어 품질 보증	+
2	소프트웨어 형상 관리	+
3	조직 프로세스 촛점	+
3	조직 프로세스 정의	+
3	프로그램 훈련	--
3	통합 소프트웨어 관리	--
3	소프트웨어 프로덕트 공학	++
3	조직간 조정	++
3	동료에 의한 검토	++
4	**정량적 프로세스 관리**	--
4	**소프트웨어 품질 관리**	--
5	결함 예방	+
5	기술 변경 관리	--
5	프로세스 변경 관리	--

\+ : XP에서 부분적 언급
++ : XP에서 대체적으로 언급
-- : XP에서 언급되지 않음

XP의 소프트웨어 개발 생명 주기 지원 현황을 보면 표 1과 같이 개념 생성, 인수시험, 동작중 시스템 생명주기에 대한 지원이 안되고 있으며, 프로젝트 관리는 전체 생명주기에서 모두 지원되지 않고 있다[1]. 본 논문에서는 인수시험 활동을 위해 프로세스 지원하고, 실무/활동/산출물 정의를 시도하였다.

XP의 프로세스가 CMM 핵심 프로세스 영역을 만족하는 정도는 표 2와 같다[20]. 표 2에서 레벨 2의 하청 관리는 XP가 중소 소프트웨어 개발 조직에서 사용되기 때문에 고려할 필요가 없고, 레벨3의 프로그램 훈련은 XP 자체의 범위 밖이다. 본 논문에서 제시한 프로세스를 통하여 레벨 4의 정량적 프로세스 관리와 소프트웨어 품질 관리 영역을 보완할 수 있다.

표 3은 XP가 CMM 핵심 프로세스 영역에 대하여 규정 실무를 만족하는 정도를 보여주고 있다. 본 논문에서 제시한 프로세스를 통하여 규정 실무의 소프트웨어 품질보증이라는 실무를 보완할 수 있다.

표 3. XP와 CMM 규정 실무
Table. 3 XP and Practices

공통 특징 (각각의 KPA)	CMM 실무	만족도
수행에 대한 일치	정책	--
	리더쉽과 스폰서쉽	--
수행에 대한 능력	조직적 구조	+
	자원과 자금	+
	훈련	+
측정과 분석	측정	+
구현 검증	선임 관리자 감독	--
	프로젝트 관리 감독	++
	소프트웨어 품질 보증	+

+ : XP에서 부분적 언급
++ : XP에서 대체적으로 언급
-- : XP에서 언급되지 않음

V. 결 론

극한 프로그래밍이 산업 실무현장에 더욱 확산될 것으로 예측되고 있다. 빠르고 쉽게 소프트웨어를 개발하고자 하는 극한 프로그래밍의 원래 취지를 준수하면서, 극한 프로그래밍의 사용성 개선과 같은 학문적으로 접근하는 연구가 필요하다.

본 논문에서 제안한 극한 프로그래밍 프로세스는 6개 단계와 11개 활동으로 구성되어 있으며, 4개의 활동에서 사용성 목표 설정 및 사용성 평가를 수행하도록 하였다.

제안한 프로세스의 타당성을 4가지 측면에서 확인하였다. XP의 사용성 향상을 위한 최초의 연구이며, XP 프로세스가 인수시험 활동을 지원하도록 개선하였으며, XP가 CMM 레벨4를 지원하도록 하며, XP가 CMM 품질보증 실무를 지원하도록 하였다.

추후 개발 사례들을 수집, 분석하여 제안된 프로세스의 우수성을 정량적으로 평가할 수 있도록 할 계획이다.

감사의 글

본 논문은 2003년도 정보통신부 정보통신기초기술연구지원사업(1-2003-A1-2000-0249) 지원으로 이루어진 연구로서, 관계부처에 감사 드립니다.

참고 문헌

[1] Abrahamsson, P. and et al., New Directions on Agile Methods:A Comparative Analysis, ICSE'03, IEEE, 2003.

[2] Agile Alliance Web Site: Manifesto for Agile software Development. On-line at: http://agilemanifesto.org/.

[3] Agile Methodologies Survey Results, On- line at: http://www.shinetech.com/agile_ survey _results.jsp

[4] Anderson, J. and et al., Integrating Usability Techniques into Software Development, IEEE Software, Vol. 18, No. 1, Jan./Feb., 2001.

[5] Beck, K., Embracing Change With Extreme Programming, IEEE Computer, Vol. 32, No. 10, 1999.

[6] Beck, K., Extreme Programming Explained :Embrace Change, Addison-Wesley, 2000.
[7] Becker, S. A. and Mottay, F. E., A Global Perspective on Web Site Usability, IEEE Software, Vol. 18, No. 1, Jan./Feb., 2001.
[8] Cockburn, A., Agile software Development, Addison-Wesley, 2002.
[9] Dix, A. J. and et al., Human-Computer Interaction, 2nd Edition, Prentice Hall, 1998.
[10] Ferre X. and et al., Usability Basics for Software Developers, IEEE Software, Vol. 18, No. 1, Jan./Feb. 2001.
[11] Fowler, M., The New Methodology, On-line at: http://www.martinfowler.com/articles/newMethodology.html.
[12] Giga Information Group Inc.http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,69182,00.html
[13] Highsmith, J. A., Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Publishing, 2000.
[14] ISO 9241-11, Ergonomics requirements for office work with visual display terminals(VDTs) part 11 : Guidance on Usability.
[15] ISO/IEC 9126, Information Technology -Software quality characteristics and metrics, 1998.
[16] ISO/TR 16982, Ergonomics of human -system interaction-Usability methods supporting human-centerd design, 2002.
[17] Jacobson, I., Booch, G., Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.
[18] Kieras D., A Guide to GOMS Model Usability Evaluation using NGOMSL, anon ymous ftp ftp.eecs.umich.edu people/kieras , 1996.
[19] Nielsen, J., Usability Engineering. Morgan Kaufmann, 1993. see also http://www .useit.com/papers/heuristic/heuristic_evaluation.html
[20] Paulk, M. C., Extreme Programming from a CMM Perspective. IEEE Software, Vol. 18, No. 6, Nov./Dec. 2001.
[21] Pressman, R. Software Engineering : A Practitioner's Approach(5th Edition), McGraw Hill, 2001.
[22] 권호열, 소프트웨어 개발 프로세스의 연구동향, 정보과학회지, 제20권 제3호, 2002.

저자 소개

이상준(Sang-Jun Lee)

1991년 전남대학교 전산통계학과 (이학사)
1993년 전남대학교 전산통계학과 (이학석사)
1999년 전남대학교 전산통계학과(이학박사)
1995년 ~ 현재 서남대학교 컴퓨터정보통신학과 조교수
※ 관심분야 : 소프트웨어공학, 컴퓨터보안, 컴퓨터교육, 모바일프로그래밍