국내 소프트웨어 개발 환경에서의 Extreme Programming 개발 방식의 적용성 연구

A Study of Applying Extreme Programming Method in Korean Software Development Environment

저자
(Authors)

공재원, 심우곤, 백인섭
Jae-Won Gong, Woo-Gon Shim, In-Sup Paik

출처
(Source)

한국정보과학회 학술발표논문집 28(1A), 2001.4, 577-579 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE00615014

APA Style

공재원, 심우곤, 백인섭 (2001). 국내 소프트웨어 개발 환경에서의 Extreme Programming 개발 방식의 적용성 연구. 한국정보과학회 학술발표논문집, 28(1A), 577-579.

국내 소프트웨어 개발 환경에서의 Extreme Programming 개발 방식의 적용성 연구

o 공재원, 심우곤, 백인섭
아주대학교 정보통신공학과
{jwgong, startear, ispaik}@madang.ajou.ac.kr

(A Study of Applying Extreme Programming Method in Korean Software Development Environment)

o Jae-Won Gong, Woo-Gon Shim, In-Sup Paik
*Graduate School of Information and Communication, Ajou University.

요 약

eXtreme Programming(이하 XP)은 프로젝트를 수행하는 데에 있어서 흔하게 발생하는 위험 요인들을 Light-weight 한 방법으로 해결하기 위한 방법론이다. XP 개발 방법론은 개발 주기의 단위를 작게 나누어서 피드백을 받는 기간을 짧게 잡았다는 점과, Pair Programming 방법을 도입했다는 점, 개발 과정에서의 고객의 비중을 높이고, 다양한 모듈 테스트/기능 테스트를 통해서 제품의 오류를 최소화 했다는 점 등의 특징을 갖고 있다. 그러나 XP 는 소규모의 개발팀일 경우에만 그 효용성이 입증되고 있으며, 수많은 테스트 과정과 개발의 전과정에서의 고객의 적극적인 참여의 요구는 생산 비용면에서나 고객의 부담에 있어서 단점으로 여겨지고 있다. 따라서 본 고에서는 이러한 점들을 보완하기 위한 몇 가지 방안과 더불어 우리나라의 개발 환경에 효과적으로 적용하기 위한 방안을 제시하고자 한다.

1. 서 론

프로젝트를 수행할 때에 반드시 지켜야 하는 것이 개발 완료 일정이라고 해도 과언은 아니다. 개발 일정을 못 지켰을 경우에 프로젝트의 성격에 따라 보상금만 지불하면 되는 경우부터 프로젝트 자체가 무산되는 경우까지 있다. 이러한 개발 일정을 방해하는 요인들에는 고객의 요구사항이 계속 바뀌는 경우라거나, 일정에 따라 작업 분배 계획이 제대로 나누어지지 못한 경우, 혹은 작업들간의 의존 관계 때문에 블로킹(blocking)상태에 빠진 경우들이 있을 수 있다[1][6].

본 고에서 다루고자 하는 eXtreme Programming 은, 객체지향 개발 방법론에서 많은 연구를 하고 있는 Kent Beck 과 Ward Cunningham 이 제안한 것으로, 위에서 제시한 것과 같은 상황 하에서 중소 규모의 개발 팀이 가장 효과적으로 개발을 할 수 있는 개발 방법론이다. 이 방법론에서 주목할 점은 개발 주기의 단위를 아주 작게 나누어서 피드백을 받는 기간을 짧게 잡았다는 점과, Pair Programming 방법을 도입했다는 점, 계속적인 단위 기능 테스트를 통해서 제품의 오류를 최소화 했다는 점 등의 특징을 갖고 있다.

그러나 이러한 XP 개발 방법론은 작은 규모의 개발팀 환경에는 적합하지만, 큰 규모의 팀 환경에서는 그 효과가 제대로 나타나지 못한다는 문제가 있다. 또 국내 소프트웨어 개발 환경에 적용하기에 적당하지 않은 몇 가지 문제점들이 있다. 따라서 본 논문에서는 이러한 문제에 대한 보완 방법을 제안해보고자 한다.

2. Extreme Programming

앞 서론에서도 언급했지만, 프로젝트를 수행하는 데 있어서 개발 일정이 조금씩 빗나가서 결국에는 최종 완료 일을 지키지 못하는 경우와 고객의 요구 사항이 계속 들어오거나 변경이 되는 경우가 대표적인 위험으로 꼽히고 있다. 개발한 프로그램이 주위 사용 환경의 변화로 인해서 계속적인 수정 작업이 필요해지다가 아예 못쓰게 되는 경우라던가, 개발되었던 소프트웨어의 결함이 점점 증가하여 결국 전체 소프트웨어를 버려야 하는 일도 생긴다. 또한 개발자와 고객간의 의사 교환이 제대로 이뤄지지 않아서 결국 요구사항에 부합하지 못하는 결과물이 나오는 상황도 많이 발생하며, 개발 기간이 늦춰지면서 중간에 개발진의 교체가 일어나는 경우도 흔하게 발생하는 위험 요소라고 할 수 있다.

XP 개발 방법론은 이와 같은 문제들을 짧은 기간의 단위 기능 개발과 많은 수의 테스트 작업, 그리고 고객을 개발의 중심에 두는(Customer-centered) 방법을 통해서 해결하기 위해 제안된 방법론이다.

그림 1 에서 보듯이 XP 개발 방식에서는 받아들인 고객의 요구사항(User Story)을 바탕으로 작업의 일정(Release Planning)을 잡는다. 잡혀진 일정은 최대한 작게 나뉘져 있기 때문에 반복적으로 일을 수행한다.(Iteration) 개발 작업 도중에 고객의 새로운 요구사항이 들어오면 언제든지 Release Planning 작업으로 돌아간다. 구현된 기능에 대해서 계속적인 테스트를 거치고(Acceptance Test), 일정 기간 단

위로 만들어진 기능들을 통합하는 작업을 한다 (Small Release).

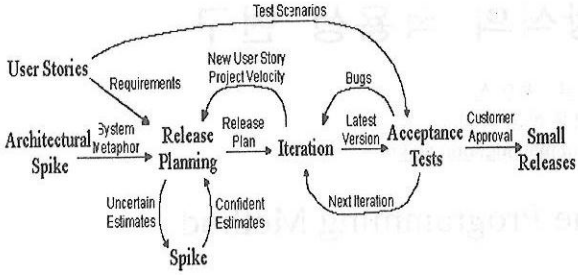

그림 1 Extreme Programming Project

2.1 짧고 반복적인 개발 주기

개발 기간 계획하는 과정에서 기초가 되는 자료는 고객이 작성한 User Story 카드이다. 아래 그림 2 와 같이 고객은 Story Card 를 작성한 것을 필요한 정도에 따라서 세가지로 값을 매긴다.(Define Value) 값이 매겨진 Story Card 에 대해서 개발자는 Cost 와 소요되는 기간을 다시 세 단계로 분류(측정이 정확하게 나오는 경우, 어느 정도 측정이 가능한 경우, 측정이 불가능한 경우)해서 고객에게 알려준다.(Estimate Cost) 개발자에 의해서 매겨진 Cost 와 개발 기간을 토대로 고객은 다시 Story Card 를 선별하게 된다.(Choose Value) 고객에 의해 선택된 Story Card 에 대해서 개발자는 기능을 구현한다(Build Value).

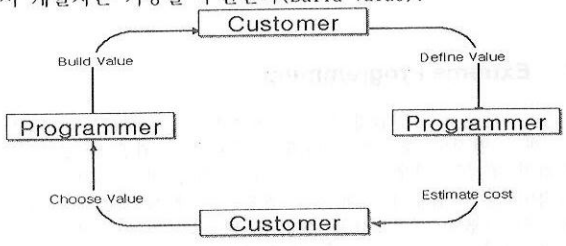

그림 2 Programmer estimates, customer chooses

구현을 할 때에는 기능을 최대한 간단하게 해야 하며 사용하는 알고리즘도 쉽고 간단해야 한다. 복잡한 알고리즘을 사용하거나 불필요한 기능을 추가하게 되면 초반에 예상하지 못했던 부분에서 문제가 생겨서 작업 일정을 어긋날 수 있기 때문에 핵심적인 기능만을 추려내는 작업이 상당히 중요하다.

테스트 및 구현을 하기 전에 간단한 기능 설계 과정을 거치기도 하는데 이때에는 CRC 카드 작성방식을 사용한다. 그러나 XP 는 설계에 들이는 시간은 최대한 짧게 잡고 테스트를 통해서 기능의 실용성 및 구현 가능여부를 판단하게 된다.

2.2 Pair Programming

Pair Programming 은 두 명의 개발자가 한 대의 컴퓨터를 두고 함께 알고리즘을 생각하고, 디자인을 하고, 코딩을 하는 방식이다. 이 방법을 처음 접하는 관리자의 입장에서는 중요한 인력 자원을 이중으로 낭비한다는 생각을 할 수 있고, 개발자의 입장에서는 Pair 로 작업하는 것이 방해로 여겨질 수 있다.

그러나 몇몇 사례 연구들을 살펴보면, Pair 로 작업하는 과정을 반복하는 동안에 개발 속도에서 오히려 속도의 향상을 가져왔을 뿐만 아니라 좀더 짜임새 있는 프로그램이 작성되었고, 에러 발생률도 줄어들었으며, 문제가 발생하였을 경우에 수정을 하는 시간도 더 적게 드는 등 다양한 효과를 가져온다는 것이 발표되었다. 부수적인 효과로 Pair Programming 을 하면 개발자들 간에 친목도가 높아져서 좀더 즐겁게 개발을 할 수 있다는 장점이 있다[8][9]

Pair Programming 방식은 아래 그림 3 에서 보듯이, 새로운 작업이나 실패한 적용테스트에 대해서(New Task or Failed Acceptance Test) Pair 를 구성하고 새로운 Unit Test 를 실시한다. 반복적인 Unit Test 를 통해서 새로운 기능이 구현되면(Create a Unit Test, Pair Programming) 통합 작업을 한다(Continuous Integration). 작업 도중에 구현할 기능에 따라 파트너의 교체가 일어날 수 있으며(Move People Around), 복잡한 코드의 경우에는 분해작업을 통해서 간단한 코드로 고치도록 한다(Refactoring).

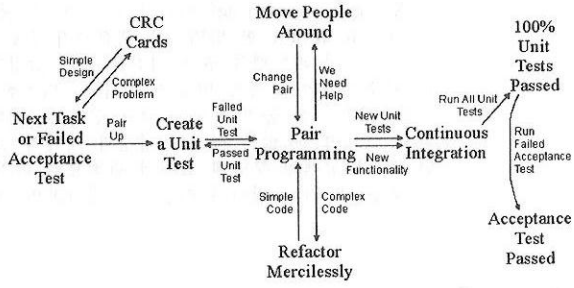

그림 3 Pair Programming[8]

2.3 많은 테스트 과정

XP 개발 방법론은 코드 중심(XP 방법론의 Key Activity)의 방법론이기 때문에 User Story 카드에서 제시된 기능들에 대해서 실용성 여부, 제대로 작동하는지 등의 판단 뿐만이 아니라, 팀원간의 의사교환, 복잡한 객체의 활동사항 정의, 문제 해결 결과보고서 작성, 그리고 작게 나누어진 기능들의 올바른 통합 여부도 다양한 테스트를 통해서 파악된다.

수시로 수행하는 테스트 결과를 통해 고객은 자신이 요구한 사항의 구현 상황을 금방 확인할 수 있게 된다. 이러한 빠른 피드백의 효과로 고객들은 좀더 명확하게 다음 요구사항을 제시할 수 있게 되고 개발자는 개발 번복의 과정을 줄일 수 있기 때문에 프로젝트 수행의 기간을 낭비하지 않게 할 수 있다. 또한 많은 테스트를 거치면서 다듬어진 코드는 좀더 견고해지고 간단해지기 때문에, 프로그램 내부에 결함이 생기거나 복잡해지는 것을 막아주며 나중에 여러 가지 개발 환경의 변화가 생기더라도 쉽게 대응할 수 있게 해준다.또, 계속적인 고객과의 의견 교류를 통해서 불필요한 기능이 생기는 것을 최대한 막아준다.

3. 국내 개발환경 적용에 대한 문제점과 보완 방안

3.1 문제점

XP 가 갖는 특성들을 살펴보았으나, 우리나라 개발 환경에 적용하기에는 다음과 같은 몇 가지 문제점이 있다.

첫째, XP 개발 방식은 팀원들간에 서로의 지식과 코드를 항상 공유하고, 자신이 맡은 부분 뿐만이 아니라 다른 사람의 작업 내용도 잘 알고 있어야 한다는 기본 성질 때문에 개발팀의 규모가 10 명 이하의 소규모 팀에서만 큰 효과를 발휘한다. 그러나 여러 팀이 함께 작업을 해야 하는 성격을 갖는 프로젝트의 경우에 XP 는 각 팀간의 업무 한당 방법이라던가, 팀간의 의사 소통 방법, 팀들의 운영 방법에 대한 정확하고 구체적인 Guideline 을 제시하고 있지 않다. 그렇기 때문에 이러한 작업 환경에는 XP 가 적합하지 못하다는 문제가 있다.

둘째, XP 는 고객간/개발자간의 의견교환을 통해서 User Story 를 분석하고 테스트하는 작업을 중심으로 하기 때문에 설계 과정을 대부분 생략한다(Quick Design). 그러나 팀의 규모가 커지게 되면 직접 의견교환을 하는 방식은 시간 소모가 커진다는 문제가 있으며, XP 에서 사용하는 CRC 카드는 텍스트 중심이라는 제한된 성격 때문에 많은 사람들이 쉽게 이해하는 데에는 비효율적이다[5]. 또한 설계 과정 부분을 약화시킨 것은 긴 기간동안의 개발에는 적합하지가 않다.

셋째, XP 에서 사용하는 Pair Programming 방식이 여러 가지 효과를 나타내고 있다고 하더라도 우리나라 개발 환경에 적용하기에 문제가 있어 보인다. 중소 규모의 개발 업체에서의 가장 큰 문제가 개발 인력의 부족이라고 해도 과언이 아닌데, 이 방식은 회사 경영진이나 관리자 입장에서 쉽게 받아들이기 어렵다.

넷째, XP 개발 방식은 고객이 개발팀에 합류하여 항상 개발자들과 함께 테스트를 하는 방식을 취하도록 하고 있다. 하지만 우리나라 개발 환경에서는 이와는 반대로 소수의 개발자가 고객쪽으로 파견을 나가서 개발을 하는 형태를 취하고 있기 때문에 XP 에서 말하고 있는 방법으로 전환하는 데에는 의식의 전환부터 필요하다.

3.2 보완 방안

첫째, XP 를 일정규모 이상의 프로젝트에 적용하게 되는 경우 전체 프로젝트를 유닛(unit)단위로 나누기 위해서는 프로젝트 초기에 시스템 아키텍처를 구상하는 작업이 필요하다. 아키텍처는 시스템 구축전반에 걸쳐 적용되는 최상위 수준의 지침이며 이를 통해 사각, 상충, 중복 개발을 미연에 예방할 수 있는 장점이 있다. 현재의 XP 로는 개발사이클의 극대화를 꾀하기 위해서는 자칫 쉬운 부분의 작업만을 개발초기에 구축하는 우를 범할 수 있는데, 시간이 소요되더라도 시스템의 핵심부분이나 공통부분의 개발이 선행되어야 하기 때문이다. 특히 아키텍처를 구상함으로써 규모가 커짐으로써 발생할 수 있는 '팀간의 의사소통' 문제나 '각 팀별 작업의 분배' 의 근거로 삼을 수 있다.

둘째, '팀간의 의사소통' 문제해결을 위해서는 각 팀별로 CRC 카드와 병행하여 의미전달의 애매모호성을 최대한 줄이고 서로간의 이해를 도모하기 위한 기법의 사용이 요구된다. 이를 위하여 시각적인 효과를 통해 이해력을 높일 수 있는 UML 의 Use Case 다이어그램은 가장 대표적인 기법이라 할 수 있겠다. 더불어 Formal method 의 사용도 시도될 수 있으나 개발자 모두가 Formal method 를 익혀야 하는 부담이 따른다.

셋째, 사용자 중심의(Customer-centered) 개발 기법인 XP 를 국내에 도입키 위해서는 사고의 전환이 필요하다. 전술한 바와 같이 사용자를 위한 시스템의 구축이라고 하더라도 발주측으로 개발자가 일부 파견을 나가는 형식이 주류를 이루는 국내환경에서는 제대로 된 XP 의 적용이 요원하다. 따라서 개발이전에 발주측을 충분히 이해 시키고 계약서에 사용자가 개발측으로 파견되어 모든 개발주기 동안 개발에 있어 중심이 될 수 있도록 명시하는 것이 필요하다.

넷째, 중소기업의 경우 Pair Programming 은 용납할 수 없는 효율성 저하라고 단정할 수도 있고, 한편으로는 인력의 이동으로 인하여 수행 중이던 프로젝트의 진행에 차질을 겪는 경우도 있으므로 반대만 할 수도 없는 상황이다. 그러므로 국내환경에서는 모든 스텝에 Pair Programming 을 도입하기 보다는 프로젝트의 마감에 모든 개발자들이 모여 수행한 프로젝트에 대한 기술, 비기술적 요소에 대한 Review 을 수행하는 방안을 생각해 볼 수 있다. 만일 개발기간이 길고 이직률이 높은 상황에는 중요 단계, 즉 Milestone 이 있는 단계마다 이러한 Review 를 수행하는 것도 해결책으로 제시될 수 있다.

끝으로 우리나라의 개발 여건에서 XP 를 효과적으로 도입하기 위해서는, 개발 비용에서 큰 압력을 받지 않으면서 개발 인력이 충분한 학교 수업과 산학 협동으로 수행되는 프로젝트에서 우선적으로 XP 개발 방법을 다양하게 시도해 볼 필요가 있다. 이렇게 시도된 결과에 대해서 사례 연구가 활발하게 진행된다면 개발자들이나 관리자들에게 XP 방법론을 인식시키는 데에 많은 도움이 될 것이고 우리나라 개발 성격에 적당한 *Pair Programming* 방식을 찾아낼 수 있을 것이다.

4. 결론 및 향후 연구과제

본 글에서는 프로젝트 수행 시에 흔하게 발생하는 문제들에 대해서 XP 개발 방식의 해결 방안을 살펴보았다. 또한 XP 개발 방식을 확장된 환경에 적용하고자 할 때와 우리나라 개발 환경에 적용하고자 할 때 나타날 수 있는 문제점들을 살펴보았고 그에 대한 보완 방안을 일부 제시해 보았다.

향후 과제로는 수행하게 될 프로젝트에 XP 개발 방법론을 적용하여 보고 도출할 수 있는 문제점과 이를 지원·및 개선할 수 있도록 개발 도구, 기법, 지침 등에 대한 실질적이고 구체적인 분야에 대한 연구가 필요하다.

5. 참고문헌

[1] Kent Beck, "Extreme Programming Explained:Embrace Change", Addison-Wesley, 2000.

[2] Kent Beck, "Extreme Programming Explained:Planning", Addison-Wesley, 2001

[3] Alistair Cockburn, Laurie Williams, "The Costs and Benefits of Pair Programming", XP2000 Conference, 2000.

[4] Ron Jeffries, "Extreme Programming:Installed", Addison-Wesley, 2001.

[5] Craig Larman, "Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design", Prentice Hall PTR, 1997.

[6] Roger S. Pressman, "Software Engineering:A Practitioner's Approach", 4th edition, McGraw-Hill, 1997.

[7] J. Donovan Wells, http://www.extremeprogramming.org, 1999.

[8] Laurie Willams, Ward Cunningham, Robert R.Kessler, Ron Jeffries, "Strenthening the Case for Pair Programming", IEEE SOFTWARE , July/August 2000.

[9] RoleModel Software, Inc. http://www.rolemodelsoft.com/XP/xpStudio.htm.