콘텐츠 소프트웨어 개발 환경을 위해 확장된 익스트림 프로그래밍 방법

서영수*, 정헌, 강병욱

The Method of Extended Extreme Programming for Content Software Development Environment

Yeung-Su Seo*, Byung-Wook Kang, Hun Jung

요 약

소프트웨어 개발 프로젝트를 수행할 때 가장 중요한 것은 개발 기간 내에 계획된 품질의 결과물을 얻는 것이다. 특히 배포 시기와 품질에 민감한 콘텐츠 소프트웨어 개발의 경우 프로젝트가 연기되거나 원하지 않는 품질의 결과물이 나오게 되면 바로 전체 프로젝트의 실패로 연결된다. 익스트림 프로그래밍 기법은 개발 기간이 중요한 프로젝트의 위험 요소를 줄이기 위해 보다 작은 단위로 개발 주기를 나누는 방법이다. 본 논문에서는 콘텐츠 소프트웨어 개발 환경을 위해 개발 기간과 품질이라는 두 가지 주요 요소를 동시에 고려할 수 있는 확장된 익스트림 프로그래밍 방법을 제안한다. 제안하는 기법은 개발 프로젝트 진행 중에 제안되는 잉여 아이디어의 문서화 장치와 페어 프로그래밍 기법을 확장하여 다중 역할 모델을 적용하는 방안이다.

Abstract

When performing a software development project, the most important thing is building a result with planned quality within development period. Particularly, if the project is delayed or has no good quality in the case of the content software development project which is sensitive to the release time and quality, it is immediately connected to the failure of the whole project. Extreme programming is a methodology that divides the development cycle into smaller units for reducing the risk factor of the project in which the development period is important. In this paper, we suggest the expanded extreme programming which can consider the development period and quality at the same time for content software development environment. The suggested methods are documentation mechanism that is upcoming during the development project and multiple role model which is extended from pair programming method.

*Keywords : 콘텐츠(Content), 소프트웨어(Software), 익스트림(Extreme), 프로그래밍(Programming), 메서드(Method)

※ 접수일 : 2008.09.18 , 심사완료일 : 2008.11.28

*영남대학교 컴퓨터공학과

Ⅰ. 서론

전통적인 소프트웨어 개발 프로세스 모델 중에서 가장 보편적으로 사용되며 검증이 잘된 모델로 인정받고 있는 폭포수 방법론이 성공적으로 프로젝트를 완료하려면 설계 공정에 들어가기 전 단계에 이미 사용자가 개발자에게 개발하고자하는 프로젝트에 대한 모든 요구사항을 정확하게 전달해야만 한다. 그러나 사용자는 설계 이전 단계에서는 아직 추상적인 형태로만 존재하는 완성품에 대한 요구사항을 정확하게 파악하지 못하는 경우가 일반적이다. 따라서 개발 초기 시점에서 개발 완료 시점에 필요한 요구사항 전체를 일관성 있게 모두 전달하는 것은 어려우며 오히려 개발자에 의해 제품이 구체화됨에 따라 그 시점에 이르러서야 비로소 사용자는 구체적인 요구사항을 인지하게 되는 경우가 대부분이다[1]. 특히 최근의 소프트웨어 개발 추세를 보면 이러한 요구사항의 점진적인 전달 경향은 일반적인 형태이며, 특히 사용자가 초기 개발 단계에서 제시한 요구사항과는 달리 변경된 요구사항을 프로젝트 진행 중간에 제시하는 경우가 자주 발생한다.

기존의 소프트웨어 개발 방법론이 해결할 수 없는 이와 같은 문제점을 해결하기 위해 소프트웨어 개발주기를 보다 작은 단위로 나누어 해당 공정을 반복적으로 수행함으로써 프로젝트 개발공정의 중간에 요구사항이 변경되는 경우에도 위험요소를 줄일 수 있는 애자일(agile) 프로그래밍 기법이 제안되었다[2][3]. 특히 애자일 기법 중에서 익스트림 프로그래밍(extreme programming) 기법은 프로젝트의 개발주기를 보다 작은 단위로 나누어 개발자가 설계, 구현, 시험과 같은 개발공정을 전체 프로젝트 개발기간에 걸쳐 반복적으로 수행하도록 하면서 사용자에게 적극적으로 구체화된 중간 결과물에 대한 정보를 전달하여 요구사항의 변경에 대한 대응성을 높여 위험요소를 줄이고 개발 비용을 줄이도록 한다[4].

소프트웨어 시장의 규모가 커지고 관련 기술이 발전함에 따라 과거에는 시스템 소프트웨어와 응용 소프트웨어로 구분되던 소프트웨어 제품 분류방법도 더 세분화되었다. 특히 응용 소프트웨어는 표 1-1과 같이 다양한 종류로 구분되는데, 여러 응용 소프트웨어의 종류 중에서 최근 관심이 커지고 있으며 시장에서 급부상하고 있는 콘텐츠 소프트웨어 분야에 대한 연구가 필요하다.

표 1-1. 응용 소프트웨어의 구분

구분	종류
기업용	기업용 그룹웨어 등
기업기반용	데이터베이스, 이메일서버, 네트워크, 보안관리시스템 등
정보처리용	워드프로세서, 스프레드시트, 이메일 클라이언트 등
콘텐츠제어	미디어플레이어, 웹브라우저 등
콘텐츠	게임, 플래시, 광고, 교육용, 시뮬레이션, 뉴미디어 등
미디어개발	그래픽에디터, 웹에디터, 애니메이션 에디터, 사운드에디터, 동영상에디터 등
제품개발	CAD, CAE, 컴파일러, IDE 등

콘텐츠 소프트웨어도 기본적으로는 다른 소프트웨어처럼 디지털 기기 상에서 동작하기 때문에 콘텐츠 소프트웨어 개발방법은 비콘텐츠 소프트웨어의 개발방법을 차용하고 서로 다른 특성에 대해서는 차별화된 지원방법을 마련해야 한다.

익스트림 프로그래밍 기법은 개발 프로젝트 전체과정 동안 사용자의 의견이 결과물의 요구사항에 잘 반영될 수 있으므로 이 기법을 개발기간 동안 사용자의 수정된 의견을 항상 반영해야만 하는 콘텐츠 소프트웨어 개발에 활용하는 것은 타당하다. 그리고 개발 프로젝트를 진행하는 중에 제안되는 아이디어의 문서화, 페어 프로그래밍 기법에 다중 역할 모델의 도입 등을 통해 콘텐츠 소프트웨어 개발의 효율성을 향상시킬 수 있도록 기존의 익스트림 프로그래밍 기법을 확장할 필요가 있다[5].

본 논문에서 제안하는 익스트림 프로그래밍 기법의 확장방안을 모바일 RPG 게임을 개발하는 프로젝트에 적용하여 그 효과를 분석함으로써 제안한 확장방안의 타당성을 보인다. 모바일 게임은 비교적 개발기간이 짧고, 투자비용이 적게 들며, 사용자들의 트렌드 변화가 심한 콘텐츠 소프트웨어이다. 사용자는 모바일 게임을 개발하는 프로젝트 진행 중에 다양한 요구사항을 변경하여 제시하게 되며, 개발자는 이에 유연하게 대처해야 한다. 따라서 익스트림 프로그래밍 기법을 적용하기에 적합한 분야이다. 다양한 게임 장르 중에서 RPG 장르는 비교적 많은 사용자층을 확보하고 있으며 시장 민감성이 다소 낮아서 적용 사례 분석을 위한 대상으로 사용한다.

II. 본론

2.1 연구배경

익스트림 프로그래밍 기법은 개발기간이 매우 중요한 프로젝트를 위해서 고안되었다[6]. 이 기법은 전통적인 방법보다 개발공정 주기의 단위를 작게 나누며 이러한 개발 단위를 반복적으로 수행하면서 전체 프로젝트를 진행하게 된다[7]. 이때 완성품에 대한 요구사항을 제공하는 사용자는 전체 프로젝트가 완료될 때까지 반복되는 작은 개발 단위의 테스트 단계에 참가하여 피드백을 제공하게 된다.

또한 이 기법에는 같은 개발환경에 두 명의 프로그래머들이 투입되는데, 한 명은 코드를 입력하고 다른 한 명은 그 코드를 지켜보는 공동 작업을 통해 같은 결과물을 완성해 나가는 페어 프로그래밍 기법을 사용한다. 코드가 입력되는 동안에 코드를 지켜보는 개발자는 개발 작업에 대한 전략적인 방향, 더 나은 개선책, 앞으로 나타나게 될 문제점 등에 대해 고려해야한다.

2.2 익스트림 프로그래밍

익스트림 프로그래밍은 애자일 소프트웨어 개발 기법에서 온 소프트웨어 공학의 방법론의 하나다[2]. 익스트림 프로그래밍의 목표는 소프트웨어 개발 프로젝트 진행 중에 사용자의 요구 사항이 변경되어 개발 기간이 변경되는 개발관련 위험요소를 줄이는 것이다. 전통적인 개발 방법론에서는 대개 애플리케이션에 대한 요구 사항이 개발 기간의 초기에 주로 결정되고 그 시점 이후에는 변경되는 경우가 거의 없다. 익스트림 프로그래밍 기법에서는 요구사항 변경에 대한 위험 부담을 줄일 수 있도록 사용자가 짧은 개발 반복 공정에 직접 참가한다. 개발 프로젝트에 참가한 사용자는 개발자가 제공하는 현재 단계에서 실행 가능한 예제를 작동시켜보고 그 결과를 다음 개발 반복 공정에 반영한다. 따라서 익스트림 프로그래밍 기법을 적용하면 사용자의 요구사항 변경에 대해 보다 주의를 기울이게 되므로 소프트웨어 개발 프로젝트가 유연해진다[3].

그림 2-1을 살펴보면, 익스트림 프로그래밍 방법에 비해 전통적인 객체지향 컴포넌트 기반 개발 방법에는 개발 단계에서만 피드백 장치가 마련되어 있을 뿐 고객의 역할이 거의 없으며 개발 주기의 단위가 훨씬 길다.

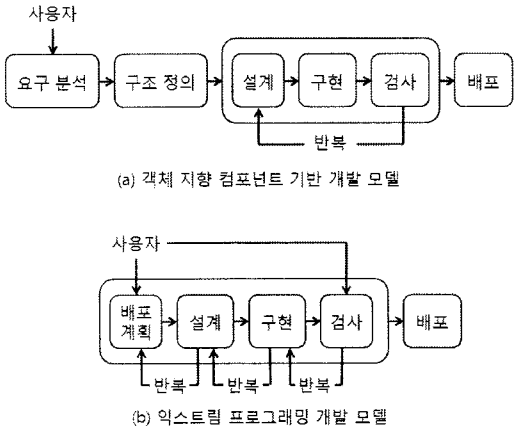

그림 2-1. 전통적 개발 모델 대비 익스트림 프로그래밍 개발 모델의 반복 공정상의 특징

익스트림 프로그래밍 기법이 기존의 개발 방법론에 비해 항상 장점만 가지는 것은 물론 아니다. 이 기법은 짧은 개발 공정을 얻기 위해 문서화에 대한 부분을 희생시켰다. 따라서 문서화를 중요하게 다루는 개발 방법과 비교할 때 몇 가지 잠재적인 약점을 가지고 있다. 첫 번째로 사용자가 불안정한 요구 사항을 제시하는 경우에 대한 문제가 있고, 두 번째로 개발자 상호 간이나 개발자와 사용자 사이의 대립이 발생한 경우에 문서화된 합의나 절충 방법이 없다는 문제가 있으며, 세 번째로 총체적인 설계 명세나 문서가 부족하다는 문제가 있다[4].

2.3 페어 프로그래밍

페어 프로그래밍은 두 명의 프로그래머가 한 대의 컴퓨터를 함께 사용하여 공동의 소프트웨어를 개발하는 방법이다[8]. 그림 2-2에서 표시한 것과 같이, 한 명이 개발과 검증에 대한 전체책임을 지는 전통적인 기법과 달리 한 명이 프로그램 코드를 입력하는 동안 다른 한 명은 그 코드를 자세하게 검토한다[9]. 코드를 입력하는 역할을 하는 프로그래머를 드라이버라고 부르고 드라이버가 입력하는 코드를 관찰하는 역할을 하는 프로그래머를 옵서버라고 부른다[10]. 두 명의 프로그래머는 필요한 경우 각자의 역할을 서로 교대할 수도 있다. 옵서버는 드라이버가 주동적으로 만들어내는 코드를 잘 관찰하면서 해당 작업에 대한 전략적 방향을 고려하고 더 좋은 아이디어나 잠재적인 문제가 발생할 가능성이 있는 경우 이를 알려준다[5].

(a) 전통적인 프로그래밍 (b) 페어 프로그래밍

그림 2-2. 전통적인 프로그래밍과 페어 프로그래밍

옵서버의 역할을 맡은 개발자가 해당 작업에 대한 전략적 방향을 고려하고 더 좋은 아이디어나 잠재적인 문제가 발생할 가능성이 있는 경우 이를 알려줌으로써 드라이버는 자신의 주의를 현재의 작업을 완료하는데 전적으로 집중할 수 있다[11].

III. 확장 방법

콘텐츠 소프트웨어 개발의 경우 개발 기간과 결과물의 품질이 동시에 중요한 요소이다. 이러한 특성을 지원하는 개발 방법으로 익스트림 프로그래밍 기법이 있다. 이 기법을 콘텐츠 소프트웨어 개발 환경에 적용할 때 추가적으로 지원해야하는 장치가 있다. 개발 과정 중에 제안되는 아이디어를 관리할 수 있는 방법과 다양한 개발 주체들 간의 원활한 의사소통을 지원할 수 있는 방법을 마련해야 한다.

3.1 아이디어 문서화

비콘텐츠 소프트웨어의 경우 출시시기를 단축하려는 노력이 대부분이며 공정상의 문제가 발생하여 개발이 지연되지 않는 한 일부러 출시를 늦추는 일은 거의 없다. 그러나 이와는 달리 출시시기에 따라 제품의 가치가 크게 달라지는 콘텐츠 소프트웨어는 제품의 가치를 더 높이기 위해 초기 개발 프로젝트에서 계획했던 것과는 달리 의도적으로 출시시기를 앞당기거나 늦추는 경우가 있다. 이것은 항상 변하는 시장 동향이나 기업의 전략에 따라 최대의 이익을 얻을 수 있도록 콘텐츠 소프트웨어의 배포 일시가 조정될 수 있기 때문이다. 즉, 콘텐츠 소프트웨어 개발자들은 개발 프로젝트의 완료 일정이 변화하거나 취소될 수도 있다는 사실을 피할 수 없음은 물론이며 오히려 이에 따르는 위험요소에 항상 잘 대비하고 있어야 한다[7].

현실적으로 개발사의 입장에서는 현재 진행 중인 프로젝트가 어떠한 상황에 놓여있더라도 최대 수익을 얻을 수 있도록 노력해야만 한다. 어떠한 상황이라는 것은 개발 프로젝트가 지연되거나 혹은 완전히 중지되고 다른 프로젝트를 시작할 수도 있다는 것을 뜻한다. 따라서 개발 프로젝트가 진행되고 있는 전체 일정 동안에 있어서 최선의 해결책은 개발자들이 바로 현재 시점까지 완료된 모든 자원을 항상 확보하고 있는 것이다. 이렇게 준비된 자원들은 설계 명세, 그래픽 이미지, 사운드 자원, 프로그램 모듈 등으로서 개발 프로젝트가 다음 단계로 진행되면 다음 단계에 사용하고 그렇지 못하고 전체 프로젝트가 중단되는 경우에는 추후에 다른 프로젝트에서 사용하도록 한다.

물론 익스트림 프로그래밍 기법을 도입하여 수시로 변할 수 있는 요구사항에 유연하게 대응할 수 있도록 하고는 있지만 기존의 익스트림 프로그래밍 기법은 주로 개발주기를 줄이는 방법에 대해 집중하고 있을 뿐 다른 소프트웨어 개발방법에서 중요하게 다루는 문서화에 대해 중요하게 고려하고 있지 않다[12]. 비콘텐츠 소프트웨어 개발환경에 익스트림 프로그래밍 기법을 적용한 경우라면 현시점까지의 자원만 확보하고 문서화에는 무게를 두지 않아도 되겠지만 콘텐츠 소프트웨어의 경우 개발공정 중에서 생기는 아이디어가 다음 공정 혹은 다른 프로젝트의 품질을 향상시키고 비용을 절감하는데 큰 영향을 미친다. 그러므로 아이디어를 관리할 수 있는 문서화 장치를 마련할 필요가 있다.

표 3-1. 콘텐츠 소프트웨어 개발 과정 중에 나타나는 아이디어의 종류와 특성

	활성 아이디어	비활성 아이디어	잉여 아이디어
적용 비용	낮음	높음	중간
적용 효과	중간	낮음	높음
적용 시기	현재	없음	미래
파급 효과	프로젝트 일부	없음	프로젝트 전체
양	많음	많음	적음
가치	중간	낮음	높음
비용 절감 효과	중간	낮음	높음
품질 향상 효과	중간	낮음	높음

표 3-1에 콘텐츠 소프트웨어를 개발하는 과정 중 발생하는 아이디어의 종류와 특성에 대해 나타냈다. 현재 개발이 진행 중인 프로젝트에 바로 적용이 가능하며 적용 비용도 적당한 활성 아이디어는 자원으로 간주하여 보존이 되고 현재 프로젝트나 추후 프로젝트에 적용할 경우 드는 비용이 크고 적용 효과가 낮아 적용이 불가능한 비활성 아이디어는 폐기가 되지만 주로 다음 프로젝트의 시발점이 되는 중요한 잉여 아이디어의 경우에는 적용 시기가 현재가 아니기 때문에 별다른 보존 장치가 없이 개발자에 의해 개인적으로 관리되거나 아예 보존되지 못하고 버려지는 경우가 많다.

본 논문에서는 이러한 문제점을 보완할 수 있는 아이디어 문서화 장치를 익스트림 프로그래밍 기법의 개발 프로세스 구조에 추가했다. 추가된 아이디어 문서화 단계의 시점은 그림 3-1에서 책 모양의 아이콘으로 표시했다. 개발자들은 현재 시점까지 완료된 각자의 결과에 대한 아이디어 문서화를 세 부분에서 수행한다.

첫 번째 문서화 장치는 기획 단계로서 기획자가 주역할을 하게 되며 현재 시점에서 적용하기는 곤란하지만 다음 프로젝트의 시작점이 될 만한 잉여 아이디어만 메모한다. 개발공정을 반복하면서 잉여 아이디어는 발생할 수도 있고 하지 않을 수도 있으므로 문서화 작업으로 인해 개발이 지연되면 곤란하다. 두 번째 문서화 장치는 개발 반복 단계이므로 프로그래머와 디자이너가 주역할을 하게 되며 다음 프로젝트에서 적용 가능한 기법이나 그래픽에 대한 잉여 아이디어만 메모한다. 마지막 문서화 장치는 검사 단계이므로 평가자가 생각하는 새로운 아이디어 중에서 다음 프로젝트에 적용했으면 좋겠다는 잉여 아이디어를 보존한다.

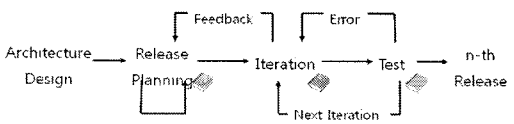

그림 3-1. 잉여 아이디어 문서화

잉여 아이디어를 문서화함으로써 얻어지는 이점은 다음과 같다.

첫 번째로 개발 프로젝트 자체가 중단되는 경우와 같은 극단적인 위험에 대한 부담을 줄일 수 있다. n번째 개발 반복 과정까지 보존된 문서들은 다음 프로젝트에서 유용하게 재사용될 수 있다.

두 번째로 현재 프로젝트 보다는 다음 프로젝트에 적용했을 때 가치가 높은 잉여 아이디어를 기록함으로써 현재 개발 중인 프로젝트에 대한 이해도를 높일 수 있고 현재의 개발과정이 어느 정도 진척된 상황인지 파악할 수 있다. 잉여 아이디어는 활성 아이디어보다 추상화 정도가 높기 때문에 복잡한 시스템의 경우라도 거시적으로 전체 시스템의 개요를 한 눈에 쉽게 살펴볼 수 있는 특징이 있기 때문이다.

세 번째로 개발자 상호간의 잉여 아이디어 문서화 내용을 살펴봄으로써 서로가 생각하고 있는 개념을 파악할 수 있게 되어 개발자들 간 의사소통에 도움이 된다. 이것은 현재 프로젝트를 개발하고 있는 개발자들은 현재의 시스템을 바탕으로 항상 미래의 프로젝트를 생각하게 되는데 이러한 생각은 현재의 시스템 구성 작업에도 영향을 미치기 때문이다. 개발자들이 서로의 의사소통을 원활하게 할 수 있으면 의견 불일치에 따르는 피드백의 수를 현저하게 줄일 수 있다.

3.2 다중 역할 모델 적용

기존의 페어 프로그래밍 기법에서 일반적인 개발환경은 두 명의 프로그래머가 함께 작업하는 것이다[13]. 콘텐츠 소프트웨어 개발환경의 경우 기획자는 다른 기획자, 그래픽 아티스트, 프로그래머와 함께 공동 작업을 하고 프로그래머는 기획자, 그래픽 아티스트, 다른 프로그래머와 함께 공동 작업을 한다. 즉, 작업의 종류가 애플리케이션 소프트웨어 개발의 경우처럼 프로그래밍만으로 국한되지 않으며 공동 작업에 투입되는 사람의 수가 매우 가변적이다.

콘텐츠 소프트웨어 개발 프로젝트의 개발자들은 비교적 컴퓨터라는 개발 매체에 익숙한 프로그래머 이외에도 그렇지 못한 기획자와 디자이너도 있다. 물론 예외도 있지만 일반적인 경향은 그렇다. 비콘텐츠 소프트웨어의 개발환경과 달리 기획자와 디자이너 역시 콘텐츠 소프트웨어의 경우에는 개발공정에 직접적으로 참여하는 개발의 주체들이므로 이들 모두에 대해 협동 작업을 원활하고 효율적으로 수행할 수 있는 장치가 필요하다.

그림 3-2에서 전통적인 콘텐츠 소프트웨어 개발공정을 보인다. 모든 개발 주체들이 자신의 영역에서 작업을 하는 이 방법은 시스템의 규모가 크고, 개발자들을 잘 이해시키고, 그들에게 확정된 계획을 줄 수 있도록 구분된 공정을 필요로 하는 복잡한 개발에 적합하다.

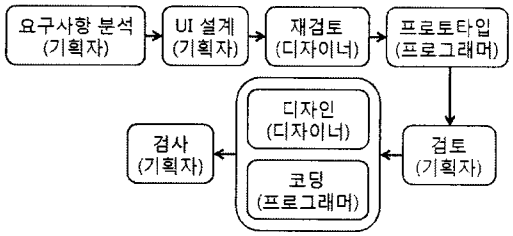

그림 3-2. 전형적인
콘텐츠 소프트웨어 개발 역할 분담 구조

제안하는 다중역할 모델이 적용된 페어 프로그래밍 방법에서 일반적인 개발환경은 다수의 개발자들이 공동 작업을 하는 것이다. 특히 서로 관련된 작업 영역을 가지고 있지 않더라도 문제점의 발견이나 발전적인 의견 제시 등의 활동으로 상호 협력자가 될 수 있으므로 개발자들 고유의 작업 영역은 더 이상 문제가 되지 않는다. 두 명의 개발자로 국한되지 않으므로 기존의 페어 프로그래밍 방법에서 사용하는 드라이버와 옵서버라는 용어 대신 주와 부 역할자로 구분한 다중역할 모델에 대한 개발 분담 구조를 그림 3-3에서 나타낸다.

다중역할 모델은 한 사람이 책임을 맡고 있는 특정 작업 영역에 대해서 다양한 상호 협력자들의 의견과 협조를 얻을 수 있도록 한다. 한 가지 주의해야할 점은 자신의 작업 영역에 대한 책임을 계속 미루고 부역할자의 임무만 수행하는 개발자가 있다면 전체 개발공정이 멈추게 된다는 사실이다. 이 문제는 전체 개발공정에 대한 일정을 주기적으로 재전파하고 프로젝트 진행 상황을 개발 주체들이 잘 파악할 수 있도록 프로젝트 매니저가 활동함으로써 해결할 수 있다.

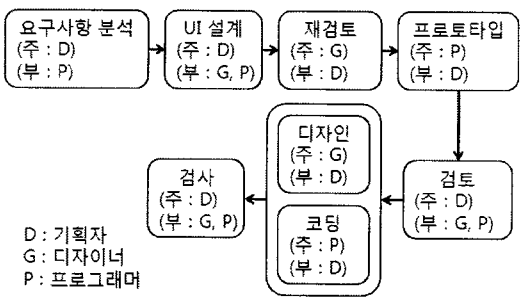

그림 3-3. 다중역할 모델이 적용된
콘텐츠 소프트웨어 개발 분담 구조

Ⅳ. 결과

제안하는 기법을 모바일 RPG 게임 개발 프로젝트에 적용해 보았다. 콘텐츠 소프트웨어 중에서도 모바일 게임 분야는 사용자와 시장의 흐름에 민감하며 비교적 개발기간이 짧고 투자비용이 적다. 따라서 모바일 게임은 개발공정 중 요구사항 수정이나 개발 프로젝트 일정 변경이 빈번하며 제안하는 확장된 익스트림 프로그래밍 기법을 적용할 때 큰 효과를 볼 수 있다[7].

잉여 아이디어를 문서화하는 경우와 문서화하지 않는 경우에 대한 게임 품질의 차이를 비교하기 위해서 여덟 가지 전투수행 스테이지를 대상으로 분석 작업을 했다. 1, 3, 5, 6 스테이지는 잉여 아이디어를 문서화하면서 개발을 했고 2, 4, 7, 8 스테이지는 문서화 작업을 하지 않으면서 개발했다. 각 스테이지마다 지형, 장애물, 적 캐릭터 등의 조건이 서로 다르기 때문에 분석 작업을 객관적으로 수행하기 위해 주인공 캐릭터를 움직여서 지정된 장소로 이동하면서 하나의 적 캐릭터를 제거하고, 두 가지 스킬을 사용하고, 한 번의 타격을 받는 시나리오를 분석의 대상으로 했다.

분석을 위해 작성된 휴리스틱 평가 문항은 Nielsen의 열 가지 평가 기준을 바탕으로 작성되었다[14]. 휴리스틱 평가방법은 전문가에 의한 평가로서 기존의 연구결과 및 경험을 바탕을 바탕으로 시스템을 설계하는 방법 가운데 많이 사용되고 있는 평가방법이다[15]. 이상적으로 시스템의 품질을 검증하려면 현재까지 나온 평가 기법의 다양성 때문에 체계적으로 평가 기법을 선택하는 과정에서 과부하가 걸린다. 따라서 이러한 복잡한 지침을 적용하고 검증을 통해 확인하는 과정을 대신하여 소수의 전문가들로 하여금 그들의 기준으로 시스템의 품질을 평가하는도록 하는 것이 좋다. 일반적으로 전문가들이 수행하는 휴리스틱 평가에 의해 전체 시스템 설계에 존재하는 약 80%의 문제점이 검증되며 이때 발견하지 못한 부분은 인수 검사 과정에서 설문지 평가를 통해 사용자의 검증을 통해 발견하도록 한다.

평가 기준에 따라 작성된 휴리스틱 평가 항목에 따라 모바일 게임 관련 전문가 그룹이 평가지를 작성했다. 모바일 게임 관련 전문가 그룹은 모두 5년 이상의 경력자로서 기획자 한 명, 프로그래머 두 명, 디자이너 한 명, 마케터 한 명으로 구성했다. 전체 문항은 총 열 개로 구성되어 있으며 각 문항의 점수는 10점 만점이다.

표 4-1. 잉여 아이디어 문서화 유무에 따른 품질평가 점수(100점 만점)

스테이지	휴리스틱 평가(점)	설문지 평가(점)
1	81	82
2	82	84
3	87	91
4	83	84
5	88	90
6	90	92
7	83	85
8	84	86

설문지의 문항은 국내 이동통신 3사에서 모바일 게임의 심사에 사용하는 품질 기준을 참조하여 작성했다. 설문지 평가는 인수 시험 단계에서 사용자를 대상으로 수행되며 퍼블리셔의 기획자와 마케터를 평가자로 선정하였다. 전체 문항은 총 40개로 구성되어 있으며 각 문항의 점수는 2.5점 만점이다. 평가자들은 점수 이외에도 평가 항목에 대해 자신의 의견을 기술하도록 했다.

표 4-1의 분석결과를 보면 전달되는 잉여 아이디어가 없는 1, 2 스테이지에 대한 지수는 비슷하다. 1 스테이지의 잉여 아이디어가 전달되는 3 스테이지는 전달되는 정보가 없는 4 스테이지에 비해 평가점수가 향상되었다. 게임에 대한 평가 향상 효과는 잉여 아이디어의 전달이 이루어지는 5, 6 스테이지와 전달이 이루어지지 않는 7, 8 스테이지를 비교해도 뚜렷하게 구분된다.

다중 역할 모델을 적용한 경우와 적용하지 않은 경우에 대한 사용성 품질 향상 정도를 비교하기 위해서 장비창, 능력치 창, 상점 창을 대상으로 분석했다. 표 4-2의 분석 결과를 보면 다중 역할 모델을 적용하여 개발한 시스템의 경우에 대해 사용자가 좋은 평가 결과를 보였다.

표 4-2. 다중 역할 모델 적용 유무에 따른 품질평가 점수(100점 만점)

평가 종류	적용안함	적용함	점수향상
휴리스틱 평가(점)	80	88	10.0%
설문지 평가(점)	84	93	10.7%

V. 결론

본 논문에서는 일반적인 소프트웨어 개발 환경과 다른 특징을 갖고 있는 콘텐츠 소프트웨어 개발 환경을 위한 새로운 기법을 제안했다. 이 기법은 익스트림 프로그래밍 기법을 확장한 것으로서 개발자들이 익숙한 기존의 개발 방법론을 그대로 사용하되 콘텐츠 소프트웨어 개발 환경을 고려한 장치가 추가되었다. 제안한 방법에 대한 검증을 위해 모바일 RPG 게임 개발 프로젝트를 이용하여 전문가와 사용자의 품질 평가 점수를 비교했다. 확장된 익스트림 프로그래밍 기법을 사용하여 개발된 경우에 모두 높은 점수의 평가 결과가 나타났다

지금까지 콘텐츠 소프트웨어는 비콘텐츠 소프트웨어에 비해 상대적으로 그 중요도가 낮았다. 하지만 현대의 소프트웨어 시장 동향을 살펴볼 때 소프트웨어 시장에 있어서 콘텐츠 소프트웨어의 자리매김은 결코 무시할 수 없는 중요도를 보이고 있다. 따라서 콘텐츠 소프트웨어의 특성을 잘 살펴보고 적절하게 개발 환경을 지원할 수 있는 연구가 활발하게 진행되어야 할 것이다.

현재까지의 소프트웨어의 품질에 관한 연구는 기존의 하드웨어 분야에서 사용하는 기능성 관련 연구의 결과를 따르는 것이 많다. 그러나 무형의 산물이며 추상적인 대상인 소프트웨어의 품질평가를 위해서는 사용성 품질에 대한 정형화 작업 및 평가방법 등의 관련 연구가 필요하다. 특히 콘텐츠 소프트웨어는 사용성 품질이 제품의 가격에 큰 영향을 주기 때문에 사용성 품질에 관련된 기법을 발굴하여 지원할 필요성이 있다.

앞으로 익스트림 프로그래밍 기법이 산업 실무현장에 더욱 확산될 것으로 예측된다. 이 기법은 다양한 개발환경에 대해 최적의 솔루션이 될 수 있도록 부족한 부분을 확장하여 필요한 개발환경에 특화될 수 있는 장점을 가지고 있다. 빠르고 쉽게 소프트웨어를 개발하고자 하는 익스트림 프로그래밍 기법의 원래 취지를 준수하면서 단점을 개선하기 위해서는 보다 다양한 연구가 필요하다. 소프트웨어 개발 초기부터 개발에 사용자가 참여하도록 하는 익스트림 프로그래밍 기법의 특성을 살리면서 콘텐츠 소프트웨어의 개발 효율 향상을 위해서 사용자 입장에서 느끼는 사용성 품질 향상에 대한 연구가 필요하다.

콘텐츠 소프트웨어를 기존의 비콘텐츠 소프트웨어와 분리해서 그 특성에 맞추어 연구를 하는 경우는 드물다. 따라서 콘텐츠 소프트웨어의 특성에 대한 선행 연구를

통한 다양한 지표의 발굴이 시급하다. 현재로서는 기존의 개발방법을 차용하고 수정하여 콘텐츠 소프트웨어 개발환경에서 이용하고 있는 수준이다. 관련 연구가 진행됨에 따라 전용의 이론, 개발방법론, 도구들이 개발될 것으로 기대한다. 이러한 연구는 현재 시작 단계에 있으며 지금부터 본격적으로 관심을 집중해야 할 것으로 본다. 이러한 연구는 현재 시작 단계에 있으며 지금부터 본격적으로 관심을 집중해야 할 것으로 본다.

참 고 문 헌

[1] Aoyama, M., "Web-Based Agile software Development," *IEEE Software*, November/December, 1998.

[2] Cockburn, A., Agile software Development, Addison-Wesley, 2002.

[3] Martin R. C., Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002.

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

[5] Yeung-Su Suh, Byung-Wook Kang, and et al., "A Study of Applying Extreme Programming Method in Mobile Game Development Environment," *ICIC2008* vol. 1, pp. 59-62, 2008.

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

[7] Kent Beck, Extreme Programming Explained: Planning, Addison-Wesley, 2001.

[8] Jae-Won Gong, Woo-Gon Shim, In-Sup Paik, "A Study of Applying Extreme Programming Method in Korean Software Development Environment," *KIISE Conference*, 2001.

[9] Laurie Williams, "Building Pair Programming Knowledge through a Family of Experiments", *Proceedings of ISESE,* 2003.

[10] Williams, Laurie, Pair Programming Illuminated. Addison-Wesley, 2003.

[11] Alistair Cockburn & Laurie Williams, "The Costs and Benefits of pair programming," *XP2000 submission*, 2000.

[12] Ambler, S., Agile Modeling : Effective Practices for Extreme Programming and Unified Process, John Wiley & Sons, Inc., 2002.

[13] 권호열, "소프트웨어 개발 프로세스의 연구동향," 정보과학회지, 제20권 제3호, 2002.

[14] Nielsen, J., Usability Engineering. Morgan Kaufmann, 1993.

[15] Sang-Jun Lee, Seok-Chan Bae, "A Plan for Improvement of Usability in Extreme Programming," KIPS(Korea Information Processing Society) transactions, Part D, Volume 11D. Issue 3, pp.635-648, 2004.