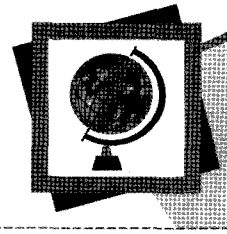
| 사례발표 02 |



게임에 있어서 개발 방법론의 적용 -거상 2 개발 과정을 중심으로

최 일 곤
(조이온)

목 차

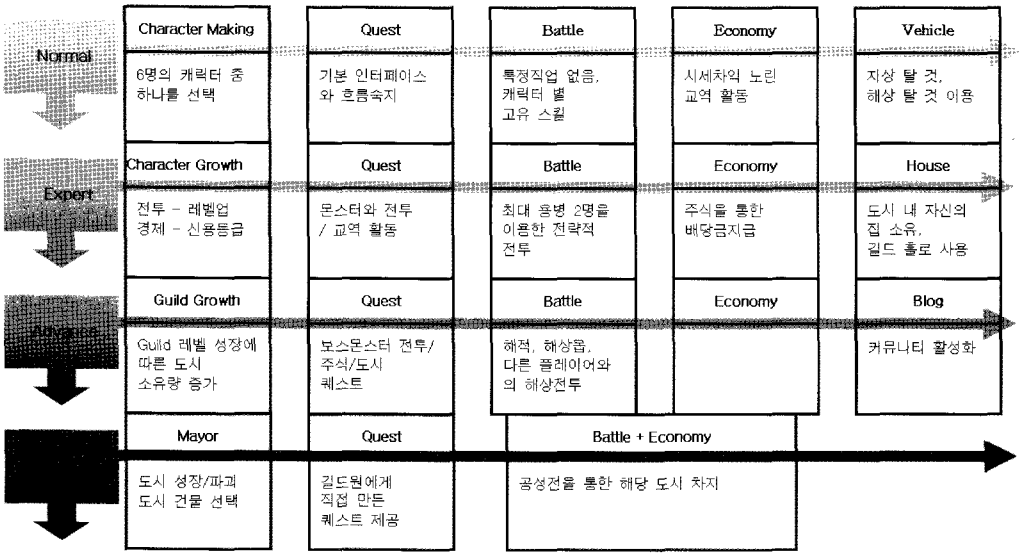
1. 서 론

2004년도 국내 게임 시장은 전년도에 비하여 10%에 가까운 성장으로 약 4조 3천억의 규모가 되었다. 그러나 소수의 개발그룹을 제외하고는 아직도 주먹구구식 게임개발과정을 면치 못하고 있는 것이 현실이다. 소수의 선도적인 개발자들은 그들의 프로젝트에서 부분적으로나마 SI 분야에서 적용되었던 소프트웨어 개발방법론을 시도하지만, 대부분의 경우에는 어떤 정련된 시스템에 의존하기 보다는 특정한 인력에 의존하여 개발하는 경우가 더 많으며, 심지어 개발자 자신이 개발방법론의 도입 자체에 대하여 거부감을 느끼기까지 하는 실정이다.

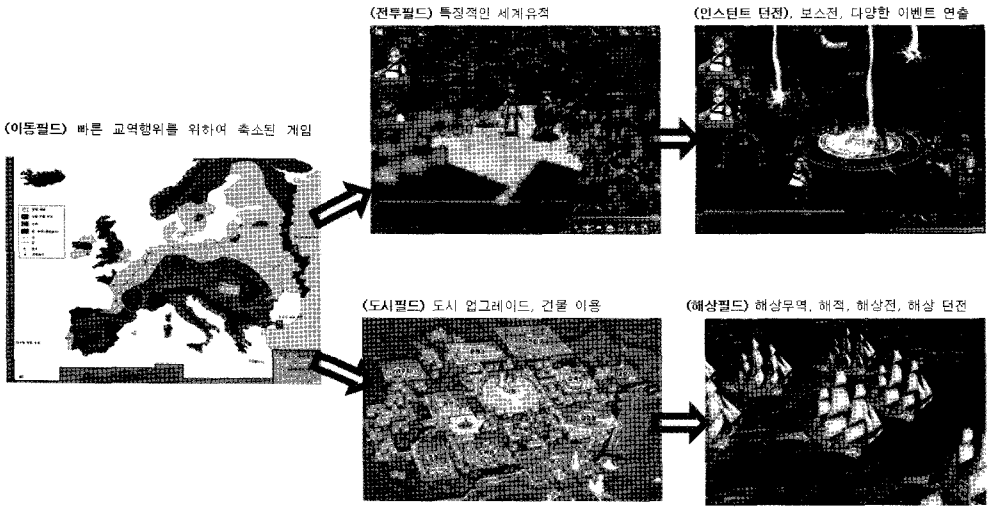
게임소프트웨어는 일반적인 어플리케이션 소프트웨어와 달리 실행 코드 외에 그래픽, 사운드, 그리고 스크립트같은 코드 외의 내용을 많이 포함하고 있다. 다시 말해서 게임 제작 공정은 기술적인 부분 외에 예술적인 면까지도 고려되어 준비되어야 한다는 것이다. 최근 미국이나 일본 등 선진외국의 개발자들을 면담해보면 다양한 개발방법론을 적용 또는 원용하고 있다는 사실을 확인할 수 있었으나, 복합적 소프트웨어인 게임에 어떤 것이 가장 최적인 것인가에 대해서는 의견이 분분하며 여전히 다양한 실험들이 이루어지고 있다는 것을 알 수 있었다. 특히 게임 개발 공정 전반에 걸쳐서 방법론을 적용한 성공적 사례는 아직 발견하기 어렵다.

㈜조이온에서는 전작인 '거상'을 포함하여 다수의 프로젝트를 진행하면서 이러한 문제점을 깊이 인식하고 해결방법을 찾기 위하여 노력한 결과 사내에서 진행되는 '거상 2'를 포함한 여러 프로젝트에 몇 가지 시도들을 하게 되었다. 본 원고에서는 가까운 미래에 출시를 앞두고 있는 '거상 2' 프로젝트에 적용한 개발방법론을 소개하고자 한다. 코드뿐만 아니라 그래픽이나 사운드 등의 제작에도 여러 가지 방법론을 적용하였으나 여기에서는 주로 프로그램 부분에 대한 적용을 중심으로 기술하고자 한다.

우선 '거상 2' 프로젝트에 대하여 게임 플레이 패턴과 게임 구성을 통한 간단한 소개와 함께 구체적으로 적용된 방법론에 대하여 기술하



(그림 1) 게임 플레이 패턴



(그림 2) 게임 구성

고 그 결과물로서 몇 가지 산출물에 대한 소개를 할 것이다. 그리고 마지막으로 방법론 적용 결과에 대해 고찰을 한 후에 향후 방향성에 대한 논의를 할 것이다.

2. 거상 2에 대한 간략한 소개

2.1. 게임 소개

거상 2는 일반 MMORPG와는 다르게, 16C 세계 무역시장을 배경으로 한 경제 시뮬레이션 (Massively Multi-player Online Economic Simulation, MMOES) 게임이다. 거상 2의 특징으로는 전작인 '거상'에 비하여 전 세계로 확장된 배경, 위험과 낭만이 가득한 바다를 넘나들며 자신의 항로를 개척하게 되는 해상전, 용병과 함께 모험을 할 수 있게 되는 동료와의 여행, 자신이 직접 시장이 되어 한 도시의 경영권을 맡게 되는 경영시스템, 강화된 경제 컨텐츠로서

의 주식과 교역시스템 그리고 게임 내 모든 요소들이 각각의 특별한 이야기들을 가지고 있는 스토리위주의 진행을 들 수 있다.

2.2. 게임 플레이 패턴

(그림 1)에서 볼 수 있듯이 거상 2에서는, 단순히 전투위주의 진행이 아닌, 경제와 전투, 퀘스트가 맞물려 플레이어가 취할 수 있는 여러 가지 패턴이 복합적으로 일어나게 된다.

2.3. 게임 구성

거상 2는 (그림 2)에서 볼 수 있듯이, 이동필드, 도시필드, 전투필드, 인스턴트 던전 필드, 해상필드는 여러 곳의 필드를 이동할 수 있도록 구성되어있다.

3. 거상 2에 적용된 개발방법론

위의 게임 설명에서 볼 수 있듯이 거상 2는 초기 기획 시부터, 타 MMOG와는 다른 경제 시뮬레이션 게임이라는 모토를 걸었고, 이는 곧 개발의 난항을 의미하기도 했다. 타 온라인게임과 같은 전투위주의 컨텐츠보다는, 수많은 데이터를 처리해야 하는 경제 관련 컨텐츠가 많은 부분을 차지하고 있으며, 동시에 전투와 맞물려 돌아가게 되어 있다.

개발자 중 일부는 '거상 1' 개발경험을 가지고 있으며 다른 개발자들은 다른 프로젝트를 1번 이상 완료한 프로그래머들이며 이전 프로젝트에서 기획에서의 요구 사항 외에도 예측할 수 없는 일정이나 통제되지 않는 요구사항들, 아키텍처 부족에 의한 통합의 어려움, 업무 분담, 커뮤니케이션 문제 등을 이미 경험하고 있었다. '거상 1' 개발 당시, 많은 기획 아이템들이 실제 개발 시의 난해함으로 사장되는 것을 수없이 지켜본 결과 거상 2는 이와 다른 시각으로 접근해야 함을 깨달았고, 다음과 같은 요구사항에 직면하게 되었다.

■ 거상 2의 방대한 기획과 온라인 게임의 특성상 추후 수없이 바뀌는 요구사항에 신속히 대처할 수 있어야 한다.
- 온라인게임의 경우 추가된 컨텐츠들이 유저의 피드백에 의해 쉽게 추가, 삭제, 수정되기 때문에, 잦은 요구사항변화에도 신속하고 안전하게 대처할 수 있어야 한다.

■ 거상 2 개발에 소요되는 일정과 비용이 예측 가능해야 한다.
- 새로운 컨텐츠를 추가하기 위해 드는 비용, 그리고 추가될 경우 영향을 끼치는 컴포넌트들은 무엇인가 등을 신속히 알아내고 계산해 낼 수 있어야 한다.

■ 개발 후 들어가는 유지보수가 용이해야 한다.
- 상용 서비스 후에도 지속적으로 새로운 패치를 통해 변화를 꾀해야 하는 온라인게임의 특성상, 개발자들이 손쉽게 새 시스템에 적응할 수 있도록 문서화되어 있어야 하며, 또한 추후 유지보수 작업이 용이해야 한다.

■ 재사용이 가능해야 한다.
- 여러 온라인게임에 공통적으로 쓰일 수 있는 인벤토리, 경매소, 잡화점 시스템 등 많은 부분들을 재사용하여 향후 개발 시 유용하게 사용할 수 있어야 한다.

새로운 프로젝트를 시작하면서 이러한 실수를 반복하지 않기 위해서 많은 자료들을 참조하고 수 차례의 회의를 한 끝에 '거상 2'에 적용한 방법론과 개발된 프로세스는 RUP(Rational Unified Process) 기반의 프로세스이다. 게임 개발에 맞게 각 절차를 커스터마이징하였으며, 이를 블루스 프로세스, 프레임워크(Blues process, frame work)라고 칭하였다. 블루스 프레임워크는 공통

적인 유틸리티, 패턴, RPC 등의 지원을 통해 게임 개발의 생산성 향상을 지원해주는 윈도우 기반의 프레임워크이다.

'거상 2'의 개발 방법론 적용에는 많은 시행착오가 있었다. 처음으로 시도한 방법론은 UP와 CBD를 변형한 형태이다. 그러나 너무 많은 문서 제작과 제약들이 프로젝트의 진행을 더디게 만들었다. 특히, 유스 케이스(use case)의 범위 등 경직된 방법론은 그 자체가 장애가 되었다. 개발팀은 실용주의 프로그래머의 정신을 따르기로 하였다. 실용주의 프로그래머는 어떤 특정 기술에 매이면 안 되며, 개별 상황마다 그 상황에서 좋은 해결방안을 고를 수 있도록 충분한 배경지식과 경험을 소유하고 있어야 한다. 개발 프로세스를 따르지만 방법론의 노예가 되어서는 안되며 또한 아키텍처를 중시하고 프로세스에서 명시된 산출물을 제작하려고 노력한다. 따라서 활동성을 제한하는 프로세스 및 프로세스에 따른 산출물을 최소한으로 유지하려고 노력하였으며 이는 바로 애자일 프로세스(agile process)에 기초를 둔다. 개발팀은 CBD의 원칙을 깨지 않는 범위 내에서 제약이 적은 방법론을 찾고자 애자일 프로세스, 특히 XP를 주로 검토하여 그 중 XP와 애자일의 장점 몇 개를 CBD에 적용하여 블루스 프로세스를 제작하게 된다. XP는 UP에 비해서 제약이 거의 없다고 할 정도로 단순함을 추구한다. 블루스 프로세스는 XP의 TDD나 Pair Programming, 단순함, 피드백, 잦은 반복 등을 적용한다.

3.1 블루스 프로세스

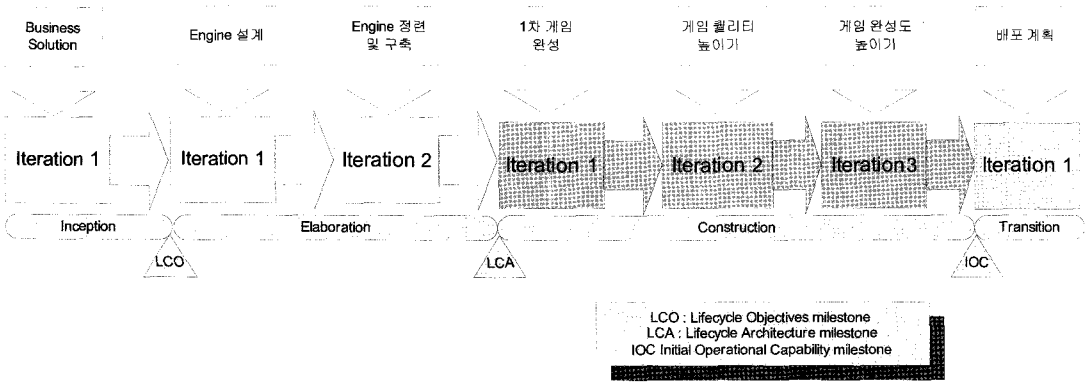
RUP, XP, CBD 등의 방법론을 검토하였고, 최소한의 프로세스만을 유지하기 위하여 다음과 같은 프로세스를 도입하였다[SPSP][RUP][XPI].

각 반복(Iteration)은 요구사항 수집, 분석 및 설계, 구현, 테스트, 배포를 포함한다. Inception 단계에서는 주로 계획을 세우고 요구사항을 수집하며, 아키텍처 및 설계에 대한 가이드라인을 세우는데 초점을 맞추었다. Elaboration 단계에서는 아키텍처 문서를 제작하고, 수집된 요구사항에 대한 디테일한 설계에 초점을 맞추었다. Construction 단계에서는 아키텍처 및 설계도를 기준으로 구현 및 테스트를 수행하였다. Transition 단계에서는 클로즈 베타를 위한 배포 계획을 세우고, 테스트로 초점을 맞추었다.

블루스 프로세스의 주요 특징은 다음과 같다.

■ 일련의 반복을 통한 개발 절차를 제공한다.

- 블루스 프로세스는 RUP의 핵심개념인 'Inception, Elaboration, Construction, Transition의 각 주기마다 반복을 가지는 절차'를 게임 개발에 맞



(그림 3) RUP기반의 블루스 프로세스

게 커스마이징하여 사용함으로써, 개발 시 나올 수 있는 여러 위험들을 미리 식별하고 제거할 수 있다.

■ 아키텍처 패턴을 기반으로 개발할 수 있다.
- 블루스 프로세스는 이전의 MMOG 개발에서 축적되었던 여러 설계 방식들을 패턴화하여 아키텍처 패턴으로 제공함으로써, 안정적인 시스템을 구축 할 수 있다.

■ 개발 표준안, 용어사전 등을 지원한다.
- 블루스 프로세스는 공통적인 표준안, 용어사전을 통해, 개발 사항의 이론적 틀이나, 사고를 등을 통일할 수 있도록 지원함으로써, 개발의 방향성을 확립해준다.

■ 컴포넌트 기반의 개발을 지원한다.
- 블루스 프로세스는 컴포넌트 인터페이스 단위의 조립을 통해 각 기능의 개발/대체가 손쉽기 때문에 여러 결과물들의 재사용성을 높여준다.

3.2 블루스 프레임워크(Blues Framework)

'블루스 프레임워크' 의 주요 특징은 다음과 같다.

■ 게임 개발을 위한 자체적인 컴포넌트 생성, 배포, 관리 환경을 제공한다.
- 블루스 프레임워크는 기존의 .NET, J2EE, CCM 이 아닌, 블루스(Blues)만의 개발환경을 제공한다. 윈도우의 DLL 기술을 이용하여 블루스 표준으로 컴포넌트를 생성하고, 추후 오퍼레이션의 추가 삭제 등의 관리까지 플러그인을 통해 제공하며 그 과정이 자동화된다.

■ 게임개발에 필요한 유틸리티, 보안, 트랜잭션 등의 패키지를 제공하여 준다.
- 블루스 프레임워크는 개발자의 요구사항을 즉각 받아들일 수 있는 구조로, 요구사항에 맞게 게임 개발에 필요한 여러 유틸리티 패키지를 제공해줌으로써, 이에 드는 제작비용을 감소시켜준다.

■ 코드규약, Exception, 패턴 등의 게임개발을 위한 표준안을 제시한다.
- 블루스 프레임워크는 코드규약, Exception 규약, 패턴 등 게임 개발에 필요한 여러 가지 사항에 대한 표준안을 제시함으로써 성공적인 CBD 프로세스를 지원해준다.

■ 게임에 최적화된 자체 RPC(Remote Procedure Call)를 이용하여, 분산 컴포넌트 환경을 지원한다.
- 게임에 가장 최적화된 네트워크 구조를 통해, 효율적인 RPC를 지원함으로써 분산컴포넌트 환경을 구축할 수 있으며, 이로 인해 보다 나은 컴포넌트 기반 소프트웨어를 만들 수 있다.

각 단계에서 초점을 맞추는 부분이 다르기는 하지만 매 반복을 시작할 때 새로운 요구사항을 수집하였고, 기존 요구사항을 분석하고, 설계를 재검토하는 시간을 가졌다. <표 1>은 각 단계를 거치면서 제작된 결과물이다.

<표 2>는 단계별 거상 2의 실행 결과물의 내용이다.

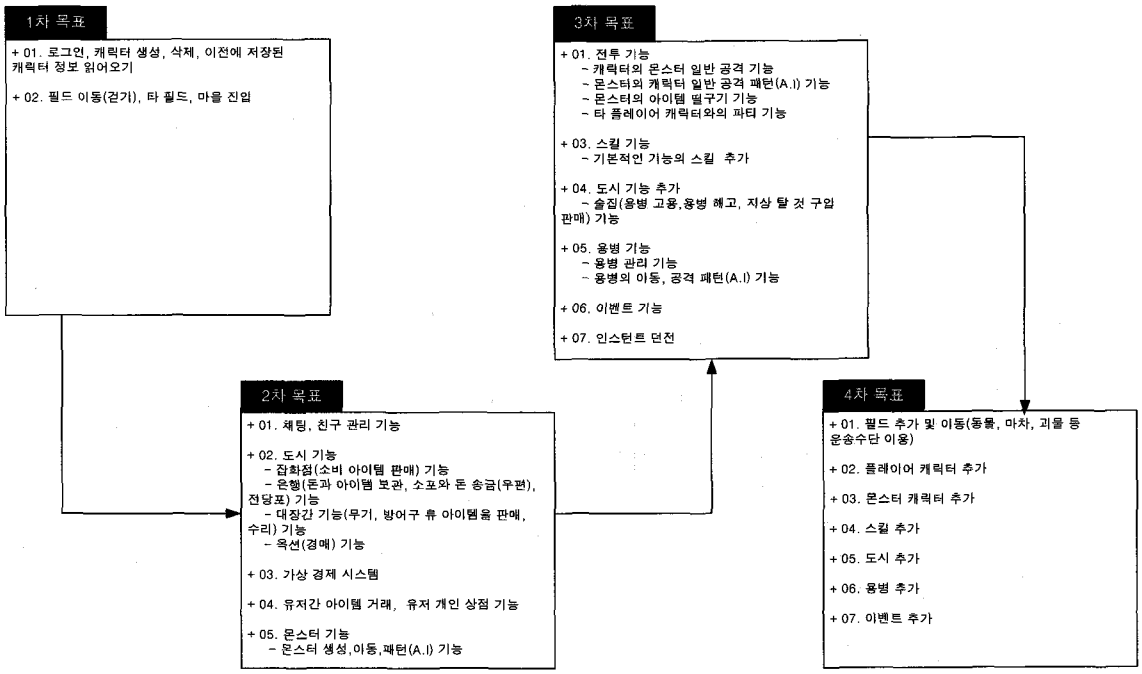
3.3 계획 세우기

프로젝트 계획 단계에서 다음을 수행한다.
- 소프트웨어 개발 계획
- 프로젝트 추정
- 품질 보증 계획
- 단계별 납품 계획
- 요구사항 개발
- 아키텍처
- 상세설계

〈표 1〉 단계별 산출물

도입(Inception)	1) 프로젝트 수행 계획서 – 우리의 비전은 무엇이고, 어떠한 방식으로 테스트를 진행하여, 각 단계에 어떤 결과물을 내놓으며, 이러한 과정을 통해 최종 목표를 실현한다는 것을 서술한다. 2) 요구사항 요청 기술서, 테크니컬 문서, 작업결과 보고 문서, 아키텍처 문서, 유스케이스 기술서 등의 문서 공통 포맷. 3) 중요 요구사항 및 유스케이스 간략 기술서 4) 용어 사전 5) 위험관리 계획서 6) 표준안 – 개발환경, 방법, 코딩 룰 등의 전체적인 시스템의 표준이 포함되어야 한다. 7) 프로토타입 – 기획자가 그려서 넘겨주는 러프한 UI 프로토타입 및 게임의 방향성을 결정할 수 있는 실행 가능한 프로토타입을 제작한다. 8) 요구사항 기술서(상세히 설계할 필요가 있는 경우의 요구사항만 기술한다) 9) 아키텍처 기술서 – 전체 시스템의 설계도, 청사진이 될 자료들이 포함되어야 한다. – 필요한 경우 액터, 유스케이스, 액터 간 상호관계, MVC 개념 모델, 시퀀스 다이어그램을 포함한다. – DB 설계도 – 간략 화면 설계서(각 윈도우의 관계도까지만 표시한다)
발단(Elaboration)	1) 요구사항 기술서(업데이트) 2) 아키텍처 기술서(업데이트) 3) 클래스 모델 및 설계도 – 상세히 정리할 필요가 있는 유스케이스에만 적용된다. 4) 컴포넌트 모델 및 설계도 – 위의 클래스 모델을 모아서 컴포넌트 모델을 만든다. 5) 실행 결과물 6) 테스트 프로그램
구축(Construction)	1) 요구사항 기술서(업데이트) 2) 아키텍처 기술서(업데이트) – 클래스, 컴포넌트 구체화 모델 및 설계도 – 실제 구현이 가능한 레벨의 상세 설계를 포함한다. – 세부 화면 설계서: 모든 예, 아니오 등의 예외 윈도우까지 포함한다. 3) 클래스 모델 및 설계도(업데이트) 4) 컴포넌트 모델 및 설계도 – 실행 가능한 컴포넌트: 설계도를 통해 만들어진 실행 가능한 컴포넌트이어야 한다. 5) 실행 결과물(업데이트) 6) 테스트 프로그램(업데이트) 7) DB 구축 – 상세히 DB를 설계하고 구축한다.
이전(Transition)	배포가 가능한 시스템

〈표 2〉 단계별 거상 2 실행 결과물



프로젝트의 가시성을 높이기 위하여 먼저, 프로젝트의 대의명분을 유지할만한 비전이나 목표를 설정한다. 프로젝트를 10% 정도 수행한 후 PCR(Planning Checkpoint Review)[SPSG]을 실시한다. 계획 대비 실적을 정기적으로 비교하여 계획대로 잘 진행되고 있는지 파악한다. 그리고 시정 조치가 필요한지, 계획이 수정되어야 하는지 등을 판단한다. 일이 끝났는지 여부를 확인하기 위하여 상세한 마일스톤(binary milestones)을 설정한다. 프로젝트 각 단계가 완료되면 주요 추정치를 수정한다. 이렇게 함으로써 계획도 충실해지며, 주요 가정(assumptions) 사항들도 신뢰도가 높아진다.

3.4 요구사항 관리

처음에 시도하였던 요구사항 관리는 Writing Effective Use Cases[WEUC]에서 소개된 방식을 사용하였다. [WEUC]에서 소개된 방식은 정확한 요구사항을 수집할 수 있도록 하고, 요구 사항에 대한 시나리오를 작성하고 변경에 대한 통제도 가능하다. 그러나 시간이 지나면서 요구사항의 추가는 매우 힘든 작업이며 변경 또한 쉽지 않다. 무엇보다 요구사항 문서를 작성하는데 너무 많은 시간을 소비한다. 따라서 XP에 소개된 스토리 작성을 요구 사항 관리의 방법으로 선택하게 되는데 스토리 작성 방법은 매우 간단하다. 각자가 맡은 파트(예를 들어 서버, 클라이언트, 엔진, 툴 등)에 대해서 서버와 클라이언트는 기획자를 고객으로, 엔진 프로그래머는 클라이언트 프로그래머와 그래픽 디자이너를 고객으로 하여 스토리를 수집하여 메모지에 작성한다. 1주일에 한번 정도 수집된 스토리를 가지고 토론을 하였고 계획을 세웠으며 업무를 분담한다. 또한 각 스토리에 대한 짧은 설계를 같이 하고 요구 사항 및 아키텍처 문서를 업데이트한 후 구현에 들어간다.

3.5 품질 보증

품질 보증 계획으로 결함 추적, 단위 테스트, 소스코드 추적, 테크니컬 리뷰, 통합 테스트, 시

스템 테스트 등의 품질 보증 작업을 수행한다. 테크니컬 리뷰를 통하여 경험 있는 개발자와 초급 개발자가 서로 교류할 수 있게 하고, 품질 높은 설계도와 소스 코드를 생산하게 되며 다음과 같은 품질 보증 계획을 세운다.

- 소프트웨어 품질 보증 활동에 대한 계획을 세운다.
- 소프트웨어 품질 보증 활동 계획을 문서화 한다.
- 소프트웨어 품질 보증 활동은 소프트웨어 요구사항 활동 혹은 그 이전에 시작한다.
- 2~3명의 소프트웨어 품질 보증 담당자를 정한다.

3.6 아키텍처

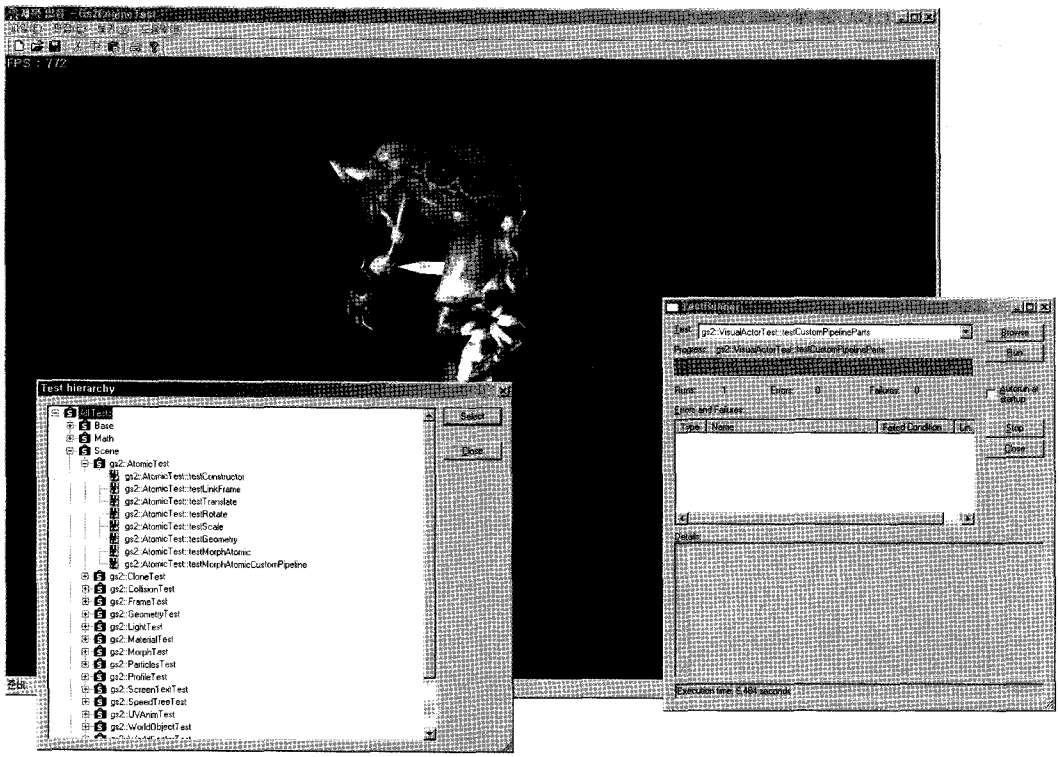
엔진, 클라이언트, 서버, 툴 등이 서로 잘 통합될 수 있고, 업무 분담을 할 수 있는 4~50 페이지 분량의 아키텍처 문서를 제작한다. 개발에 사용된 아키텍처 문서의 일부를 아래에서 설명한다.

3.7 테스트

테스트를 작성하면 기능이 완료되었는지, 기획에서 요구하는 기능을 만족하는지를 테스트하고, 버그가 발견될 경우 특정 기능에 초점을 맞춰서 버그를 추적할 수 있다. 또한 각 스토리의 완료 여부를 테스트 통과 여부로 결정할 수 있고 프로그램 또한 인터페이스에 맞춰서 프로그래밍이 되기 때문에 깔끔한 구조를 유지하도록 도와준다. XP에서 강조하는 TDD(test driven development)를 테스트 방식으로 결정한다[XPI][TDD].

테스트 주도 개발(TDD)의 리듬은 다음과 같다.

- 빠르게 테스트를 추가한다.
- 모든 테스트를 실행하고 새로 추가한 것이



(그림 4)

실패하는지 확인한다.

- 코드를 조금 바꾼다.
- 모든 테스트를 실행하고 전부 성공하는지 확인한다.
- 리팩토링을 통해 중복을 제거한다.

일반적으로 TDD는 주로 특정 기능에 대한 테스트 완료 여부를 계산에 의해서 검사할 수 있는 기능에 대한 예제만을 설명한다. GUI 기능에 대한 테스트 방식을 설명하는 부분이 있기는 하지만 3D 엔진에 대한 쉽고, 직관적인 테스트를 가능케 하고자 한다. 3D 렌더링에 관한 테스트를 눈으로 확인할 수 있도록 하는데 (그림 4)는 스펙큘라 매핑을 몬스터 캐릭터에 적용하는 기능을 테스트하는 그림이다. 물론 TDD에서 추구하는 완전 자동화된 테스트는 아니지만 그런대로 만족스러운 결과이다.

3.8 리스크 관리

리스크 관리를 위하여 10대 리스크 목록을 관리하고, 각 리스크 별 세부 관리 계획을 수립한다. 10대 리스크 목록이란 프로젝트가 그때그때 직면하고 있는 주요 리스크에 대한 목록을 말한다. 요구사항 분석이 시작되기 전에 예비 리스크 목록을 만들어야 하고, 프로젝트가 끝날 때까지 이 목록을 업데이트해야 한다.

<표 3>은 2005년 전반기에 가지고 있던 10대 리스크 목록의 예이다.

3.9 게임 빌드 공정

프로젝트의 빌드 자동화를 위하여 빌드 스크립트를 제작하는데 다음의 과정을 실행하였다 [GameCode].

- 빌드용 컴퓨터를 정리한다.
- 최신 소스 코드를 가져온다. 이전 단계에서

〈표 3〉 리스크 관리 목록

우선순위	이전 순위	문제사항	문제 제기자	순위 변동	제기날짜
1	new	기존 툴 재사용 (Entity, UI Editor)에 대한 결정이 빨리 되어야 할 듯	AAA		05/03/09
2		일정이 조금 연기가 될 것 같음	BBB		05/02/25
3		툴 제작 인원을 보충할 필요가 있을 것 같음	BBB		05/02/25
4		XX씨 후임이 없음. 그래서 YY가 임시로 맡고 있음 → 신규충원 계획 중(충원대상 : 3D 경력자)	CCC		05/02/23
5		3D 메인 개발자 인력 필요	DDD		05/02/23
6	5	Entity Editor 에 대한 요구사항 수집이 더 필요하며 기간도 더 늘려야 할 것 같음.	BBB	1▼	05/02/25
7	6	서버 쓰레드 처리 문제- 속도가 나올 거 같지 않음. -> 보류	CCC	1▼	05/02/23
8	7	스크립트 언어선정 문제	AAA	1▼	05/02/25
9	8	GUI 툴을 제작하는데 필요한 MFC 경험이 없음.	EEE	1▼	
10	9	서버구조를 좀 더 견고하게 할 필요가 있음	DDD	1▼	05/02/25
11	10	각 툴 담장자에 대한 역량 파악 준비	CCC	1▼	05/02/23
12	11	모니터 교체	EEE FFF GGG	1▼	05/02/25
13	12	팀 분위기 저하에 대한 방안연구 (예)장비지원, 팀 T-셔츠제작, 토요일에 업무가 아닌 경쟁사게임을 다 같이 해보면서 의논하기, 모두 참여하는 친선 게임 개최 등	AAA	1▼	05/02/23
14	13	인력 배치와 이동이 너무 잦은 것 같음	AAA	1▼	05/02/23

전체 프로젝트가 모두 지워졌으므로, 이 단계에서는 저장소에 있는 모든 것이 다운로드 된다.

- 최신 버전 번호를 얻고 빌드에 레이블을 붙인다.
- 프로젝트의 디버그 버전과 릴리스 버전을 컴파일, 링크한다. 결과물은 프로젝트 설정에 등록된 폴더에 저장된다.
- 자동적인 테스트 스크립트를 실행한다. 자동화된 테스트 스크립트는 빌드가 제대로 되었는지 점검한다.
- 빌드 결과를 최종 폴더에 복사한다.

3.10 상세 설계

그 동안 C++를 사용하였지만, 단순한 C++ 문법 사용을 넘어서서 진정한 OOP를 사용하기로 하였으며 설계 도중 이러한 설계 원칙을 지키려고 노력한다.

■ 객체 지향 설계 원칙

- SRP : 단일 책임 원칙(Single Responsibility Principle)
- OCP : 개방-폐쇄 원칙(Open-Closed Principle)
- LSP : 리스코프 교체 원칙(Liskov Substitution Principle)
- DIP : 의존 관계 역전 원칙(Dependency Inversion Principle)
- ISP : 인터페이스 격리 원칙(Interface Segregation Principle)

설계의 악취 및 객체 지향 설계 원칙에 대한 자세한 내용은 [Refactoring][Agile][UML]을 참조한다.

3.11 설계문서를 통해 보는 블루스 프로세스의 실제 적용

여기에서는 요구사항에서부터 실제 바이너리 코드의 시스템이 나오기까지 실제 거상 2에 어떠한 식으로 블루스 프로세스가 적용되었는지에 대해 알아보겠다.

■ 기획자가 제공한 기획서와, UI 흐름도를 통해 설계할 게임을 분석한다.

이 과정에서 기획자는 일종의 고객이 되며, 그 정확한 의도와 요구를 파악하기 위해 긴밀하게 협조하며 얘기하는 것이 필요하다. 실제 기획자뿐 아니라 같이 일하는 프로그래머 동료 등도 기술적인 부분에 있어서 중요한 고객(어느 기술이 구현 가능한지 여부 등에 따라 비기능 요구사항들을 정확히 파악하는데 도움이 된다)이 되기 때문에, 개발에 참여하는 여러 사람과의 대화가 필요하다. 그리고 UI 흐름도의 경우, 실제적인 사용자 경험 모델이기 때문에 전체적인 도메인을 파악하는데 큰 도움을 준다.

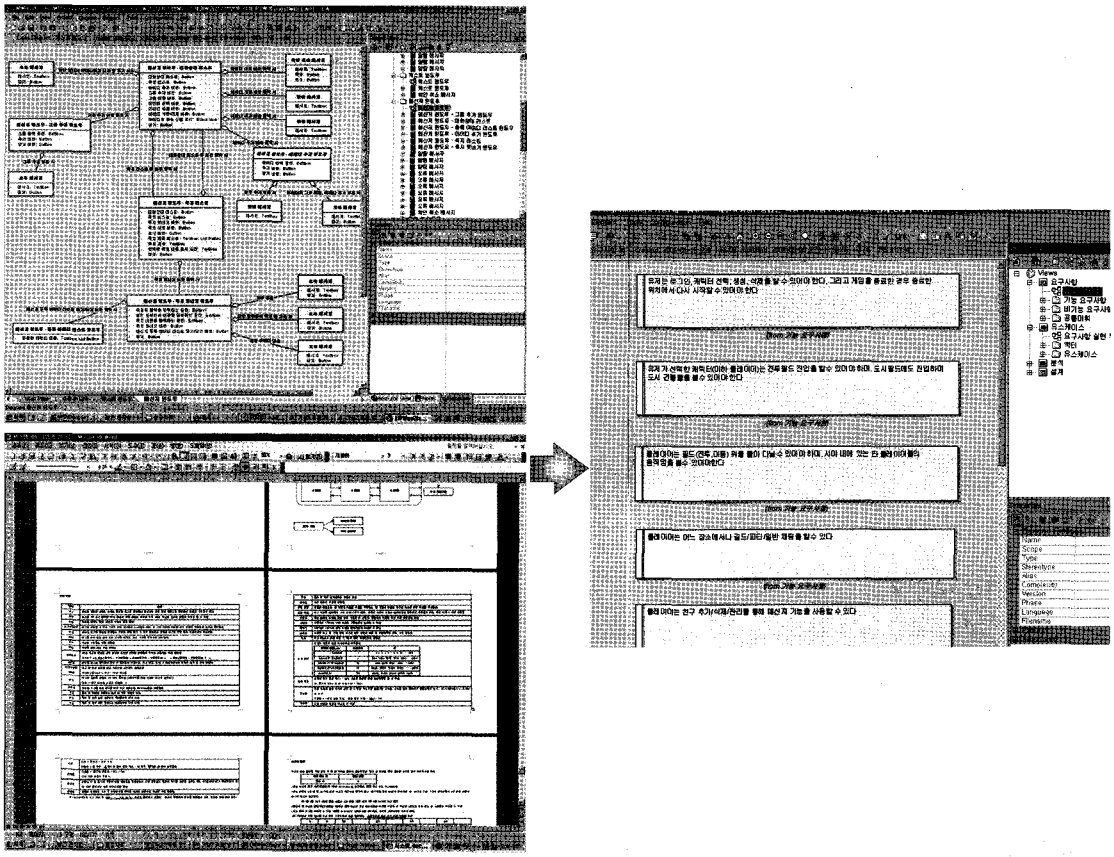
■ 파악된 요구사항을 통해 유스케이스를 추출해 기술한다.

요구사항을 분석하여 유스케이스를 뽑아내는 일은, 그 범위를 정하는 것(유스케이스의 여부)에 따라 이후 설계작업이 결정되기 때문에 매우 중요하다. 유스케이스의 경우 그 수가 크게 늘어나지 않는 범위에서 큰 단위로 묶어 추출하고, 추출된 유스케이스를 그룹화시켜 관리하기 쉽게 한다.

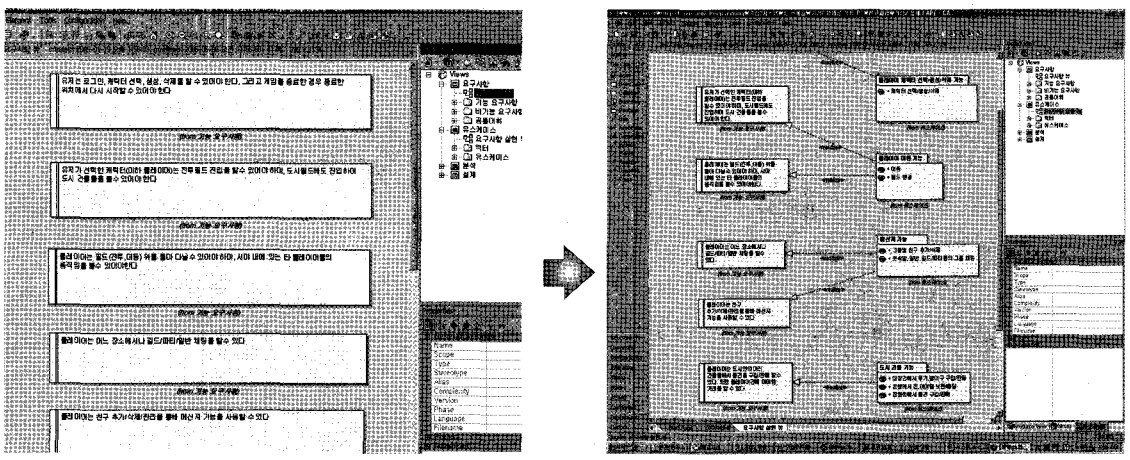
■ 파악된 유스케이스을 분석하여 각각 시퀀스 다이어그램을 그리며 분석 매커니즘, 분석클래스 등을 도출해낸다.

유스케이스를 실현하며, 각 시퀀스간에 전달되는 데이터, 서비스 등을 파악하면, 클래스 후보들이 도출되고 이들을 MVC 패턴에 맞추어 나누어 분석하다 보면, 해결해야 할 메커니즘들이 같이 도출된다. 이 매커니즘들은 비기능 요구사항으로부터 바로 추출되기도 한다. 여기서 아주 세부적으로 알고리즘들을 어떻게 처리해야 할 것인지 정도까지는 내려가면 전체적으로

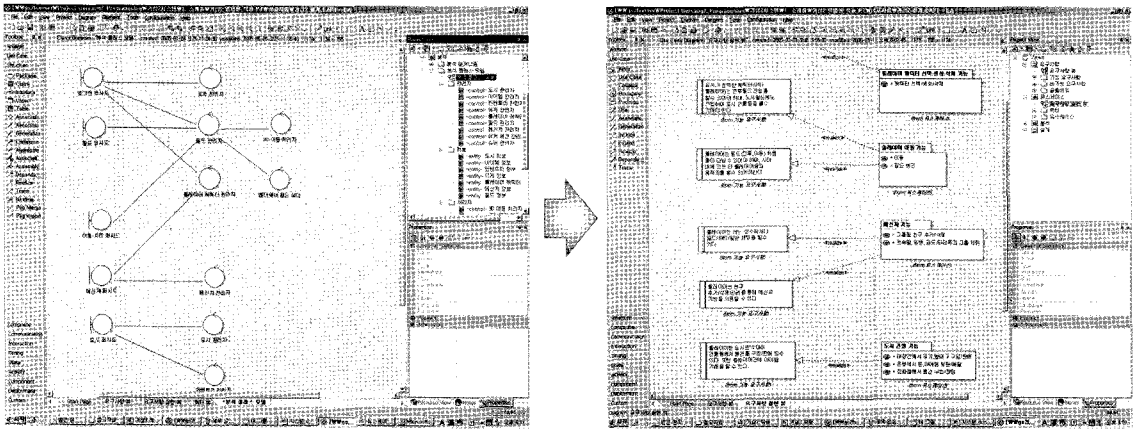
분석단계에서 설계에서 코딩까지 신경 쓰는, 즉 숲을 봐야 하는 사람이 나무를 보는 문제가 생기므로 그 선을 잘 정해야 한다. 전체적으로 MVC를 나누는 기준은 시퀀스 다이어그램을 통해 데이터의 실 소유 여부에 따라 결정한다. 거상 2에서는 Viewer는 Façade로 생각하고 Façade 패턴을 구현하는 것으로 Controller는 Manager, Model은 Info에 매치된다.



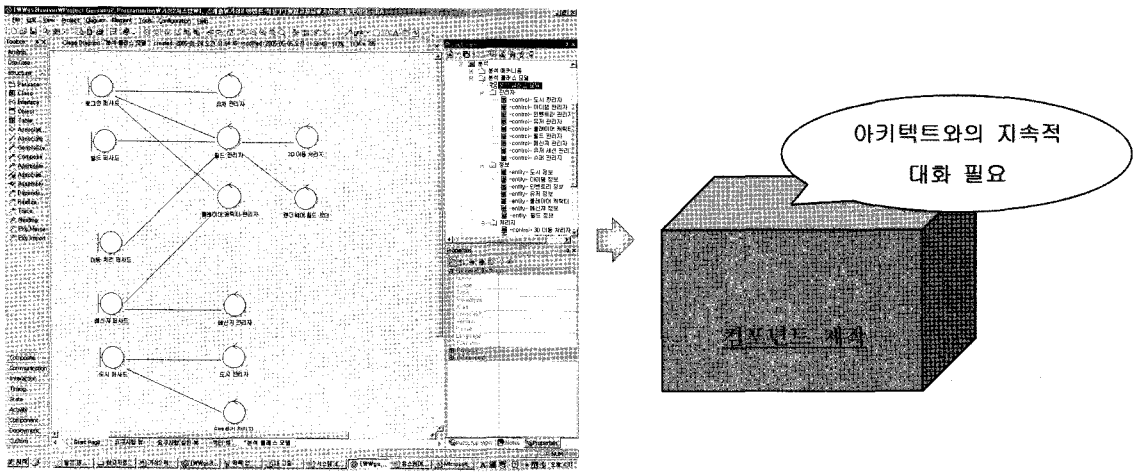
(그림 5) 기획서와 U.I.흐름도를 통한 기능/비기능 요구사항 분석



(그림 6) 요구사항으로부터 유스케이스 추출



(그림 7) 유스케이스로부터 분석메커니즘, 클래스 등 도출



(그림 8) 분석 매커니즘, 클래스로부터 세부 설계 후 코딩

■ 제작해야 될 각각의 부분에 대해 각 담장자가 세부 설계 후 코딩에 들어간다.

이 과정에서는 먼저 분석 클래스로부터 클래스 간에 서로 긴밀하게 협력하는 클래스 그룹을 식별하고, 이들의 결합 정도를 고려하여 컴포넌트를 정의한다. 그리고 이후, 기본 RUP 절차에 따라, 각 컴포넌트의 데이터, 오퍼레이션, 인터페이스 등에 대해 상세 기술해야 하지만, 이럴 경우 설계과정이 상당히 길어지며 위에서 자주 언급했던 요구사항에 급박한 변화가 있을 경우 이런 것들이 모두 헛수고가 되는 문제가 있다. 그래서 거상 2에서는 이 부분을 생략하고, 바로 각 컴포넌트의 담당 개발자에게 이 컴포넌트의 제작에 바로 들어갈 수 있게 한다. 실질적으로 이 부분은 방법론 적용 시 가장 어려운 부분으로서 컴포넌트의 담당 개발자가 아키텍트로부터 받은 상세한 컴포넌트의 명세 없이 작업에 들어가기 때문에, 처음 아키텍트가 의도했던 것과는 다른 산출물이 나오는 문제가 발생할 수 있다. 따라서 먼저 작업에 들어가기 전 이 컴포넌트 제작을 2단계로 나누어, 1단계에서는 전체적인 구조보다 컴포넌트의 내부 알고리즘과 기능구현을 초점 맞추어 제작에 들어가도록 한다. 그리고 상세히 컴포넌트 제작을 하면서 나온 설계적인 문제(실제 담당자는 부족한 요구사항들을 직접 고객과 대화하여 더 얻기도 한다)들을

아키텍트에게 피드백 해주어 수정할 수 있도록 한다. 그리고 1 단계에서 모든 내부 알고리즘과 기능들이 문제없이 작동할 경우 2단계로 넘어가 아키텍트는 앞서 피드백했던 설계 상의 문제점들을 수정하고 컴포넌트 개발자는 실제 전체적인 구조를 생각하며 컴포넌트를 재제작하게 된다. 그리고 이런 중간중간에 계속해서 아키텍트와의 긴밀한 의사소통이 있어야 처음 의도한 설계대로 목적을 완수할 수 있다. 이는 초기의 TOP-DOWN 방식의 설계는 상당한 문제가 있음을 인지하고 TOP-DOWN과 BOTTOM-UP방식이 결합된 하이브리드 형태로 개선하게 된 것이다.

4. 결과 및 고찰

가장 큰 방법론 두 가지의 장점들을 취합하여 게임 개발에 가장 적합한 형태로 변형 시켰다. CBD의 컴포넌트 및 아키텍처 중심 개발과 XP의 단순함을 결합하여 너무 크지도 작지도 않은 방법을 찾았다. 즉, 활동성을 제한 받지 않으면서 아키텍처 중심의 개발을 할 수 있도록 하였다. 개발방법론을 적용하여 얻게 되는 장점 및 단점들은 다음과 같다.

■ Enterprise System 전체를 볼 수 있는 시야를 가질 수 있다.

- 게임 개발을 진행하다 보면 세부적인 개발에 치중하여, 아주 간단하더라도 반드시 지켜야 하거나 개발해야 할 사항들을 잊기가 쉽다. 그리고 이런 일이 반복되다 보면 초기 의도는 사라지고 들쑥날쑥한 구조로 변경되기 시작한다. 그러나 개발방법론에 의거한 설계 방식을 가지게 되면 소프트웨어 아키텍처 기술서를 통해 일관적으로 전체를 볼 수 있는 시선을 가질 수 있게 된다.

■ 요구사항 변경/추가시 그 파급효과를 파악하기가 수월해 변화에 능동적으로 대처 할 수 있다.

- 잘 나누어진 컴포넌트는 이후 요구사항의 변경/추가로 이한 새로운 컴포넌트의 추가나 기존 컴포넌트의 수정, 삭제 시 그 파급효과를 파악하기가 수월해 작은 요구사항 변화에 능동적으로 대처 할 수 있다.

■ 위험요소의 파악

- 여러 단계를 통해 분석된 요구사항에 의해, 쉽게 놓칠 수 있는 선 위험요소들을 미리 파악할 수 있는 한편, 요구를 정확히 완수할 수 있는 이점이 있다.

■ 전체적인 시스템의 안정감

- 모든 장점들 중에서 위와 같은 일련의 절차에서 오는 시스템 구축 시 느껴지는 안정감은 큰 프로젝트일수록 그 효과가 더해지는 이점이 있다.

■ 초기 방법론 적용시에 드는 시간 부하

- 처음 프로세스를 적용했을 때는, 여러 가지 경험 부족으로 현재와 같은 커스터마이징을 거치기까지 많은 시행착오를 겪게 되는데 이는 아직까지 게임개발에 방법론을 적용시켜 프로젝트를 진행 중인 성공케이스가 없기 때문에 더욱 큰 부담으로 다가오게 된다.

■ 개발자의 방법론에 대한 이해

- 방법론에서 가장 중요한 요소로 꼽는 것은 방법론을 적용하기 위한 기반구조 및 문화, 방법론을 위한 활동과 조직적인 역할 및 책임이다. 이를 이루기 위해서는 반드시 모두가 방법론에 대한 이해를 가지고 있어야 하며, 서로가 충분히 여러 가지 개념들에 대해 통일된 사고를 가져야 한다. 그러나 게임 개발분야에서는 아직까지 객체지향 프로그래밍에 대해,

디자인 패턴 보단 알고리즘적인 면과, 어떠한 실질적인 기술이 우선시 되고 있는 것이 현실이다. 따라서 프로세스를 적용하기 전에, 전체 팀원들이 프로세스에 대한 이해가 충분한지를 반드시 검토해야 한다.

'거상 2' 개발과정을 거치면서 향후 '거상 2'에 적용된 방법론이 개선되어가야 할 방향을 정리해 보았다.

■ 자동화 기능 강화

- 현재 '거상 2'는 블루스 프레임워크가 지원하는 실제 프로젝트를 생성하고 오퍼레이션을 추가하는 컴포넌트 자동화 툴의 완성도의 부족으로 RPC 부분을 적용하여 완전한 분산 컴포넌트 환경을 구축하는 것을 실현하지 못하고 있다. 그래서 이후 이 자동화 툴을 개선하여, 이후 추가될 에피소드나, 패치에 완전한 분산컴포넌트가 적용된 컴포넌트 환경을 구축해야 할 것이다.

■ 각 설계 담당자의 확실한 역할 분담

- 일반적으로 RUP 및 CBD에서는 각 단계의 설계 담당자를 각각 두어 시스템을 바라보는 뷰의 입장을 분배하게 된다. 예를 들어 비즈니스 분석가는 요구사항을 분석하고 파악한다. 어플리케이션 아키텍트는 각 유스케이스를 추출하고 초기 아키텍처 개요를 작성하며, 후보 비즈니스 컴포넌트를 도출한다. 기술 아키텍트는 요구사항 및 유스케이스, 초기 아키텍처를 통해 분석 매커니즘을 식별하고 실제 컴포넌트들의 기반환경이 될 프레임워크를 설계한다. 이는 매우 중요한 점으로, 이전과 같은 워터폴 방식으로, 즉 요구사항의 변화가 적은 시스템일 경우 한 명의 아키텍트가 단 하나의 관점으로 시스템을 주시하여도 문제가 발생하지 않지만, MMOG와 같은 엔터프라이즈 시스템일수록 하나의 관점으로 시스템을 바라보는 것은 큰 위험요소가 될 수 있다. 하지만 현재는 아키텍트 인력부족으로 실제적으로 소수의 아키텍트가 전체를 관여하는 경우가 많다. 그렇기 때문에 이후, 팀원들 간의 아키텍처 교육 등을 통해 여러 사람이 각자의 관점으로 시스템을 바라볼 수 있게 개선해야 할 것이다.

참고문헌

[1] [SPSG] Steve McConnell, Software Project Survival Guide, Microsoft Press, 1997.

[2] [RD] Steve McConnell, Rapid Development, Microsoft Press, 1996.

[3] [PSD] Steve McConnell, Professional Software Development, Addison Wesley, 2003.

[4] [F-Book] Robert L. Glass, Facts and Fallacies of Software Engineering, Addison Wesley, 2002.

[5] [WB] Tom DeMarco, Waltzing with Bear, Dorset House, 2003.

[6] [Peopleware] Tom DeMarco, 피플웨어, 매일경제신문사(매경출판주식회사), 2003.

[7] [RUP] 조창현, RUP Development Solution, 영진.COM, 2004.

[8] [XPI] eXtreme Programming Installed, Addison Wesley, 2000.

[9] [Pragmatic] Andrew Hunt, David Thomas, The Pragmatic Programmer, Addison Wesley, 2000.

[10] [Agile] Robert C. Martin, Agile Software Development, Prentice Hall, 2002.

[11] [GameCode] Mike McShaffry, Game Coding Complete, Paraglyph Publishing, 2003.

[12] [TDD] Kent Beck, Test-Driven Develop-

ment by Example, Addison Wesley, 2002.

[13] [CodeComplete2] Steve McConnell, Code Complete 2nd, Microsoft Press, 2004.

[14] [Refactoring] Martin Fowler, Refactoring, Addison Wesley, 1999.

[15] [WEUC] Alistair Cockburn, Writing Effective Use Cases, Addison Wesley, 2000.

[16] [UML] Robert C. Martin, UML for Jave Progrmmer, Prentice Hall, 2003.

[17] [CBD1] 전병선, .NET Enterprise System 객체지향 CBD 개발 방법론, 2004.

[18] [CBD2] 채흥석, 객체지향 CBD 개발 바이블, 2003.

[19] [COM] 전병선, Component Development With Visual C++ & ATL, 2004.

저자약력

최 일 곤

1984년 한양대학교 기계공학과(학사)
1988년 한양대학교 기계공학과(석사)
1996년~현재 한양대학교/대학원 기계공학과 출강
1999년 한양대학원 기계공학과(박사)
2000년~2002년 한국원자력연구소
2002년~현재 (주)조이온 부사장/CTO
관심분야 : 전산역학, 게임개발방법론, 게임엔진(실시간 렌더링, 물리 엔진 등)
e-mail : justyou@joyon.com