비디오 게임 개발을 위해 스크럼 방법론을 사용할 때 빠지기 쉬운 열 가지 함정 (Top 10 Pitfalls Using Scrum Methodology for Video Game Development)

Paul Miller

가마수트라 등록일(2008. 7. 15)

(http://www.gamasutra.com/view/feature/3724/top_10_pitfalls_using_scrum_.php)

[업계에서 오랜 경험을 쌓아 온 Miller는 게임 개발을 위해 주된 애자일(Agile) 방법론에 주목하였으며, 비디오 게임 프로젝트를 관리하기 위해 스크럼(Scrum)을 사용하는 경우에 대하여, 열 가지 주요 함정들과 이를 극복하기 위한 방법들을 제시한다.]

게임 개발을 위해 스크럼을 사용하는 것과 관련하여 일부 과장된 광고가 있었는데, 그 가운데는 게임 개발자 회의에서의 프레젠테이션도 포함되어 있다. 스크럼에 대해 기술하는 서적들과 많은 인터넷 리소스들이 있으며, 본 기사는 독자가 제품 개발을 위한 스크럼과 애자일의 방법론들에 익숙해 있을 것을 가정하여 기술되었다.

스크럼이 여러 종류의 소프트웨어 프로젝트에 기여할 수도 있겠으나, 비디오 게임 프로젝트를 관리하기 위해 스크럼을 채택할 때에 빠지기 쉬운 몇몇 함정들도 있다.

잘 확립되어 있는 기존의 최선책들의 중요성이 무시됨으로 인해 몇몇 해로운 요소들이 발생할 수 있다. 스크럼이 기존의 최선책들을 대체하여 사용함으로 인해 초래되는 결과들도 있다. 여기 스크럼 방법론을 사용할 때 프로젝트에서 나타나는 10 가지 함정을 소개한다.

10. 백로그 스프레드시트로 교체함에 따라 더 이상 게임 설계 문서(Game Design Document; GDD)는 필요하지 않다.

GDD 와 백로그(backlog)를 둘 다 관리하는 동안에도, 팀은 과도한 서류작업에 매일 시달리고 있다. 백로그를 담당하는 프로듀서 또는 스크럼마스터는 결국 GDD 를 관리하고

작성하는 데 결국 더 적은 시간을 소비하게 되며, 그 결과 제품 설계 사양의 결핍이 야기된다. 팀원들은 어떤 코드를 써야 하는지, 어떤 구성요소들이나 자산을 체계화하여 게임을 만들어야 할지를 모르게 된다.

교훈점: 백로그 스프레드시트는 프린터-친화적인 게임 설계의 문서화를 위한 대체 수단이 아니다. 팀 협력은 제품 설계 사양을 문서화하는 사람을 대체할 수는 없다.

백로그 스프레드시트의 프린터 친화성 부재로 인해 사실상 의사전달 능력이 억제되고 있는지는 처음부터 식별하기가 힘들다. 백로그 스프레드시트 이전에, GDD 를 인쇄하고 이를 팀 회의에 가져가 그 위에 간략하게 글을 남길 수 있었다. 서면으로 피드백을 남기기도 더 쉬웠다. GDD 를 인쇄하여 회의에 가져감으로써, 설계 내용에 대해 이야기할 수 있고 컴퓨터 앞에 앉지 않고도 이를 확인할 수 있었다.

설계 문서의 대체 수단으로 백로그 스프레드시트를 사용하는 동안, 팀원들이 깨달은 한 가지 문제점은 회의 후에 각자의 사무 책상으로 돌아가서 업무에 집중하고 나면 회의 중에 논의한 여덟 가지 요점 중 일곱 가지는 잊게 된다는 것이었다. 제안된 해결책은 팀 회의에 모두가 노트북을 가져가 팀원들이 동시에 스프레드시트에 입력할 수 있게 하는 것이다. 팀 회의가 LAN 파티가 되어, 모두가 각자의 스프레드시트에서 서로의 커서를 볼 수 있다고 상상해 보라.

안타깝게도, 스프레드시트는 멀티플레이어 모드를 갖추고 있지 못하며, 회사는 단순히 스크럼 방법론을 위해서 모두에게 노트북을 구매해 주지는 않을 것이다. 확실한 해결책은 모두가 회의에 메모지와 펜을 가져와서, 필기를 하는 것인데, 그렇다면 왜 처음부터 설계자는 설계 내용을 서면으로 작성하지 않았는가? 왜 설계를 그처럼 바쁘게 수행하고, 왜 항상 여러 비서들에게 편지 내용을 받아 적게 만드는 경영자처럼 모두가 필기를 하게 만들어야 하는가?

다중 팀 프로젝트 검토 회의에서, 스크럼마스터는 “팀 내에 더 원활한 의사소통이 필요하다”고 말하였다. 그러자 모든 개발자들이 이리저리 훑어보며, “설계 사양서는 어디 있는가? 프린터 친화형 GDD 는 의사소통을 위한 것이 아니었는가?"하고 묻는다.

최선책: 스크럼이 있기 전에는, 마이크로소프트 워드에 GDD 를 작성하여 이를 소스 관리(source control)하는 것이 최선책이었다. GDD 로서 wiki 가 소스 관리하는 워드 문서보다는 아마도 더 나을 것이다.

9. 15 분간 일일 스탠드업 회의는 사람들이 하던 일의 맥을 끊는다.

모두의 아웃룩 달력에 일일 15 분 스탠드업 회의 일정 메모를 적어두고, 스크럼마스터는 회의에 2 분 늦게 도착하여 10 분 늦게 도착하는 팀원들을 정중하게 기다린다.

질문: 어떻게 6 개월 프로젝트가 예정보다 1 년이 늦춰질 수 있는가? <Mythical Man Month>라는 책은 읽지 않았다 하더라도 그 대답은 어렵지 않다. 한번에 하루씩 늦춰진 것이다. 15 분간 매일 회담을 갖는 것은 집중해서 여러 시간을 끊이지 않고 지속하면서 생산성을 재촉하는 것만큼 부담감이 더하지 않다.

기술적인 일에 몰두하고 있는, 스크럼의 “헌신적인 돼지들”로도 알려진 프로그래머들을 단순히 “일이 진전이 안 되는가?”하고 묻기 위해서 회의실로 불러들여 방해하는 것은 전체 프로젝트의 계획을 틀어지게 하는 것일 뿐 아니라 반나절 분량의 작업을 망쳐놓기 좋은 방법이다. 그 대화란 것이 보통은 다음과 같이 진행될 것이다:

스크럼마스터: "일이 진전이 잘 안 되는가?"

엔지니어: "아닙니다. 단지 “S”자 곡선을 삽입하기 위해서 처음 Hermite 의 기본 기능을 사용해 화면 상에서 위젯 슬라이드가 부드럽게 진행되도록 하려고 하고 있는 중입니다."

스크럼마스터: "난 기술자가 아니라 무슨 말인지 잘 모르겠는데, 좀 쉽게 얘기해 주겠나?"

엔지니어: "애니메이션이 자연스러우면 UI 가 더 보기 좋을 겁니다."

스크럼마스터: "의견서는 작성했나?"

엔지니어: "내일 하려고 합니다."

교훈점: 15 분 일일 스탠드업 회의는 팀원들이 제품 선적에 중요하지 않은 태스크를 처리하느라 하루를 소진하지 않도록 하기 위한 마련이었다. 만약 스크럼이 있기 전에, 개발자들이 이미 다양한 프로젝트를 진행 중이었고, 정확한 태스크를 수행하고 제품을 정시에 선적하는 추적 기록 시스템을 갖추고 있었다면 어떻게 할 것인가? 15 분 일일 회의가 어떻게 차질이 생기지도 않은 어떤 일을 조정해야 한다는 말인가? 대답은 그럴 필요가 없다는 것이다.

교훈점: 10 분 늦게 시작하는 회의를 위해 15 분짜리 메모를 작성하는 것은 팀원 모두에게 25 분씩 낭비하는 것과 같다. 회의 후에 개발자들이 각자의 자리로 돌아가 회의 전에 하던 일에 집중하느라 5 분에서 15 분을 더 소비해야 한다고 생각해 보라. 15 분 회의 시간은 하루 8 시간을 근무하는 동안 낭비되는 약 40 분의 주의산만 시간에 추가될 뿐이다.

개발 주기에서는 이것이 몇 주의 시간이 될 수도 있다. 문제는 이것이다: 이 시간이 정말 잘 사용되었다고 할 수 있는가? 일일 회의는 의도한 바를 잘 달성하는가? 그 유익이 비용을 충당할 만큼의 가치가 있는가? 같은 일을 시간을 들이지 않고 달성할 수 있는 다른 방법은 없는가?

엑셀 스프레드시트에 잔무를 버그 데이터베이스와 구분 짓는 것은 두 가지 별개의 도구에 해야할 일 목록을 나누어 담게 하여 일의 양을 늘리는 결과를 초래한다. 분명 개발 태스크와 결함은 모두 같은 종류의 데이터이다. 다시 말해, 데이터베이스 소프트웨어는 태스크와 결함을 구분하지 못한다.

태스크와 결함의 유일한 차이점은 태스크가 수행되기 전에 있다는 것과, 결함은 수행 후에 있고 이를 재생하기 위한 단계들을 포함한다는 것이다. 태스크와 결함이 동일 데이터베이스의 프로젝트 관리 도구에서 한번에 관리해서는 안 될 이유는 없다.

최선책: 일부 개발 태스크들은 완료하는 데 몇 분 밖에 소요되지 않는가 하면, 어떤 태스크들은 며칠이 소요되기도 한다. 스크럼 마스터와 매일 같은 대화를 나누는 것 보다는 모든 태스크를 하나의 태스크 관리 데이터베이스에 통합하는 것이 더 나을 것이다.

태스크 데이터베이스는 어떤 태스크가 오늘 처리되어야 하는지를 보기 위해서 일종의 폴더 또는 쿼리 메커니즘을 갖추고 있어야 한다. 팀 내 각자는 자신의 폴더를 보유할 수 있으며, 각 폴더는 “WorkComplete", "WorkInProgress" 및 "WorkToDoNext" 등으로 불릴 수 있다. 각 팀원은 태스크를 바꿀 때마다, 또는 개발 태스크를 마쳤거나 버그를 수정했을 때마다 자신의 폴더를 업데이트한다.

팀 내 각자는 자신의 기기를 사용하여 모두의 폴더를 볼 수 있으며, 따라서 다른 팀원들이 무슨 작업을 하고 있는지를 알 수 있다. 팀의 의견서나 남은 시간은 각 태스크 항목에 기재할 수 있다. 팀원들이 오늘 무슨 일을 하고 있는지를 알고자 하는 사람이 있다면, "WorkInProgress" 폴더를 살펴보기만 하면 된다. TestTrack, DevTrack 및 FogBugz 는 이를 지원할 수 있는 기능들을 포함하는 세 가지 상용 출하대기, 결함 추적, 및 프로젝트 관리 데이터베이스 도구들이다.

스크럼 태스크 보드를 가시화할 가치가 있다고 보거나, 아직도 Gant 차트를 사용하고 있다면, 데이터베이스에서 모두의 폴더와 태스크를 열 수 있고 (Gant 차트에서와 같이)

수평 시간 축에 태스크 관련 데이터의 비환식 그래프 상의(스크럼 보드의 태스크 카드처럼) 노드로 직무를 표시할 수 있는 작은 그래픽 프로그램을 만들 것인지 고려해 볼 수 있을 것이다.

그런 다음 프로젝터를 사용하여 벽면에 내용을 표시할 수 있다. 소견서와 남은 시간에 대해(아바타 상단의 건강 바와 같이) 각 태스크에 작은 진행상황 바를 추가할 수 있다고 상상해 보라. 팀원들이 책상에 앉아 태스크를 완료했다고 할 때에(윈도우 익스플로러에 쓰레기통 속에 들어가는 파일의 애니메이션처럼) 애니메이션 효과를 넣을 수도 있을 것이다.

팀 내에 100 명의 개발자들과 연결되어 있을 경우, 이 애니메이션 효과가 마치 폭죽처럼 펼쳐지는 모습을 상상해 보라. 개발자들의 실제 작업시간이 산정 시간과 거의 일치함으로써 경험 포인트가 3 점 상승하고, 작업 과정을 마치면서 수준도 향상될 것이다. 경영자들 역시 팀워크와 생산성을 눈으로 실시간 확인할 수 있음에 기뻐할 것이다.

무엇보다도, 이러한 전자 스크럼 보드는 5 명의 개발자로 구성된 소규모 팀에서부터 250 명의 게임 개발자 규모에 이르기까지 그 크기가 다양하다. 대형의 그래프 시각화를 위해 쌍곡선 형태의 어안 조망 방식으로 부드럽게 확대/축소하는 것도 잊어서는 안 된다.

8. 팀원 전체를 전용 팀 회의실이나 팀 구역으로 불러들이는 것은 서로 다른 분야들을 한 데 결합시키는 타화수분 방법론과는 다르다.

엔지니어들을 나란히 앉혀 두고 문제에 대한 해결책을 함께 고민하게 하는 것은 좋은 방법이다. 엔지니어들이 컨텐츠 제작자와 나란히 앉게 하여 엔지니어들이 고객들을 위한 저작도구를 지원할 수 있게 해주는 것도 좋을 것이다. 개발자들이 서로 협업하기 위해 하루에도 여러 차례 직접 사무실을 걸어서 찾아가야 한다면 일의 능률은 그만큼 떨어질 것이다. 여기 문제점 중 하나는 책상과 칸막이 벽이 이동식이 아니라는 점이다.

교훈점: 협업을 원한다면, 명령을 하지도 방해를 하지도 말아야 한다. 대신 우선 억제 요소를 제거함으로써 협업이 가능하게 한 다음, 자연스럽게 또는 필요할 때 협업이 이루어지게 하는 것이다.

최선책: 협업이 방해를 받지 않게 하는 한 가지 방법은 우선 책상에 바퀴를 다는 것이다. 둘째로, 모두에게 중단 없는 파워 서플라이와 무선 네트워크 접속을 제공한다. 다음으로는 칸막이 벽 일부를 제거할 수 있는지 확인해 본다. 팀원들은 이제 기기를 끄고 재부팅할 필요도 없이 책상을 옮겨서 개방된 대기작업 영역으로 이동할 수 있게 되었다.

따라서 반나절을 얼굴을 맞대고 협업함으로써 장비와 연결 케이블을 옮기고 리부팅을 하는 동안 방해를 받을 염려는 없어졌다. 분명 모두가 중단 없이 전력을 공급받을 수 있는

장비를 고가일 것이다. 그러나 생산성 없는 시간, 생산성의 억제, 데이터 손실, 장비 손상 등도 비싼 값을 치르기는 마찬가지이다.

최종적으로, 개발자들에게 개발 태스크를 정시에 완수하는 것이 중요하다고 말하고, 작업 완수를 위해 책상과 컴퓨터는 원하는 곳으로 이동할 수 있다고 말해 준다. 그 결과 개발자들은 책상을 필요에 따라 서로에게 가까운 곳으로 이동해야 할 것이며, 이는 굳이 전문적인 프로젝트 관리에 따른 명령이 아닌, 자발적으로 필요에 따른 조정이 될 것이다.

좀 더 편안한 조건에서, 각각의 개발팀과 집단은 지정된 팀 구역과 이동형 개인 화면, 팀 회의실을 가질 수 있다. 바퀴가 달린 대형 화이트보드를 제공하는 것도 잊어서는 안 된다.

7. 한 팀원이 말하길, "내가 일이 제대로 되고 있지 않은 것은 해야 할 일이 잔무 목록에 올라 있는데, 반드시 먼저 프로젝트 매니저가 내게 지정을 해줘야 진행할 수 있기 때문이다."

재차 확인되는 교훈점: 이처럼 적극적인 사람은 다음에 해야 할 일을 찾아서 하는 유형으로, 고객에게 모든 것이 제대로 되고 있는지 묻고, 이제 무엇을 할 것인지를 생각하여 다음 달 업무 부담을 줄일 줄 안다. 이런 경우, 두 주 분량의 스프린트(sprint) 작업을 전력으로 끝내 놓고는 다음 두 주 작업을 받을 때까지는 아무 일도 하지 않아도 된다고 믿는다.

교훈점: 전체 팀은 단지 한번에 두 주 분량을 작업에 초점을 맞출 것이며, 지정된 경우가 아니라면, 직무를 더 수행할 수 없다고 말하는 것은 사람들이 수동적이 되게 만드는 길이다. 모든 팀원이 전체 흐름을 파악할 수 있게 해주고 세세하게 관리하지 않고서도 전체 흐름의 구성요소들과 세부점들에 대한 책임을 이행하도록 독려해 주는 것이 중요하다.

최선책: 스크럼 스프레드시트에 태스크를 추가하면 사소한 세부점들에 대해서도 승인을 얻어야 하는 것처럼 느낄 것이다. 개발자들은 프로젝트에 대한 소유권과 책임을 모두 떠맡는다. 그들은 다음달에 무슨 일을 해야 할지를 예견해야 한다. 개발자들은 일정에 앞서서 직무를 능동적으로 수행할 수 있으며, 직무를 할당하지 않고도 고객 서비스를 제공할 수 있다.

6. 팀원 모두가 구성되기 전에 스토리와 스프린트 관리를 시작한다. 자신의 스크럼 스토리와 스프린트가 프로젝트의 QA 테스트를 하기 수 개월 전에 이미 100% 완수하겠다고 결심한다.

재차 확인되는 교훈점: 프로젝트 첫 달에 QA를 갖지 않고, 죽음의 행진과 위기의 시간을 갖게 될 것으로 마음먹고 있을지 모른다. 프로젝트 초반에 QA를 추가하는 것은 그저 중요한 정도가 아니라, 필수적인 미션이다.

최선책: 개발 주기 전반에 걸쳐서, 엔지니어들과 컨텐츠 제작자들이 버그를 수정하게 하여 전체적인 버그 수를 항상 최소화하는 것이 좋다. 개발 주기 초반에 프로젝트에 대한 QA를 가져서 버그를 보고하고 이를 확인 및 제거해야 한다. 이것은 개발 주기 말엽에 압도적으로 많은 버그를 수정하느라 보내는 시간을 줄이기 위해 매우 중요한 과정이다.

매일 모든 개발자들이 드러난 결함의 수를 살펴보고 자신의 현재 직무 목록을 살펴볼 수 있다. 직무보다 더 많은 수의 결함이 있을 경우에는 버그수가 너무 많은 것이므로 다음 직무로 넘어가기 전에 먼저 당일 중 상당 시간을 버그 수를 최소화하는 데 투자해야 한다.

더 나은 방법으로, 팀 내 각 사람이 고객을 상대하여 고객들과 팀 동료들에게 버그 수정을 막는 어떤 문제들이 있는지를 물어보는 것이다. 일부 버그는 다른 사람들이 수정하고 있는 버그에 따라서 수정이 불가능할 수도 있음을 인정해야 한다.

5. 스크럼 함정: 15 분간의 일일 스탠드업 회의는 돌아가며 각자 상태를 보고하는 수단 정도로 퇴색해 버리기가 쉽다.

일일 회의에서는 활동 항목이 없으며, 사실상 생산성을 향상시키는 데 도움이 되지 않는다. 일일 회의에서, 아마도 스크럼마스터는 팀원들이나 팀의 활동이 “활발하지 않다”고 말하는 것이 전부일 것이다.

교훈점: 단순히 스크럼을 위해서 스크럼을 하는 것에는 아무런 가치도 목적도 없다.

재차 확인되는 교훈점: “활동 항목 없이 원탁에 둘러앉아 보고하는” 회의는 그저 시간 낭비일 뿐이다. 모두가 소비하는 시간을 합하면 회의 주최자의 시간을 덜어주는 것을 제하고도 분명 낭비되는 시간이 더 많을 것이다.

최선책: 모두의 주당 40 시간 대역폭 통로(bandwidth pipe)를 극대화하는 것이 하루 15 분 회의보다 더 중요하다. 팀원 한 명이 일의 대부분을 맡고, 다른 사람들이 다음 날의 작업이나 다음 두 주의 스프린트를 기다리고 있다면, 이 프로젝트 혹은 프로젝트 관리 상 분명 문제가 있는 것이다. 작업량은 반드시 고르게 분배되어야 한다.

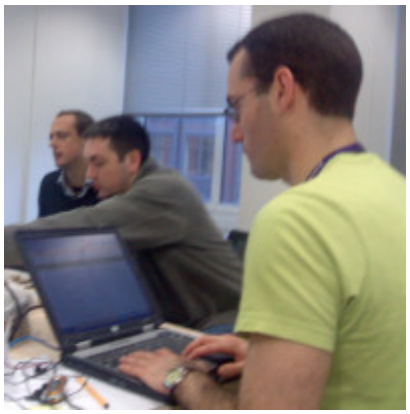
최선책: 모든 회의에 의제가 있어야 하며, 의제가 회의 초대장에 표시되어 있어야 한다. 활동 항목 가운데는 수행될 직무, 직무를 수행할 사람의 이름을 포함해야 하며, 마감일이나 적어도 시간 프레임이 기록되어 있어야 한다. 예를 들어 보자: "Joe는 다음 날 마칠 때 즈음에 소스 제어에 포함시킬 캐릭터 모델에 플레이스홀더(placeholder)를 사용하여 표시를 해 둘 것이다. 이렇게 하여 설계자들과 엔지니어들이 이번 주에 이 내용으로 작업을 할 수 있다."

회의 후에 누군가를 위한 활동 항목이 없다면, 개발자들이 하던 일을 중단하고 회의실에 소집해야 할 아무런 목적도 없는 것이다. 상황 보고나 다른 의사소통은 이메일이나 메시지 또는 전화로도 얼마든지 가능하다.

4. 스크럼을 의무화하라.

주주들과 경영자들은 개발팀에 대해 스크럼을 의무화하고 정량적인 프로젝트 완수의 예측 자료를 기대한다. 매니저들은 팀원들에게 피드백 조차 제공하지 못하며, 또는 개발자들이 한 벌의 카드에서 숫자를 선택하여 스크럼의 규모를 조절하는 것이 실제로 무슨 가치가 있는지조차 알려주지 못한다.

더 심각한 상황도 있다: 경영자들이 단순히 프로젝트 매니저의 직감에 의존해 아직 정량적인 예측 데이터를 확보하지 못한 상황에서 이 과정 자체가 무가치한 것이다. 어쩌면 경영자들은 팀원들이 아무도 관심조차 갖지 않는 정량적 예측 데이터를 얻으려고 스크럼 스토리의 크기를 정하는 우스운 숫자 맞추기 연습이나 하는 데 왜 시간을 소비하고 있는지 도무지 이해하지 못할 수도 있다.



교훈점: 의사소통은 양방향으로 이루어져야 한다. 다시 말해 명령 체계의 상하 모두로 전달되어야 한다는 말이다. 주주와 경영자들의 피드백과 가시성은 주주들에게 프로젝트 현황에 대해 보고하는 것만큼이나 중요하다.

최선책: 업무의 이정표가 제시간에 확정되거나 게임이 제시간에 발송되면, 주주들과 선임 경영진과 심지어 경영자들도 각 구획들을 돌아다니며 조직 체계 말단에 있는 개개의 팀원들에게까지 이렇게 말해야 한다. “이보게, 이번 업무의 이정표를 제시간에 완수했다는 보고를 들었네, 정말 수고 많았어! 이것이 바로 이 회사에서 바라는 바야! 정말 잘 해냈구먼!"

조직 체계 정상에서부터 전해오는 긍정적인 격려의 메시지는 MMO 에서는 누구나 얻을 수 있는 어떤 금보다도 더 가치가 있다. 경영진과 경영자들에게 당부한다: 생산성 향상을 원한다면, 긍정적인 말 몇 마디 해주는 시간이 직원들이 거대한 좀비 거미들을 쏘아 맞추거나 레벨 업을 하느라 시간을 보내게 내버려 두는 것보다 더 가치가 있고 더욱 만족을 주는 일이라는 것을.

이정표가 제시간에 마련되었을 때, 제품 개발의 VP 와 어떤 일이 있어도 결코 자신의 700 평방피트 되는 사무실 밖으로 나올 것 같지 않던 마케팅 VP 가 개인적으로 공로를 치하해 주고 보상을 해 준다고 상상해 보라. 그런가 하면 제시간에 이정표를 마련했어도

통지조차 하지 않고 노고에 대해 관심도 가져주지 않는다면 어떻겠는가? 성공이 더 큰 성공을 부른다! 사기가 낮으면 직원들이 적극적이 되게 하기는 그만큼 더 힘들기 마련이다.

3. 2 주일 분량의 작업만 수행하고, 게임 설계에 대해 현재 알고 있는 것에 기초한 코드만 작성한다.

직무를 수행하도록 지정되지 않았다면, 당연히 수행할 수가 없다. 게임 설계가 개발 주기 동안 내내 바뀌게 될 수도 있다는 염려는 접어두라.

재차 확인되는 교훈점: 상향 코드 설계는 하향 코드 설계만큼 좋을 수 없다. 스크럼, 애자일, 및 익스트림 프로그래밍은 상향 코드 설계를 장려하고 촉진한다. 전체 맥락에 대한 설계를 문서화하지 않았을 경우, 한 번에 하나씩 수행한 다음 고객의 피드백에 따라 수정해 가는 방식이 되고 만다.

설계 상의 변화로 인해 어떤 코드는 포함시키고 어떤 코드는 버릴 것인지에 대한 예측이 없기 때문에 코드가 유지되게 하는 데 많은 노력이 기울여지지 않는다. 계획되지 않은 기능을 첨가하면 많은 비용이 들 수 있다. 매번 임시 방편으로 리팩토링(refactoring)하려면 많은 시간이 소요된다.

최선책: 게임의 충실도는 낮은 상태에서 높은 상태로 진전되도록 해야 한다.충실도가 낮은 버전에는 모든 기능의 원형이 들어 있다. 코드는 하향식으로, 발송될 제품에 들어갈 모든 기능들에 대해 이해하는 가운데 체계화할 수 있다. 코드와 컨텐츠는 다르다는 것에 유의하라. 아마도 게임 컨텐츠는 몇 번이고 되풀이할 수 있기를 바랄 것이다.

2. "스프린트 제로"는 사전 제작 시간이다.

모든 스토리를 스크럼 백로그 스프레드시트에 포함시켜 사전 제작을 시작한다. 사전제작은 2 주 간만 지속된다. 스프린트 1 이전에 컨셉 아트가 아직도 없다는 염려는 무시한다.

교훈점: "스프린트 제로(Sprint Zero)"는 사전 제작이 아니다. 창조성과 혁신은 관리할 수 있는 종류의 요소가 아니다. 에너지 음료를 마시며 회의실에 앉아서도 브레인스토밍은 할 수 있겠지만, 종종 최고의 아이디어는 직장에서 집으로 향하는 차 안에서 교통 체증을 겪는 가운데 떠오르기도 하는 법이다. 누군가 순간적으로 가장 창조적인 게임 아이디어를 떠올릴 것으로 요구할 수는 없다. 백로그 스프레드시트에 게임 아이디어를 써내라고 강요하는 것은 도움이 되지 않는다.

최선책: 어떤 대단한 제품이든 사전 제작에는 많고 많은 원형 제작 과정이 수반되기 마련이다. 수 백 가지의 원형들이 있다고 생각해 보라. 아마 대량 생산을 하고 싶을지도

모르겠다. 어떤 아이디어는 작은 노란 종이 위에 낙서를 한 정도에 지나지 않을 것이다. 어떤 원형들은 게임을 하는 동안 서둘러 컨텐츠를 수정하는 방법과 수 백 가지의 튜닝용 손잡이가 있는 모든 기능이 갖춰진 소프트웨어 데모일 수도 있다.

기술자가 아닌 평범한 사람들도 매일같이 반복 사용할 수 있을 만한 강력한 원형 제작 도구가 필요할 것이다. 사전 제작에는 시간이 든다. 제작팀이 만들어 곧바로 시연할 수 있는 원형이 많을수록, 더 나은 제품이 만들어진다.

1. GDD가 작성되기 전이라도 태스크의 우선순위를 정하고 시간 산정표를 요청함으로써 프로젝트 관리를 시작한다.

두 주 마다 스프린트 계획 회의 중에, 설계를 하기도 전에 태스크의 우선순위를 정하는 동일한 행동을 반복한다. 이것은 설계서류나 컨셉 관련 문서도 없이 허가자와 생산 계약에 서명하는 것만큼이나 무모한 일이다. 최악의 경우는 변경하는 데 팀이 들일 비용은 고려하지도 않은 채 전체 생산 과정 중 언제든 설계 내용을 변경할 자유를 허가자에게 부여하는 것이다.

교훈점: 프로젝트 관리는 게임 설계를 앞서 갈 수 없다.

최선책: 게임 설계 서류의 초안을 작성하고 엔지니어들, 컨텐츠 제작자들, 및 출판업자와 허가자들로부터 피드백을 얻는다. 이는 반드시 전체 생산에 대한 서약을 하기 전에 이행해야 한다. 태스크의 우선순위를 정하고 시간 산정표를 요청하기 전에 먼저 엔지니어들과 컨텐츠 제작자들이 제품 설계를 이해하고 있어야 한다.

최선책: 우선 건강을 생각하여, 머리 속에 점검목록을 만들어 두고, 개발 과정 중에 올바른 순서대로 일을 진행하는 것이 중요하다. 많은 개발자들이 죽음의 행진 프로젝트가 있을 것임을 직감한다. 만일 적절한 행동을 취하지 않을 경우 일이 잘못될 것이라는 직감이 들면, 실제로 그렇게 될 확률이 크다. 스크럼 컨설턴트의 조언을 듣고 게임 개발에 대한 자신의 직감을 믿는 것이 좋다. 스크럼 컨설턴트는 게임을 만들지 않는다. 대신 하루 종일 스크럼마스터에게 자격증 훈련 수업을 하고 돈을 번다.

결론

스크럼 및 애자일 방법은 프로젝트 관리 도구상자에 있는 도구들일 뿐이다. 몽키 렌치, 바이스 그립, 채널 렌치, 펜치가 도구상자에 있다고 가정해 보자. 프로젝트를 완수하려면 어떻게 해야 할지는 생각하지도 않은 채 그저 몽키 렌치로만 모든 것을 하려고 한다면, 결국에는 나사만 망가지고 말 것이다.

많은 게임 개발 프로젝트에서, 힘들게 교훈을 배우는 경우가 있다. 힘들게 얻는 교훈 한가지는 스크럼이 비디오 게임 제품 개발을 보다 성공적이 되게 하는 묘책은 아니라는 점이다. 다년간의 게임 개발 경험, 일련의 개발 과정들, 어떻게 하면 잘되고 잘못 되는지에 대한 경험들을 통해 얻어진 많은 기술적인 세부사항들과 노하우와 최선책들이 있기 마련이다.

스크럼이 있기 전에 존재했던 이미 잘 확립되어 있는 과정과 방법들을 스크럼으로 완전히 대체하는 것은 확실히 프로젝트에 비참한 결과를 초래하는 지름길이다. 스크럼이 문제를 해결하고 성공을 보장하기 위해 프로젝트에 추가될 때에, 함정들에 대해 적절한 시점에 대책을 마련하지 않을 경우, 문제를 해결하기 보다는 양산하게 되기가 쉽다.

본 기사는 스크럼에 대한 몇 가지 대조적인 통찰력을 제공하였다. 분명 스크럼을 의무화하려는 경영진의 결정이 있기 전에, 양쪽 진영, 즉 스크럼을 절대적으로 옹호하는 쪽과 주의 깊게 접근하면서 덮어놓고 반대만 하는 사람 모두의 의견에 귀를 기울여야 할 것이다.