게임 툴의 튜닝 업 - 유용성 향상으로 파이프라인을 최적화 함
(Game Tools Tune-Up: Optimize Your Pipeline Through Usability)

Dan Goodman

가마수트라 등록일(2009. 5. 6)

(http://www.gamasutra.com/view/feature/4016/game_tools_tuneup_optimize_your_.php)

[맞춤형 게임 툴의 완성도를 어떻게 향상시킬 수 있는가? Sega와 LucasArts의 이전 기술자이자 Robotic Arm Software 의 설립자인 Goodman은 자체 툴을 사용할 때의 방법론에 대한 팁과 용법에 대해 기술한다.]

게임 개발에서, 가장 전문적인 사람(프로그래머)이 가장 비 전문적인 사람(그 외 사람)을 위해 툴을 만드는 경우가 종종 있다. 당연히 이 "가장 비 전문적인 사람" 다수는 그들에게 전수된 툴을 사용하고자 애쓰기 마련이다.

이 툴 자체는 내재한 시스템을 이해하는 사람들이 고안한 것이지만, 그것을 사용할 사람들은 반드시 이해해야 할 필요는 없다.

유용성은 소프트웨어를 얼마나 쉽게 이용해 볼 수 있는지, 원하는 결과를 사용자가 얼마나 빨리 달성할 수 있는지, 그 결과는 에러 율을 어느 정도까지 낮출 수 있는지를 결정한다. 유용성을 테스트하고 개선하는 기술들은 다른 형태의 소프트웨어 개발 분야에서 성공적으로 사용되어 왔으며, 우리의 분야에서도 점진적으로 진보해 왔다.

유용성 관련 기술로부터 유익을 얻을 만한 툴이 독자의 파이프라인에도 있을 것이다. 그러나 개발자로서, 투자에 앞서 먼저 어떤 툴이 가장 이익이 될 지를 알기를 원할 것이다.

이번 기사에서는, 생산을 지연시키는 장애 요소를 어떻게 찾아낼 수 있는지, 입증된 기술을 사용하여 툴의 유용성을 어떻게 평가할 것인지, 전체 개발 과정을 합리화하는 방법은 무엇인지에 대해 논할 것이다.

파이프라인 맵핑

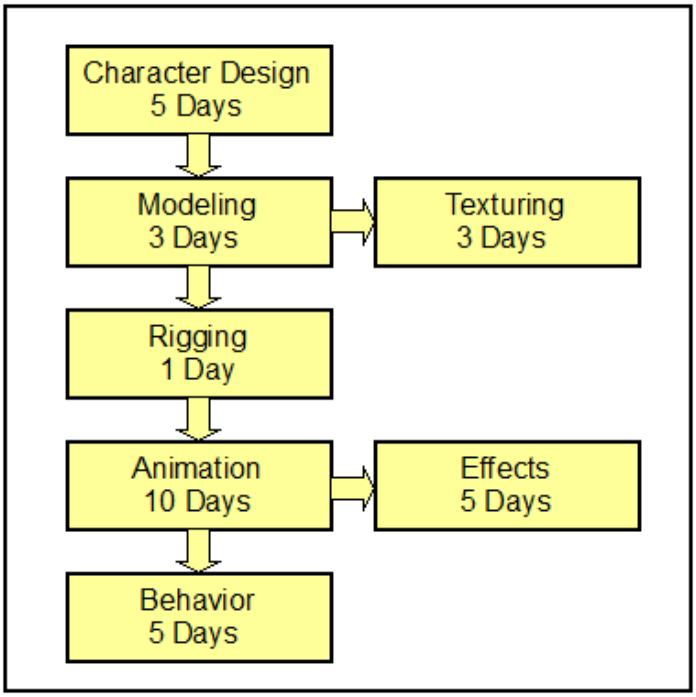
어떤 문제든 해결을 위해 가장 먼저 취할 단계는 문제를 이해하고 발생 원인을 파악하는 것이다. 개발 파이프라인의 어느 지점에서 과정의 진척이 멈추었는지를 알아야 한다. 이를 위해서는 문제 발생 공간을 맵핑 해 볼 필요가 있다.

개발 파이프라인을 바라보는 관점은 여러 가지가 있다. 자산이나 책임의 점검 목록으로 여기는 사람도 있겠지만, 이것은 개발자들 간의 의존성에 대해서는 설명해 주지 못한다.

그러한 의존성은 다른 누군가의 직무 완수를 위해 기다리는 동안 낭비되는 시간을 야기하게 마련이며, 그런 경우, 개발자들의 관리 노력이 요구된다.

이러한 의존성을 이해하기 위해, 파이프라인을 다양한 자원과 각각이 산출하는 것 간의 작업 흐름도로 바라보는 관점이 보다 편리할 것이다. 아래 표에서 캐릭터 생성을 위한 작업 흐름도의 예를 고려해 보자.

[도표]캐릭터 디자인 5 일 - 모델링 3 일 - 텍스처링 3 일 - 리깅 1 일 - 애니메이션 10 일 - 이펙트 5 일 - 행동 5 일



이 도표에서, 화살표는 종속된 직무, 곧 화살표가 기인한 요소에 의존한 직무를 가리킨다. 그러므로 모델링은 캐릭터 디자인에 의존하며, 텍스처링과 리깅은 모델링에 의존한 것이다.

이것은 전형적인 게임 개발 파이프라인의 아주 작고 불완전한 단면을 보여주는 것이다. 실제 파이프라인은 훨씬 더 복잡할 것이며, 개발(레벨 설계, UI, 게임플레이 프로그래밍 등)의 더 많은 영역, 다중 과정을 수행하는 자원, 단일 직무에 대한 다중 의존성 등을 포함할 것이다. 개발 파이프라인을 지연시키는 단계들은 다른 수단들에 더해 툴 개발을 통한 최적화를 위해 살펴볼 필요가 있다.

캐릭터 개발의 각 단계가 다음 단계를 진행하기 전 완수되어야 한다고 가정했을 때, 첫 번째 캐릭터는 24 일 이내에 완수될 수 있다. 캐릭터 디자인부터 완수까지 최장 경로(주요 개발 파이프라인)에서의 일수를 더함으로써 이 값을 얻는다.

텍스처링과 이펙트는 계산에서 제외되는데, 그 경로들이 메인에서 벗어나기 때문이다. 이 경로들을 완수하는 데 걸리는 시간은 주요 경로보다 짧거나 그와 동일하다.

캐릭터 디자인 + 모델링 + 리깅 + 애니메이션 + 행동 = 24 일

하지만 매번 캐릭터를 완수하는 데 달력일수로 24 일이 걸리는 것은 아니다. 이것은 일괄작업조직으로서, 네 개의 다른 캐릭터를 가지고 작업하는 동안, 현재 한 개 캐릭터는 완료되고 있는 경우, 곧 파이프라인의 각 단계에서 하나씩 수행된다. 때문에 첫 캐릭터 이후로 새로운 캐릭터를 완수하기까지 달력일수로 몇 일이 걸릴 것인가? 정확하게 최장 단계에서 소요되는 그 시간, 곧 10 일 될 것이다.

달력일수로 6 개월 안에 10 개의 캐릭터를 완수했다면 매우 잘 한 것이다. 물론 애니메이션에 들이는 시간이 절반으로 줄어들었을 경우, 새로운 캐릭터를 5 일마다 제작함으로써 캐릭터 산출량을 두 배로 만들 수 있다.

어떤 일괄작업조직에서든, 효율성은 직무를 구성 성분으로 분해함으로써 향상시킬 수 있는데, 이로써 제조의 각 단계가 동일한 시간을 소요하게 하기 때문이다. 이는 또한 게임 자산을 개발하는 데 있어서도 가장 효율적인 모델이라 할 수 있다.

안타깝게도, 파이프라인의 각 단계는 고유의 기술을 요하며, 종종 각 단계에 소요되는 정확한 시간을 예측하기가 어렵다. 따라서 어느 개발자에게는 일이 밀려 있는 반면, 다른 개발자는 처음 작업이 끝나길 기다리며 한가롭게 쉬고 있는 일이 생길 수 있다.

이 문제와 관련하여 다양한 방책들이 있는데, 모두 여러 배경에서 게임회사들에 의해 채택되고 있다. 관리 비용 및 개별 효율성은 종종 개발 파이프라인 범위에 따라 상관적으로 책정된다.

그러나 효율성이 진행 지연을 야기하는 파이프라인 영역에서 증가될 수 있다면, 파이프라인 범위 역시 증가할 것이며, 비용은 거의 소요되지 않을 것이다 이러한 효율성을 달성하는 한 가지 방법은 툴을 개선하는 것이다.

유용성 테스트

일단 파이프라인에서 장애가 발생하는 요인을 파악했다면, 이제는 그 과정에서 사용되는 툴을 개선함으로써 도움이 될 만한 곳이 어디인지, 어떤 개선이 필요한지를 확인해야 한다.

유용성을 테스트하는 방법은 다양한 데, 이를테면 설문조사, 사용자 인터뷰, 검사 기술, 생각 말하기(thinking aloud protocol) 및 시선 추적 등이 이에 해당한다.

어떤 기술은 시행하기가 난해하거나 전문가가 해석해야 할 만큼 주관적인 결과를 산출하는 반면, 어떤 기술은 누구든 제품의 유용성을 측정하는 데 사용할 수 있을 만큼 보다 공식적인 점수 환산이 가능한 것도 있다.

보다 난해한 것에 투자하기 전에 먼저 비교적 단순한 기술을 사용해 볼 수 있다. 이로써 노력을 집중할 만한 실제로 좋은 아이디어를 얻게 될 것이다.

시스템 유용성 척도(SUS)는 유용성 측정을 위해 디지털 이퀍먼트 코퍼레이션(DEC)에서 개발한 것이다. 설문조사를 기반으로 한 것이므로, 수행하기 가장 쉬운 방법 중 하나일 것이다.

참가자들은 아래 그림에서 열 가지 문장에 “강하게 부정함”에서 “전적으로 동의함”까지 등급을 매기도록 요청 받는다. 각각의 답변은 0 점부터 4 점이 매겨지며, 이때 짝수 항목은 점수 수치를 뒤바꾼다.

0-100 까지의 전체 점수는 개개의 수치를 합산하여 2.5 로 곱하여 도출된 것이다.

The 10 statements adapted from the original SUS survey.

1. I think that I would like to use this tool frequently
2. I found the tool unnecessarily complex
3. I thought the tool was easy to use
4. I think that I would need the support of a technical person to be able to use this tool
5. I found the various functions in this tool were well integrated
6. I thought there was too much inconsistency in this tool
7. I would imagine that most people would learn to use this tool very quickly
8. I found the tool very cumbersome to use
9. I felt very confident using the tool
10. I needed to learn a lot of things before I could get going with this tool

[박스]원본 SUS 설문조사에 사용된 10 개 문장

1. 나는 이 툴을 자주 사용하게 될 것으로 생각한다.

2. 이 툴이 불필요하게 복잡하다는 것을 알게 되었다.

3. 이 툴은 사용하기 쉽다고 생각한다.

4. 이 툴을 사용할 수 있으려면 기술자의 지원이 필요할 것 같다.

5. 이 툴의 다양한 기능은 잘 통합되어 있는 것 같다.

6. 이 툴은 너무 일관성이 없다고 생각한다.

7. 대부분의 사람이 이 툴을 매우 빨리 배울 수 있으리라 생각된다.

8. 이 툴은 사용하기에 매우 부담이 된다고 느낀다.

9. 이 툴을 사용하는 데 매우 자신감이 있다.

10. 이 툴에 익숙해지기까지 많은 것을 배워야 했다.

70-80 범위의 소프트웨어는 그 유용성이 평균에 해당한다. 자신의 소프트웨어가 이보다 높다면, 그 외의 영역에서 절약할 곳을 찾아보는 것이 좋을 것이다. 점수가 그보다 낮다면, 유용성에 투자함으로써 개발 시간을 줄여볼 좋은 기회가 될 것이다.

단일 유용성 측정표(SUM)는 SUS 보다는 수행하기가 다소 복잡한데, 이는 소프트웨어를 사용하여 사용자가 서너 개의 공통 테스크를 완수하는 동안, 조정자가 각 테스크가 완수된 방식에 점수를 매겨야 하기 때문이다. 최종 점수는 완수된 테스크 등급, 테스크 완수 시간, 에러 개수 및 사용자 만족도를 합산한 것이다.

테스크의 완수는 시도한 테스크를 성공적으로 완수한 비율이다. 에러 등급은 개별 테스크를 하위 테스크로 분할하여 측정하는 데, 이를 “에러 기회”라고 부른다. 테스크 당 성공적으로 수행한 단계들의 수를 계수한다.

만족도 등급은 각각의 테스크가 완수된 후에 주어지는 단순한 세 가지 질문의 조사 결과로 측정하는 데, 이 질문들은 사용자에게 테스크가 얼마나 쉬웠는지, 시간은 얼마나 소요되었는지, 툴을 사용하면서 만족도는 어떠했는지를 묻는 것이다.

테스크 소요 시간은 이상적인 시간과 각각의 테스크 완수에 걸린 시간을 비교해 보는 것으로서, 최고 만족도를 보고한 사용자의 테스크 소요 시간을 사용하여 이상적인 시간을 도출할 수 있다.

SUM 이 테스크 수준에서 유용성을 측정하는 것이므로, 이 방법을 개선하기 위해 어떤 기능들을 검토해 보아야 할지 결정하는 데 사용해 볼 수도 있다. 이 정보를 생각 말하기, 즉 누군가 툴을 사용하며 앉아있는 동안, 그들이 생각하는 것을 기술하게 하는 방법과 결합함으로써 문제 해결 방법에 대한 추가 세부점들을 얻어낼 수 있다.

물론 전반적인 유용성이 그 목적이어야 하며, 사용자 목적에 기초한 인터페이스 재디자인은 종종 특정 기능을 목표로 하는 것 보다 더 나은 결과를 산출하게 해 줄 것이다.

툴 범위 및 데이터 추출

가장 손쉬운 툴은 사용자로 하여금 데이터를 편집할 수 있게 해준다. 자신이 가장 좋아하는 문서 편집기가 적어도 이 정도의 기능성은 제공해 줄 것이다. 반드시 그런 것은 아니지만, 문서 형태로 제시할 수 있는 어떤 종류의 데이터이든 편집이 가능하므로 가장 폭넓은 범위를 제시하기도 한다.

이러한 툴은 사용자 보호 기능이 없으므로, 실수를 용납하지 않는다.

사용자들에게는 반복 사용이 중요한 데, 실수를 하기가 쉽기 때문이다. 실행 중 에러가 발생하면 오류 데이터를 찾아내기가 극히 어려울 수 있다. 사용자는 잘못된 소수 자리가 있는 수나 부정확한 대문자가 사용된 문서가 있는지를 확인하는 데 여러 시간을 보내야 할 수도 있다.

여러 게임 개발 툴의 핵심은 유효값에 사용자 에러가 입력되는 일이 없도록 제한함으로써 보호층을 추가하는 것이다. 이것은 중요한 진보를 이룬 것으로서, 개발자들이 에러가 덜 발생하는 게임 자산을 생성할 수 있게 해주기 때문이다. 디자이너와 작가는 “적어도 실행은 될 것이다”라는 안도감을 느끼게 될 것이다.

그러나 데이터가 게임에 어떤 영향을 주는 것인지에 대한 피드백이 거의 없으며, 따라서 변경사항을 테스트하고, 수치를 조정하는 등에 수반되는 반복 과정에 많은 시간이 소요된다. 이는 그러한 툴이 아직 설계의 수행 모델에 기초해 있기 때문이다.

다시 말해, 사용자가 직접 편집하는 데이터는 내재한 게임 시스템이 사용하는 데이터와 일치하며, 따라서 사용자가 온전히 이해할 수 있는 것보다 더 복잡할 수 밖에 없다.

툴이 달리 심각한 오류(충돌)를 야기할 만한 입력상의 오류로부터 사용자를 보호해 줄 필요가 있으나, 반복과정이 여전히 필요하긴 하다. 입력 오류가 제거되었더라도, 논리상의 오류는 여전히 만연해 있다.

비록 규모가 작기는 하나, 데이터 추출이 이러한 도구에서 나타나기 시작했다. 문서상자가 열거된 값의 드롭다운 식 목록으로 대체되고 있는 가운데, 데이터(수)의 내부 표시 형태가 사용자가 이해하기 더 쉬운 방식으로 대체되고 있다.

컬러 픽커, 그래프, 캘린더, 진행상황 표시 바 및 데이터의 그래픽 표시 모두가 이런 목적에 부합한다. 그러나 데이터 비트 각각이 통일적으로 전체를 표시해 주지는 못한다.

이를 달성하기 위해서, 일종의 추출 방식을 통해 개개의 데이터 포인트를 더 크고 개념적인 표상에 병합시켜야 한다. 실제 데이터는 데이터가 묘사하는 객체를 표현하는 외관 뒤로 감추는 것이다.

3D 모델을 고려해 보자. 실제 데이터는 정점 위치, 중량, UV 맵핑 좌표, RGBA 수치 등의 목록이다. 이 수치 각각을 직접 편집하려면 극도의 정확성이 요구되나, 훨씬 더 나은 결과를 산출하는 가운데 동일한 데이터 종류를 발생시키는 모델링

프로그램을 사용하는 것 보다 훨씬 더 많은 시간이 소요될 것이다.

이 모델링 프로그램은 데이터 자체가 아닌, 데이터가 표시하는 것을 보여줌으로써 데이터의 복잡성을 감추고, 사용자에게는 훨씬 더 이해하기 쉬운 방식으로 모델 데이터를 편집할 수 있게 해준다.

툴이 실제 데이터 값을 적절히 추출할 때에, 반복 횟수가 현저하게 줄어드는데 이는 입력오류와 논리적 오류가 모두 제거되기 때문이다. 데이터의 추상적 외관은 사용자로 하여금 그 목적을 이해하기 더 쉽게 해 줄 것이다. 최종 데이터는 게임에서 테스트해 보아야 하지만, 기본적으로 툴에서 생성된 데이터도 분명 온전해야 한다.

물론 데이터가 최상 표시 상태인지를 알아내기란 쉬운 일이 아니다. 이것은 툴의 사용자를 이해함으로써 가능해 진다. 그 표시 내용이 사용자가 이해하는 방식과 가까울수록, 툴을 사용하기가 더 쉬워진다.

목표 지향적인 설계 및 사용자 모델

때때로 우리 자신이 바라는 대로 사용자에 대해 이야기 하곤 한다. “나는 그 플레이어가 실제로 이것을 좋아할 것 같다”고 말하는 것은 종종 게임의 특정 기능을 포함시켜야 한다는 변명처럼 들리기도 한다.

그 어두운 면을 이야기하기는 쉬우나, 두 개발자가 이점에 동의하지 않을 경우 쉽게 결론이 나지는 않을 것이다.

이는 사용자가 보통 매우 잘 정의되어 있지 못하기 때문이다. 모두가 그 목표 대상이 누구인지 자신의 생각을 가지고 있으며, 툴에 대해서도 그렇다. 바로 이 때문에 사용자 모델이 필요한 것이다.

사용자 모델은 자신의 목표, 바램, 취미, 가족, 및 심지어 이름까지 가지고 있는 구체적이며 허구적인 최종 사용자이다. 이들은 반드시 “평균적인 사용자”라고 할 수는 없으나, 전형적인 사용자를 대표한다.

하나의 소프트웨어는 소프트웨어를 사용해야 하는 몇몇 사용자 모델을 가지고 있을 수 있으나, 핵심 관건은 인터페이스를 설계하여 사용자들 중 하나를 매우 잘 만족시켜 주어야 한다는 것이다. 이 사용자는 주요 사용자 모델이 되며, 전체 팀은 상기 언급한 것과 같은 논쟁을 피하기 위해 주요 사용자가 누구인지에 대해 함께 논의해야 한다.

필자가 일한 적이 있는 한 회사에서, 설계팀장이 레벨 디자인 툴에 대한 몇 가지 아이디어를 구상해 냈다. 그는 자신이 팀의 목표라고 느낀 것에 기초하여 이 일을 수행하였다.

필자를 포함하여 이 툴 개발 팀은 그의 디자인 아이디어를 거절하였는데, 우리가 분명 더 잘 알 것이라는 판단과 툴에서 편집할, 내재된 데이터를 우리가 더 잘 이해하고 있다는 판단 때문이었다. 필자는 우리가 사용자 모델 개념을 이해하고 있으며, 당시에 디자인이 어때야 할 지에 대해 논의할 때 쯤에는 이를 사용하게 되기를 바랬다.

이 툴의 성공 여부를 판가름 할, 그 사용자에 대한 상당한 지식을 가지고 있는 사람이 있었지만, 효과적으로 의사를 전달할 만한 툴을 가지고 있지 않았다. 툴 담당 팀은 최종 사용자의 대변인으로 그와 긴밀히 협력하며 작업해야 했는데, 이로써 주요 사용자 모델을 생성하고 디자인 팀의 현실적인 목표를 결정하기 위해서였다.

결국 레벨 편집자는 필요한 역할을 하지 못했고, 레벨 디자이너는 그 문제를 해결하느라 많은 시간을 소비해야 했다.

그러면 사용자 모델을 어떻게 고안할 것인가? 가장 좋은 방법은 실제 사용자들을 연구해 보고 개별적인 특성을 종합하는 공통 속성을 찾아내는 것이다. 그러나 실제- 사용자들의 그처럼 다채로운 속성들 중에서 독자의 사용자 모델은 어디에 해당할 것인가?

자신의 사용자가 시간이 지남에 따라 해당 소프트웨어에 익숙해 질 것인지를 고려해야 하지만, 대다수는 결코 “전문가” 수준까지는 도달하지 못할 것이다. 독자의 툴은 중간 수준의 사용자의 필요를 가장 잘 조명해야 하며, 그렇게 사용자 모델은 그 수준의 실력을 갖춘 누군가를 대표하는 것이어야 한다.

자신의 사용자 모델을 위한 이름을 생각해 보라. 이름을 사용하면 자신이 속한 집단과 해당 사용자 모델에 대해 더욱 효과적으로 논의할 수 있게 될 것이다.

한 예로, 새로운 레벨 디자인 툴을 제작하고 있다고 가정해 보자. 디자이너를 비롯하여 대부분의 게임 개발자들은 남자이며, 따라서 우리 사용자 모델도 남자여야 한다. 필자에게는 레벨 디자이너로서 “브래드”가 좋은 이름처럼 들린다.

브래드는 몇 살인가? 게임 개발에서, 디자인은 기술이나 프로그래밍에 비해 매우 젊은 영역이므로, 브래드는 아주 젊을 것이다. 팀의 평균 레벨 디자이너 나이가 27 세라고 가정해 보자.

브래드는 어느 학교를 다녔는가? 무엇을 공부했는가? 아마도 문과생이었을 것이다. 컴퓨터공학에 학위가 있을 수도 있으나, 그럴 경우 브래드는 보다 기술적인 디자이너가 되었을 것이다. 우리는 우리의 툴이 기술적인 경험이 덜한 사람을 목표로 하기 원하므로 대신 영문학에 학위가 있다고 가정해 보자.

브래드는 결혼한 사람인가? 아마 그렇지는 않겠지만 아마도 여자친구는 있을 것이다. 그럼 이 모든 것이 어떻게 툴을 개발해야 할 것인지에 어떤 영향을 미치는가? 툴에 직접적인 영향을 미치지는 않지만, 분명 브래드의 개인 목표에는 영향을 미칠 것이며, 다시 말해, 직업상 그의 목표에 영향을 미칠 것이다. 이 사용자 모델에 좀 더 살을 붙여 보기 위해 큰 노력이 들지는 않을 것이다. 더욱이 그러한 노력은 실재 사용자에 대한 더 깊은 이해를 얻게 해줄 것이므로 그럴만한 가치가 있다.

브래드의 목표는 무엇인가? 그는 왜 게임 개발에 몰두했는가? 현재 직위에서 달성하기 원하는 것은 무엇인가? 일단 사용자 모델의 배경을 알게 되면, 그의 동기를 이해할 수 있을 것이다.

이를 이해한다면 그의 목표를 달성하게 할 툴을 개발하는 방법을 이해하는 데에도 도움이 될 것이다. 그 목표들 가운데는 정말 도전적인 게임을 만드는 것, 초과 근무를 너무 많이 하지 않는 것, 봉급 인상, 승진 등 몇 가지가 포함될 수 있다.

이는 궁극적으로 툴 별로 영향을 미치는 고차원적인 목표들이다. 특정 차원에서 복잡한 문제들을 연이어 처리하기 원하는 경우, 이는 사용자가 도전적인 게임을 만들기 원한다는 것을 보여주는 것일 수 있다.

많은 초과근무를 원하지 않는 사용자는 사용하기 편하고 신속한 툴을 원하는 것일 수 있다. 봉급인상이나 승진을 원하는 사람이라면 자신의 업무에 대해 드러내고 두드러져 보이게 할 기회를 찾는 툴을 원하는 것일 수 있다. 궁극적으로는 사용자가 원하는 바를 이해하는 것이 그들의 필요를 충족시키고, 열심히 일하는 직원들을 행복하게 해줄 기회를 얻게 해 줄 것이다.

툴 개발 기간 동안, 개개의 사용자들로부터의 피드백이 매우 가치 있을 수도 있으나, 피드백에 대한 결론을 소프트웨어에 직접 투입하기 보다는 사용자 모델에 대한 사용자의 의견을 확인해 보는 것이 좋을 것이다.

각각의 사용자는 자신의 고유한 작업 방식이 있을 것이지만, 독자가 찾고 있는 것은 누구나 잘 사용할 수 있는 방법이다. 개개의 요청들을 다 수렴하려고 하다 보면 전체 그림에는 적합하지 않은 인터페이스 요소들로 뒤범벅이 되고 말 것이다.

사용자 모델에 기초하여 특정 사용자를 겨냥하고 그들의 목표에 초점을 맞추라. 이 사용자들은 그들이 원하는 것을 정확하게 제공받아야 한다. 그보다 덜할 경우 작업을 수행하지 못할 것이고, 그보다 지나칠 경우에는 일이 지나치게 복잡해 질 것이다. 모든 디자인 관련 결정은 목표 필터를 거쳐야 한다. 곧, "이것이 사용자의 목표에 부합하는가?" 하는 것이다. 이 요구조건에 부합하는 기능들만 디자인에 포함되어야 한다.

기능이 목표에 부합하지 않을 경우, 최종 사용자를 위해 더 적은 테스크를 선별하는 것이 목표 달성에 더 좋은 방법일 것이다. 사용자 모델의 목표를 이해함으로써 개인에게 가장 적합한 방식으로 데이터를 표시할 수 있는 방법을 찾을 수도 있을 것이다.

자신의 툴로 사용자가 더 나은 성과를 얻을 수 있도록 이러한 목표들에 어떻게 대처할 것인지는 각 툴에 따라 달라진다. 일단 제대로 수행하기만 하면, 사용자들은 단기간 내에 훨씬 더 좋은 결과를 달성해 내게 될 것이다.

결론

자신의 개발 파이프라인에서 장애 요소를 찾아 여기서 필자가 논의한 기술로 조명해 볼 수 있을 것이다. 찾아보면 다른 방법들도 많이 있을 것이며, 자신의 팀과 잘 부합하는 것을 찾기만 하면 된다.

자신의 툴이 필요만큼 편리하지 못하다고 판단된다면, 목표지향적인 수단을 사용하여 재설계를 해야 할지 고려해 보아야 한다.

사용자를 이해함으로써 그들의 필요에 부합하는 소프트웨어를 개발할 수 있을 것이다. 사용자들이 일단 최고의 툴을 손에 쥐게 되면, 더 높은 효율성과 생산성으로 업무를 진행할 것이며, 결국 더 나은 게임 개발로 이어질 것이다.

추가 정보 참고 문헌

A. Cooper. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley.

J. Brooke. (1996) *SUS: A quick and dirty usability scale*, pp.189--194. *Usability Evaluation in Industry*. Taylor and Francis.

J. Sauro & E. Kindlund (2005) *A Method to Standardize Usability Metrics Into a Single Score*. 검색일자 2008 년 12 월 17 일, 유용성 측정 웹 사이트: http://www.measuringusability.com/papers/p482-sauro.pdf

---