Identifying and Managing Technical Project Risks
기술 프로젝트 리스크의 식별 및 관리

Jamie Blum - August 19, 2019

Topics: Risk Management, Knowledge Shelf

요약
프로젝트 리스크 완화를 위한 계획수립은 프로젝트 관리 측면에서 중요하지만, 복잡한 소프트웨어 프로젝트에서는 특히 중요하다. 이러한 형태의 프로젝트 리스크는 두 개의 범주로 - 프로젝트 리스크와 기술 리스크- 나눌 수 있다. 프로젝트 리스크는 소프트웨어가 최종 사용자를 위해 잘 고안되고 작동되는 것을 보장하기 위한 기술 리스크로부터 구별되어야 한다. En Route Automation Modernization (ERAM) 소프트웨어 개발 프로젝트에서 한 가지 잠재적으로 회피가능한 기술 리스크가 논의되었고, 기술 리스크의 가능한 원인들이 프로젝트 초기에 그런 리스크를 낮출 수도 있었을 대응 계획과 함께 강조되었다.

서론
프로젝트 리스크는 피할 수 없고 가능한 최대 범위까지 관리되어야 한다. 어떠한 형태의 프로젝트들에 대해 리스크는 두 개의 범주로 나눌 수 있다 – 프로젝트 리스크와 기술 리스크. 프로젝트 리스크는 한 프로젝트의 생애주기 동안 발생하게 된다. 기술 리스크는 프로젝트 종료 후 부적합한 설계 때문에 발생할 수 있는 어떠한문제점을 말한다(Fontaine, 2016).

이 논문에서는, 특히 Fontaine (2016)가 "허용불가 기술 리스크"라고 말하는, 기술 리스크를 주로 다루고 있다. 다음은 En Route Automation Modernization (ERAM) 소프트웨어 개발 프로젝트 논의된 잠재적으로 회피할 수 있는 허용불가 기술 리스크의 한 사례이다. 기술 리스크의 가능성 있는 원인들과 허용불가 기술 리스크를 허용 가능한 수준으로 낮출 수 있었을 대응계획을 중점적으로 다루고자 한다.

En Route Automation Modernization (ERAM) 소프트웨어 고장
2014 년 4 월, 많은 항공편이 Lockheed Martin 이 설계한 En Route Automation Modernization (ERAM) 소프트웨어에서의 한번의 고장으로 지연되거나 취소되었다. 그 소프트웨어는 ERAM 시스템으로 한꺼번에 복수의 비행기들로부터 너무 많은 데이터가 전송되므로 인한 메모리 과부하 때문에 고장 났다. ERAM 소프트웨어는 각 비행기로부터 받을 수 있는 데이터의 한계를 갖고 있다. 대부분의 상황에서, 각 비행기에 의해 전송되는 데이터는 아주 작다.

하지만, 이 특별한 날엔 보다 복잡한 비행 계획이 있는 한 대의 U-2 비행기가 ERAM 이 한계를 뛰어 넘도록 요구했다. 주요 원인 중 하나는 U-2 비행기의 비행 고도가 ERAM 에서 사용할 수 없었기

때문에, 시스템은 모든 가능한 옵션을 입력하기 시작하였고, 이는 여러 번의 재시작 작업을 반복하고 오류 메시지를 표시하는 것을 초래하게 되었다.

메모리 과부하로 인해, 그 시스템은 다른 처리 기능들을 수행할 수 없게 되었고 그 결과, 미국 남서부에서 수백 건의 비행편이 취소되거나 지연되었다 (Scott & Menn, 2014). 이것이 리스크 관리 계획 수립 어디에서도 고려되지 않은 허용불가 기술 리스크의 사례이다.

이듬해인, 2015 년 8 월, ERAM 소프트웨어는 업그레이드 되었으나 비슷한 운명에 직면했다. 이 소프트웨어를 업그레이드 했다는 것은 관제사들에게 쉽게 접근 할 수 있는 사용자 정의 인터페이스를 제공하여, 그들에게 자주 참조해야 하는 데이터를 볼 수 있도록 하기 위함을 의미하는 것이다. 즉, 관제사가 시스템에서 삭제하고자 의도했다면 지워진 후, 조정된 값으로 세팅이 되어야 한다. 정보가 의도 한 대로 시스템에서 제거되는 대신, 시스템은 관제사들이 조정 한 경우 정보를 유지했다.

또 다시, 데이터로 인해 저장공간이 설정된 한계를 초과할 때 메모리 과부하는 발생하였다. 미연방항공청 (FAA)은 Lockheed Martin 과 협력하여 테스트 중에 버그가 포착되지 않은 이유를 파악했지만, 피해는 이미 발생해 버렸다. 미국 동해안의 1,000 편이 넘는 항공편이 취소 또는 지연되었다 (Halsey, 2015).

ERAM 과 기술 리스크

ERAM 소프트웨어 문제는 프로젝트 완료 및 새 버전으로 업그레이드 발표된 이후에 발생하였고, 리스크 관리 계획수립 기간 동안 기술 리스크로 식별될 수 있었다. 만일 지금의 이 두 가지 개별 고장에 의한 결과에 비참한 대중적 의견, 매출 손실 그리고 고객 불만이 포함되었다는 것을 알았다면, 이러한 문제는 거의 분명히 리스크 계획수립 중에 허용불가 기술 리스크로 평가되었을 것이다.

사실, Lockheed Martin 과 FAA 가 소프트웨어 업그레이드 테스트를 수행할 때 데이터 문제가 시스템 메모리에 영향을 줄 수 있다는 것을 인식하지 못하였다는 것과, 특히 그것이 원래 소프트웨어 고장 때문이란 것을 인식하지 못한 것에 대해 놀랐다.

ERAM 소프트웨어의 제작은 약 24 억 달러의 비용이 소요되는 복잡하고 값비싼 프로젝트였다. 이러한 규모의 프로젝트에서, 프로젝트와 기술 리스크를 해결하기 위한 리스크 관리 계획을 수립하는 것은 필수적이다. 리스크는 어떤 현재의 결정 또는 행동으로부터 초래되는 가능한 미래의 실패 또는 원치 않는 결과로 분류될 수 있다 (Samantra, Datta, Mahapatra, & Debata, 2016).

소프트웨어 프로젝트의 리스크 인자는 소프트웨어 프로젝트의 성공적인 완료 또는 수행을 위협하는 어떠한 리스크를 포함하며, 이러한 리스크는, 만일 식별이나 이해되지 않았다면, 빠르게 프로젝트 실패로 이어질 수 있다. 리스크를 완화하기 위해서는, 신중하게 계획되고 철저한 프로세스가 적소에 마련되어 있어야 한다. 이러한 프로세스는 리스크를 식별 및 평가하고 그리고 발생할 수 있는 리스크를 해결하기 위한 세밀하고 우선 순위가 지정된 대응 계획을 작성하는 것으로 시작한다.

불행하게도, 소프트웨어 프로젝트에서 리스크 관리를 취급하는 체계적인 프로세스를 검토하기란 제한적이지만, 소프트웨어 프로젝트의 복잡성을 감안할 때, 이는 개선 가능성이 큰 영역이다 (Samantra et al., 2016).

소프트웨어 프로젝트에서 리스크 관리

첫 번째 ERAM 소프트웨어 고장 이후, FAA 는 그 시스템이 모든 비행 계획에 고도를 포함하도록 요구사항을 조정하였고, 그리고 시스템에 좀 더 많은 메모리 또한 추가하였다 (Scott & Menn, 2014). 두 번째 고장 이후, 그들은 사용자 인터페이스 기능을 중단하고 Lockheed Martin 이 테스트 기간 동안에 왜 오류를 발견하지 못했는지 이유를 조사하였다 (Halsey, 2015)

우리는 Lockheed Martin 과 FAA 가 작성했을 리스크 관리 계획을 들여다 보지는 않았지만, 24 억 달러나 하는 프로젝트에 리스크 관리 계획이 적소에 마련되어 있었을 것이라는 논리적인 가정은 할 수 있다. 프로젝트 복잡성 리스크는 신규 또는 전문적으로 복잡한 기술을 포함하는 경우 회피할 수 없다. 이러한 상황에서 기술 리스크를 해결하기 위해 철저한 연구, 엄격한 QA 프로세스, 초기의 제한적인 작업, 단계별 실행 그리고 우발사태 계획 수립 등 모두 포함하여 대응계획을 수립해야 한다. (Hu et al., 2013).

Vrhovec, Hovelja, Vavpotič 과 Krisper (2015)는 소프트웨어 프로젝트 리스크 관리를 위해 가장 일반적으로 사용되는 3 가지 방법을 설명한다:

1. 첫 번째 방법은 점검 목록이며, 기술, 조직 그리고 프로젝트 리스크를 포함할 수 있다. 체크리스트는 선행 프로젝트 리스크를 기초로 할 수 있다. 이러한 리스크 형태들은 정형적으로 분리되지 않지만, 대신 가능성에 의해 우선순위화 된다.
2. 두 번째 방법은 분류이며, 일반적으로 점검 목록을 어떤 프레임워크에 따라 분류하는데 사용한다. 점검 목록은 일반적이고 상세하기 때문에, 분류는 범주별로 리스크의 우선 순위를 정하는 데 도움이 된다.
3. 세 번째 방법은 프로세스 모델링으로, 내용 제공, 리스크를 식별, 분석 및 평가하는 일련의 공식 프로세스를 통해 리스크 관리 조치를 구체화하는 것과 완화, 의사소통 및 자문을 실시하는 것이다.

비록 기술 리스크가 간과 되지 않도록 기술 리스크로부터 프로젝트 리스크를 분리하는 것은 중요하지만, 3 가지 방법 모두를 조합하는 것이 ERAM 소프트웨어 프로젝트에 유익하였을 것이다. ERAM 소프트웨어의 기술 리스크는 프로젝트가 착수되고 설계될 때 고려되었어야 하고, 프로젝트 전 생애주기를 통해서 문제점 점검을 위해, 특히 소프트웨어 설계가 구체화되는 시점에, 모니터링 되었어야 한다 (Fontaine, 2016).

또한 Lockheed Martin 이 PMI (2013)이 제시한 소프트웨어 프로젝트에 관한 기술 리스크 목록을 고려하였다면 유익했을 것인데, 거기에는 결함, 용량 규모 문제, 성능 요구사항, 소프트웨어 사용 용이성, 시나리오 변경 등을 포함한다. 프로젝트 관리자는 전문가 판단을 이용해야 하고, 가능한

비즈니스 분석가들의 통찰력을 사용하여 잠재적 리스크를 브레인스토밍하고 각 리스크의 발생 가능성을 결정하여 리스크들을 우선 순위화하고 완화 계획을 실행해야 한다. 숙련된 소프트웨어 엔지니어는 만약 불완전한 데이터가 제공되면 그러한 시스템 메모리 과부하가 가능 했음을 알고 있었을 것으로 보인다.

대응 계획

Fontaine (2016)은 허용불가 그리고 허용가능 리스크에 대한 대응 계획 작성의 중요성을 설명한다. Lockheed Martin 이 ERAM 소프트웨어를 제작하기 시작했을 때, 그들이 그 리스크들을 식별했을 것은 거의 확실하다. 그러면 불완전하거나 상당한 데이터 입력으로 인한 메모리 부족의 가능성을 간과한 것인가?

프로젝트 관리자는 철저한 리스크 관리 접근을 보장하기 위해 다음 프로세스를 준수해야 한다:

1. 허용불가 리스크 식별
2. 허용가능 리스크 식별
3. 허용불가 리스크 대응 계획 작성
4. 허용가능 리스크 감소를 위한 비용-편익 분석
5. 리스크 레지스터 산출
6. 리스크 레지스터의 주기적 그리고 각 프로젝트 단계의 끝에 업데이트 (Fontaine, 2016)

이러한 것들이 실행되었다면, 데이터와 메모리 문제가 허용불가 기술 리스크로 식별되었을 것이고 좀 더 허용 가능한 리스크로 만들도록 설계 요구 사항을 업데이트 되었을 것이다.

예컨대, 더 많은 메모리 제공 또는 ERAM 소프트웨어에 필요한 모든 데이터를 제공하도록 비행 계획 요구하는 등, 비용-편익 분석은 기술 리스크에 대한 다양한 접근 방식과 관련된 비용을 확인할 수 있다. PMI (2017)의 리스크 전가의 한 방법으로써 비용-편익 분석은 필요한 데이터를 다른 쪽에서 입력하도록 함으로써 비용도 적게 들고, 종국에는 좀 더 효율적이 될것이며 허용불가 리스크를 허용가능한 리스크로 변환시킬 수 있을 것이다.

통계적 모델과 데이터 마이닝 모델과 같이 분석에 있어 좀 더 상세한 접근 방법들이 있는데, 이 정도 규모의 소프트웨어 프로젝트에 사용될 수 있다 (Hu et al., 2013). 선택한 방법들과 상관없이, 기본 전제는 동일하게 유지된다. 기술 리스크 계획 수립은 협상 할 수 없으며, 리스크를 평가하고 정기적으로 재평가하는 것은 프로젝트 관리자와 기타 다른 이해 관계자가 변화하는 조건이나 요구 사항을 평가할 수 있도록 하므로 매우 중요하다.

결론

요약하면, 프로젝트 리스크는 특히 복잡한 소프트웨어 프로젝트에 대해서는 피할 수 없다. 리스크는 프로젝트의 이러한 형태에 따라서 프로젝트 리스크와 기술 리스크로 분류되어야 하는데, 그렇게 함으로써 최종 제품에 대한 (리스크)완화가 설계 프로세스 동안 시작할 수 있다. 허용불가 기술

리스크는 첫 번째로 해결되어야 하고 대응 계획을 포함하며, 허용가능 리스크는 비용-편익 분석이나 다른 분석으로 평가하고 대응 계획을 포함한다. 지속적인 리스크 계획의 감시가 프로젝트 성공을 위해 중요하다.

References

1. Fontaine, M. (2016). Chapter 4 – Project risk management. In P. A. J. Green (Ed.), *Enterprise risk management: A common framework for the entire organization*. Waltham, MA: Elsevier.
2. Halsey, A. (2015, August 18). FAA pins Saturday's air travel debacle on glitch in new software. *The Washington Post*.
3. Hu, Y., Du, J., Zhang, X., Hao, X., Ngai, E. W. T., Fan, M., & Liu, M. (2013). An integrative framework for intelligent software project risk planning. *Decision Support Systems, 55*(4), 927–937.
4. Project Management Institute. (2013). *Software extension to the PMBOK® guide fifth edition*. Newtown Square, PA: Author.
5. Project Management Institute. (2017). *A guide to the project management body of knowledge (PMBOK® guide)* – Sixth edition. Newtown Square, PA: Author.
6. Samantra, C., Datta, S., Mahapatra, S. S., & Debata, B. R. (2016). Interpretive structural modelling of critical risk factors in software engineering project. *Benchmarking: An International Journal*, *23*(1), 2–24.
7. Scott, A. & Menn, J. (2014, May 12). Exclusive: Air traffic system failure caused by computer memory shortage. *Reuters*.
8. Vrhovec, S. L. R., Hovelja, T., Vavpotič, D., & Krisper, M. (2015). Diagnosing organizational risks in software projects: Stakeholder resistance. *International Journal of Project Management, 33*(6), 1262–1273.