DOI: 10.3745/KIPSTD.2009.16-D.1.105

위험대상요소 분석을 위한 프로세스 마일스톤에 관한 연구

이 은 서†

요 약

위험관리는 점점 더 프로젝트 관리자에게 있어서 중요한 일중의 하나로 되어가고 있다. 그것은 개발될 소프트웨어 품질 혹은 프로젝트 일정에 영향을 미칠 수 있는 위험을 예측하는 것을 포함한다. 위험 분석의 결과가 생길 수 있는 위험의 결과와 함께 프로젝트에 문서화되어야 한다. 효율적인 위험관리는 문제에 쉽게 대처할 수 있게 해주며, 그것이 수용할 수 없는 예산이나 일정 지연이 되지 않도록 해준다. 본 연구에서는 소프트웨어 개발 시, 프로세스 이정표와 노력에 관한 위험요소 분석에 대한 기준을 제시한다. 또한 이를 정량화 하여 전이단계를 제시한다.

키워드 : 위험관리, 소프트웨어 프로세스 개선, 결함관리, 생명주기, 이정표

A Study of Process Milestone for the Analysis of Risk Items

Lee Eun-Ser†

ABSTRACT

Risk management is increasingly seen as one of the main jobs of project managers. It involves anticipating risks that might affect the project schedule or the quality of the software being developed and taking action to avoid these risks. The results of the risk analysis should be documented in the project plan along with an analysis of the consequences of a risk occurring. Effective risk management makes it easier to cope with problems and to ensure that these do not lead to unacceptable budget or schedule slippage. This research provides criteria of analysis of risk items to the estimation of process milestone on software development. Also, In this paper propose to a fixed quantity and transition phase.

Keywords : Risk Management, Software Process Improvement, Defect Management, Life cycle, Milestone

1. 서 론

공학 계통의 프로젝트는 정해진 기간 안에 한정된 자원을 사용하여 질 좋은 제품을 생산하여야 한다. 하지만 모든 프로젝트는 원하는 제품을 생산하지 못하거나, 자원이 초과 사용되거나, 정해진 기간을 넘길 위험을 가지고 있다[1][2].

위험 분석은 프로젝트에 내재된 위험요소를 인식하고 그 영향을 분석하여 이를 관리하는 활동이다. 즉 프로젝트를 성공시키기 위하여 위험요소를 사전에 예측하여 대비하는 모든 기술과 활동을 포함한다. 따라서 이와 같은 위험요소를 관리하기 위하여 많은 활동과 시스템이 만들어 지고 있다.

결함 내성 시스템은 그 예로 일부 시스템 결함이 나타난 이후에도 계속 운영될 수 있는 시스템이다. 시스템의 결함 내성 메커니즘은 시스템 결함이 시스템 고장으로 귀결되지 않도록 보장하는 것이다. 시스템 고장이 큰 재앙을 유발하거나 시스템의 운영의 손실이 경제적으로 큰 손실을 유발하는 상황에서는 결함내성이 필요하다[3].

품질에 영향을 미치지 않게 하기 위하여 각 사항들이 위험요소로 전이되는 것을 예방하고 예측하여 품질관리를 하려는 노력이 대두되고 있다. 따라서 본 논문에서는 품질관리와 연관된 위험요소 발생의 진척상황을 프로세스 이정표 관점에서 분석하여 예방 및 원인 제거를 수행하였다. 이를 수행하기 위하여 수학적인 접근방법과 기준을 제시하고, 각 단계를 개발 과정을 기반으로 하여 위험관리를 수행하고자 한다. 또한 이를 활용하여 유사한 영역의 프로젝트에서 활용하여 결함을 관리할 수 있게 된다.

2. 기본 개념

2.1 결함

결함 정보는 다른 유형의 원시데이터이며, 소프트웨어 프로젝트에서 매우 중요한 것이다. 결함은 소프트웨어의 품질과 직접 관련이 있으므로 여러 의미에서 결함 데이터는 공수 데이터보다 더 중요하다[4~6].

† 종신회원 : 안동대학교 컴퓨터공학과 조교수
논문접수 : 2008년 12월 4일
수 정 일 : 1차 2008년 12월 19일
심사완료 : 2008년 12월 22일

결함 데이터는 우선 프로젝트 관리를 위해 필요하다. 대규모 프로젝트는 수천개의 결함을 포함할 수 있는데, 이 결함은 다른 사람들이 프로젝트의 각각 다른 단계에서 발견하게 된다. 흔히 프로세스에서 결함을 고치는 사람과 결함을 발견하거나 보고하는 사람은 서로 다르다. 보통 프로젝트는 소프트웨어가 최종 인도전에 발견한 모든 또는 대부분의 결함을 제거하려고 할 것이다. 이런 시나리오에서 결함 보고와 해결은 비공식적으로 수행할 수 없다. 비공식적인 메커니즘의 사용은 발견한 결함에 대해 잊어버리는 결과를 가져올 수 있으므로 결함을 제거하지 않거나 다시 찾는데 추가 공수가 들어갈 수도 있다. 그러므로 적어도 결함을 기록해야 하고 해결할 때까지 추적해야 한다. 이런 절차를 위해서 결함의 징후, 의심되는 결함의 위치, 발견자, 해결자 등과 같은 정보가 필요할 것이다. 따라서 결함이란 프로젝트의 작업 산출물에서 발견되는 것으로 이 때문에 프로젝트의 목표를 달성하는데 부정적인 영향을 끼칠 수도 있다[7][8].

결함에 대한 정보는 여러 가지 다양한 결함 검출 활동을 통해 발견한 결함의 개수도 포함 한다. 그래서 요구 사항 검토, 설계 검토, 코드 검토, 단위 테스트, 그리고 다른 단계에서 발견된 모든 결함 등을 기록한다. 프로젝트의 서로 다른 단계에서 발견된 결함 개수의 분포 데이터 역시 프로세스 역량 기준선(Process Capability Baseline) 생성에 사용한다[9][10]. 아울러 PDB(Process Data Base) 입력항목에 몇 가지 설명이 기록되는데, 예측에 대한 설명과 위험 관리에 대한 설명이 해당된다.

2.2 위험분석

위험이 소프트웨어 프로젝트에만 한정된 것은 아니다. 소프트웨어가 성공적으로 개발되고 고객에게 인도된 후에도 위험은 일어날 수 있다. 이러한 위험은 보통 현장에서의 소프트웨어 실패와 관련이 있다[11].

비록 잘 공학화된 시스템이 실패할 확률이 낮다고 해도, 컴퓨터-기저 제어 또는 감시 시스템에서 검출되지 않은 결점은 막대한 경제적 손실을 초래하거나 사람이 부상하거나 죽음을 초래할 만큼 더욱 나쁘고 심각할 수 있다. 그러나 컴퓨터-기저 제어와 감시 시스템의 비용상의 이점과 기능상의 이점은 보통 이러한 위험보다 증요하다. 오늘날 컴퓨터 소프트웨어와 하드웨어가 안전을 증시하는 시스템을 제어하기 위해 사용되고 있다.

소프트웨어 안정성과 위험분석은 소프트웨어에 부정적으로 영향을 미치는, 그리고 전체 시스템을 작동하지 않게 만드는 잠재적인 위험의 식별과 평가에 초점을 맞춘 소프트웨어 품질보증활동이다. 만약 소프트웨어 공학 프로세스의 초기에 치명적 위험을 식별할 수 있다면, 소프트웨어 설계 특징들을 명시할 수 있고, 잠재적인 치명적 위험을 제거하거나 조정할 수 있다.

2.3 소프트웨어 프로세스 개선의 필요성

프로세스를 개선하는 합리적인 방법은 프로세스의 특정한 속성들을 측정하고, 이들 속성에 근거해서 의미있는 메트릭스의 집합을 개발한 후에 특성에 관한 전략을 이끌어 내는 지표를 제공하기 위해서 이들 메트릭스를 사용하는 것이다. 그러나 우리가 소프트웨어 메트릭스와 소프트웨어 프로세스 개선에 있어서 영향을 말하기 전에, 프로세스는 소프트웨어 품질과 조직의 성능을 개선하는데 제어할 수 있는 많은 인자중의 하나라는 것이다[12][13]. 따라서 소프트웨어 품질에 많은 영향을 주는 인자가 결함관리가 된다. 소프트웨어 품질과 조직의 성능에 영향을 주는 것으로는 프로세스를 기반으로 제품, 사람, 기술이 된다. 또한 제품, 사람, 기술이 프로세스를 일반화하는 과정에서 많은 결함들이 발생하여서 전체적인 소프트웨어 프로세스의 수준을 격하 시키게 된다. 그러므로 결함 관리의 필요성이 요구되게 된다.

프로세스 개선을 위하여 소프트웨어의 실패 분석이 필요하며, 이를 위하여 다음과 같은 방법으로 수행된다.

- 모든 오류와 결함은 원인으로 분류한다.
- 각 오류와 결함을 수정한다.
- 각 범주에 속하는 오류와 결함을 계산하여 내림 차순으로 정리한다.
- 결과 자료는 조직에 가장 많은 비용이 드는 범주를 발견하기 위해서 분석한다.
- 계획은 가장 비용이 드는 오류와 결함의 종류를 제거할 의도로 프로세스를 수정하기 위해 개발된다.

기존의 위험요소를 측정하기 위해서 많은 노력을 하고 있다. 많은 기업에서 위험요소의 분석 시, 개발되는 생산품에 주요 기능에 해당하는 위험요소인지 부가기능에 의하여 주요 요소에 영향을 주지 않는 위험요소인지를 식별하고 있다. 이와 같은 연구방향은 불량률을 줄이기 위한 6-시그마와 같은 도입을 가져왔으며, 정량적인 위험요소 분석을 위한 필요성이 대두되게 되었다.

그러나 위험요소를 분석하여 연관성을 정량화 하여 이를 분석하기 위한 연구는 많이 진행 되어 있지 않고 있다. 따라서 본 논문에서는 위와 같은 위험요소의 연관성을 정량적으로 분석하기 위하여 연구를 수행하고자 한다.

3. 본론 및 사례 연구

본 장에서는 소프트웨어 프로세스 성숙도를 측정하기 위하여 위험요소를 기반으로 분석하였다. 위험요소를 정량적으로 분석하였고, 성숙도의 수준으로 분류를 하였다.

분류된 위험요소에 의하여 소프트웨어 프로세스 성숙도의 수준을 평가할 수 있도록 하였다. 또한 위험요소의 정량적인 평가에 의하여 소프트웨어 프로세스 성숙도를 평가 모델에 의하여 평가를 받지 않고도 수준을 예측하고자 한다. 따라서 수준을 예측하는 경우, 복잡한 과정이 아닌 간략화 된 과정을 통하여 평가 기간과 노력을 줄일 수 있게 된다.

3.1 소프트웨어 프로세스 성숙도 평가를 위한 위험요소의 정량적인 분석

소프트웨어 프로세스 성숙도는 많은 평가 모델을 통하여 측

〈표 1〉 위험요소 식별을 위한 전이단계

위험 활동내용 전이단계	상세 내용
1. 이슈단계	요구사항과 연관된 모든 사항을 이슈사항으로 분석함
2. 이슈사항의 연결그룹 분석단계	분석된 이슈사항을 요구사항과 연관된 카테고리별로 그룹화 함
3. 문제요소 분석단계	분석된 이슈 사항에서 요구사항 실현에 영향을 미치는 요소
4. 문제요소의 연관성 분석단계	분석된 문제요소에서 서로의 연관성 분석
5. 결함요소 분석단계	문제요소가 실제 구현되는 기능상에서 실행을 저해하는 요소
6. 결함요소의 인과관계 분석단계	결함요소간의 인과관계 분석
7. 위험요소 분석단계	일정, 비용, 품질에 영향을 미치는 요소

정을 해왔다. 평가모델로는 ISO/IEC 15504와 CMMI(Capability Maturity Model Integration)가 대표적이다.

소프트웨어 프로세스 성숙도에서 가장 많은 영향을 미치는 것은 위험요소로서 일정과 품질과 비용에 영향을 주게 된다. 따라서 일정, 품질과 비용에 발생하는 문제를 해결하기 위한 노력이 요구되게 된다. 본 논문에서는 이전 연구를 개선하여 위험요소의 정량적인 분석을 가능하게 하고 소프트웨어 프로세스의 성숙도를 측정하고자 한다.

프로세스의 성숙도를 정량적으로 평가하기 위하여 위험요소의 영향을 분석하였다. 위험요소는 다음과 같이 분석하였다[14].

추이분석을 위하여 선행되어야 할 사항은 위험요소로 전이되는 단계를 정의하는 것이다.

전이되는 단계를 정의함으로서, 위험요소로 전이되는 상황을 판단할 수 있으며, 이를 활용하여 위험요소로 전이되는 것을 예방하고 대비할 수 있게 된다. 또한 관리자나 개발자는 현재 자신과 연관된 위험요소를 파악하여 연관된 요소들을 관리하여 위험요소의 확산이 되지 않도록 할 수 있게 된다. <표 1>과 같은 전이 단계를 제시한다.

- ✱ 이슈단계는 요구사항과 연관된 모든 사항을 이슈사항으로 분석을 하게 된다. 따라서 요구사항을 시스템으로 기능화하기 위하여 필요한 세부기능의 사항도 포함하게 된다. 이를 위해서는 영역분석을 수행하면서 요구사항 분석이 병행되어야 한다. 또한 각 이슈사항을 비 기능적 및 기능적으로 분류를 해서 추출을 하고 이를 활용하여 위험요소가 시스템에 연관된 사항인지 고객의 요구사항에 연관된 사항인지를 구분해야 한다. 이와 같은 구분은 하드웨어와 관련된 위험요소인지 소프트웨어와 연관된 위험요소인지를 구분하는 중요한 자료로 활용되게 된다.
- ✱ 이슈사항의 연결그룹 분석단계는 요구사항 분석과 연관되어서 기능별로 같은 범주에 속하는 것을 구별하는 것이다. 이와 같은 분석과정은 기능에 문제가 발생한 경우에, 문제가 발생한 부분을 손쉽게 찾아서 유지 및 보수를 원활히 할 수 있게 된다. 따라서 기능의 범주를 추출하기 위하여 기능별로 그룹화를 수행해야 하며, 이를 토대로 추가되는 기능에 대하여서는 주기적으로 범주와 기능이 추가되었는지를 확인해야 한다.
- ✱ 문제요소 분석단계는 요구사항을 시스템으로 기능화하여 구현하는 과정에서 구현에 중요한 요인이 무엇인지를 분석하고 추출하는 과정이다. 이와 같은 분석과정은 기능을 완성하는 과정에서 중요한 기능이 아님에도 프로젝트나 개발에 치명적인 요소가 발생하는 경우, 이를 대비하기 위한 단계이다.
- ✱ 문제요소의 연관성 분석단계는 요구사항을 시스템으로 기능화 하여 구현하는 과정에서 발생하는 영향에 대하여 분석을 하는 단계이다. 시스템에 구현하는 과정에서 문제가 발생하는 경우, 다른 기능에서 그 영향을 받는다면 위험요소를 그대로 다른 기능에 전이하게 된다. 이와 같은 상황을 예방하기 위하여 문제요소 분석단계를 거쳐 예방을 하기 위함이다.
- ✱ 결함요소 분석단계는 각 단계에서 가장 중요시하고 노력승수를 많이 투여해야 하는 단계이다. 그 이유는 이전단계에서는 위험요소로 전이되는 것이 명확하게 나타나지 않게 된다. 위험요소로 전이되는 것은 결함요소로 발생하면서 실제적인 기능상의 오류나 문제를 야기하게 된다. 따라서 결함요소 분석단계에서부터는 좀 더 명확하고 상세한 결함 관리가 수반되어야 한다. 결함관리를 위하여 결함종류를 분석하고 결함의 영향 정도를 파악하여 주 기능을 저해하는 결함인지 주 기능과 연관성이 없는 결함인지를 파악하여 결함정도가 미치는 영향을 정량화하여 관리하여야 한다.
- ✱ 결함요소의 인과관계 분석단계는 실제로 위험요소 전이가 되는 결함간의 연관성을 분석하는 단계이다. 결함의 연관성을 분석하여 발생하는 결함의 추적성과 수정을 동시에 수행할 수 있게 된다.
- ✱ 위험요소 분석단계는 발생된 요소가 일정을 지연시키고, 제한된 개발비용이나 프로젝트 비용을 초과하며, 초기 개발 목적에서 요구되었던 품질에 미치지 못하게 되는 경우가 발생하지 않도록 하기 위한 단계이다.

위험요소 식별을 위한 단계는 위와 같이 제시하였으며, 이를 토대로 하여 사례연구를 수행하였다.

이전 연구에서는 위험요소 식별을 위하여 전이 단계를 제시하였다[14]. 이를 토대로 소프트웨어 프로세스의 성숙도를 측정하기 위한 마일스톤을 제시하고자 한다.

3.2 프로세스 성숙도 측정을 위한 마일스톤

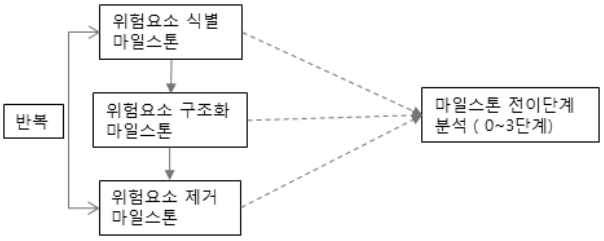
3.1절에서 제시된 위험요소 식별을 위한 전이단계는 개발관점을 기반으로 제시된 연구이다. 그러나 위험요소를 중심으로 개발관점을 분석하기 위한 마일스톤이 요구되게 된다. 따라서 본 절에서는 위험요소 중심으로 프로세스 성숙도를 측정하기 위한 마일스톤을 제시하고자 한다. 마일스톤은 3단계로 제시하였다. 각 단계의 내용은 다음과 같다.

<표 2>에서와 같이 위험요소를 식별하기 위하여 3단계를 제시하였다.

위험요소 식별 (RII) 마일스톤은 위험요소를 식별하고, 현 상태의 위험요소를 이슈, 문제, 결함, 위험요소로 구분하게 된다. 또한 위험대상요소가 다른 요소에 영향을 주는가를 식별하여 위험이 전이되는 것을 분석한다. 전이되는 과정에서 연관된 기능이 저하되는 원인이 되므로, 해당 기능에 대한 대비를 하고 인식을 위하여 위험대상요소와 연관된 기능의 파악도 동시에 수행된다.

위험요소 구조화 (RIA) 마일스톤은 위험요소가 식별된 이후에 진행되게 된다. RII 마일스톤에서는 위험대상요소 각각에 대한 분석과 식별작업이 이루어지지만, RIA 마일스톤에서는 식별된 각각의 위험대상요소간의 상호 연관성에서 발생하는 문제를 다루고 있다.

RIA 마일스톤은 식별된 위험대상요소의 연관성을 분석하여 구조화한다. 구조화시 계층을 분석하여 단계별로 그룹화하여 해당 위험대상요소가 핵심기능인지 응용과 연관된 기능인지를 파악한다. 또한 위험대상요소가 제거되는 과정에서 다른 기능에 영향을 주거나 다른 위험대상요소에 영향을 줄 수 있다. 이와 같은 상황을 위하여 위험대상요소간의 상충관계를 파악하여 2차적인 위험대상요소가 생성되는 것을 예방하고자 한다.



(그림 1) 위험대상요소 마일스톤 운용

위험요소제거 (RIR) 마일스톤은 구조화와 계층화된 위험대상요소를 최종적으로 제거하는 단계이다. 그러나 제거가 완전히 이루어지지 않는 경우가 발생하게 되므로, 남아있는 위험대상 요소를 파악하여 현 개발과정에서 해결 가능한 위험대상요소인지 해결이 불가능한지를 분석해야 한다. 이때 구조적으로 해결이 불가능한 경우를 분석하여 해당 위험대상요소를 해결하는데 요구되는 노력과 시간을 절약할 수 있다. 구조적으로 해결이 불가능한 경우는 환경과 인적자원, 하드웨어와 연관된 문제를 제시할 수 있다.

마일스톤에 의하여 위험대상요소를 관리하는 경우 각각의 마일스톤마다 위험대상요소의 정도에 따라서 위험대상요소의 위험과 전이 정도가 다를 수 있다. 이를 위하여 각각의 마일스톤마다 상태에 따라서 위험과 전이 정도를 파악할 수 있게 된다. 마일스톤의 정도는 다음과 같이 4단계로 제시하였다.

<표 3>은 <표 2>와 동시에 수행되게 된다. 예를 들면, RII 마일스톤의 정도를 파악하는 과정에서 위험대상요소가 전혀 식별되지 않은 경우에는 RII(0)으로 표시를 하여 위험정도의 진척도를 파악할 수 있게 된다. RII(3)의 경우는, 위험요소가 제거된 경우를 의미한다. 각각의 마일스톤은 한번만 수행하는 것이 아니라, 위험요소를 제거하기 위하여 반

〈표 2〉 위험요소 식별을 위한 마일스톤

위험요소 분석을 위한 마일스톤	내용	분석요소
1. 위험요소 식별 (Risk Item Identification : RII)	- 위험대상요소를 식별함 - 이슈, 문제, 결함, 위험요소로 정도를 구분 - 위험대상요소의 영향 분석 - 위험대상요소와 연관된 기능 파악	개별 위험대상요소
2. 위험요소 구조화 (Risk Item Architecture : RIA)	- 위험대상요소의 상호연관성 파악 - 식별된 위험대상요소의 구조화 - 위험대상요소의 상충관계 파악	다수의 위험대상요소 (1:다, 다:다, 다:1, 1:1)
3. 위험요소 제거 (Risk Item Remove : RIR)	- 위험대상요소를 제거함 - 남아있는 위험대상요소의 영향 파악	구조화된 위험대상요소의 그룹

〈표 3〉 마일스톤의 위험정도에 따른 전이단계

위험대상요소 분석을 위한 마일스톤의 전이 단계	내용
0단계	- 위험대상요소의 영향이 전혀 식별되지 않음
1단계	- 위험대상요소를 식별함
2단계	- 위험대상요소의 연관성을 분석함
3단계	- 위험요소가 제거됨

복적으로 수행된다. 따라서 동일한 마일스톤이 반복 수행되는 경우에, 마일스톤의 진척도를 파악할 수 있게 된다.

<표 2>와 <표 3>을 기반으로 전반적인 마일스톤 운용 알고리즘을 도식화하면 다음과 같다.

3.3 위험대상요소를 위한 마일스톤의 정량적인 측정방법

위험대상요소의 전이정도는 다양한 형태로 측정할 수 있다. 본 절에서는 3.2절에서 제시된 마일스톤을 정량적으로 측정하기 위한 방법을 제시하고자 한다.

위험대상요소의 마일스톤을 측정하기 위하여 현 상태의 파악을 선행되어야 한다. 현 상태의 위험정도를 파악하기 위하여 다음 수식을 제시하였다.

$$RM(t,p)=\prod_{i=1}^{3}C(i)+\prod_{j=1}^{4}D(j)\bullet EF(t,p)$$

RM : 위험대상요소 마일스톤 산출
t : 형태
p : 단계
C( ) : 위험대상요소별 개발진척 영향정도
D( ) : 위험대상요소별 영향정도
EF : 위험대상요소를 해결하기 위한 노력승수
i : 위험대상요소 마일스톤의 각 단계 색인
j : 위험대상요소 마일스톤 전이단계의 색인

정량화된 수식은 위험대상요소의 영향정도를 측정하여 이를 해결하기 위한 노력승수를 포함하여 마일스톤을 산정하게 되어 있다.

위험대상요소 마일스톤을 정량적으로 측정하기 위하여 위와 같은 식을 제시하였다.

수식에서는 위험요소별 측정을 단계별로 산출하게 된다. 이와 같은 이유는 각 요소별로 위험요소로 전이되는 것을 측정하기 위함이다. 수식을 구성하는 각 요소를 설명하면 다음과 같다.

t는 위험대상요소의 형태를 구분하기 위함이다. t에 의하여 위험요소가 어떤 형태인지 구분하고 분류할 수 있다. 형태는 영역에 따라서 형태가 달라지므로 영역과 개발하는 프로젝트의 형태에 따라서 달라진다.

p는 위험요소가 전이되는 단계를 나타낸 것이다. 따라서 현재 요소가 어디까지 전이되었는지 단계를 제시하여 어느 정도까지 전이가 되었는지 인지할 수 있다.

C( )는 위험대상요소별 개발진척 영향 정도이다. 개발 과정에서 발생하는 위험대상요소는 개발과정에 많은 영향을 주게 된다. 그 분야는 비용과 일정, 품질이다. 따라서 위험대상요소가 발생되면 비용, 일정, 품질에 영향을 주어서 개발진행과정에 영향을 주게 되어 프로젝트가 실패하게 되는 원인이 된다. 개발진척 영향정도를 측정하기 위하여 계획대비 일치하지 않는 비율을 측정하게 된다. 계획대비 지연된 일정은 예상된 산출을 기준으로 측정을 하고자 한다. 산출물이 만들어진 것과 계획 시에 작성한 산출물 목록을 기준으로 계획대비 지연일정을 산출하여 C( )의 가중치를 제시한다.

위험대상요소 중에서 개발진행과정에 영향을 주는 정도를 확인하는 것이 중요한 요인이 된다. 영향 정도에 따라서 프로젝트에 치명적인 요인을 확인하여 위험대상요소를 중심으로 한 마일스톤을 측정할 수 있게 된다.

D( )는 위험대상요소 자체의 결함으로 전이되는 영향을 측정하기 위한 것이다. 따라서 각 결함이 얼마나 영향을 주는지를 측정하기 위하여 다음의 기준을 제시하였다.

〈표 4〉 위험대상요소의 개발진척 영향 정도

계획대비 지연일정	C( ) 가중치
71~100%	0.9
51~70%	0.7
31~50%	0.5
16~30%	0.3
0~15%	0.1

〈표 5〉 위험대상요소 영향도 산출기준

결함 영향	D( ) 가중치
상	상(0.9)
	중(0.8)
	하(0.7
중	상(0.6)
	중(0.5)
	하(0.4)
하	상(0.3)
	중(0.1)
	하(0.1)

〈표 6〉 위험대상요소를 해결하기 위한 노력승수

EF	환경(0.5)	개인능력(0.5)
산출(환경 + 개인능력)	상(0.1) - 모든 것이 구축됨	상(0.1) - 동일분야 7년이상 경험
	중(0.3) - 요구하여 지원이 됨	중(0.3) - 동일분야 3~5년 경험
	하(0.5) - 지원이 되지 않음	하(0.5) - 동일분야 1~3년 경험

〈표 7〉 위험대상요소 마일스톤별 가중치

위험대상요소 마일스톤	가중치
위험요소식별 (RII)	0.9
위험요소구조화 (RIA)	0.5
위험요소제거 (RIR)	0.3

〈표 8〉 마일스톤의 위험정도에 따른 가중치

마일스톤	가중치
0	0.9
1	0.5
2	0.3
3	0.1

결함의 영향을 파악하기 위하여 결함이 주요한 기능인지 부가적인 기능인지를 측정하여 파악하고자 한다. 따라서 개발과정에서 기준을 정한 우선순위를 참조하여 결함이 주요한지 부가적인지를 파악하게 된다. <표 5>에서는 주요한 결함을 상으로 분류하고 중간은 중, 부가적인 결함은 하로 분류하였다. 결함의 영향정도를 상, 중, 하로 분류하였으며, 세부기준으로 상, 중, 하를 분류하여 가중치를 적용하였다.

EF는 위험대상요소가 발생하여 위험요소로 전이 되거나 영향을 주는 경우에, 이를 해결하기 위한 노력 승수를 의미한다. 치명적인 위험대상요소는 개발내용에 내재되어 있는 경우, 개발 후반부에 많은 영향을 미치게 된다. 따라서 이를 제거하고 완화시키는 작업이 필요하게 되며, 추가적으로 인력과 시간과 비용과 같은 노력이 요구되게 된다.

위험대상요소를 효과적으로 제거하기 위해서는 환경과 개인의 능력에 관한 인프라를 측정해야 한다. 따라서 인프라를 측정하기 위하여 환경에서는 하드웨어와 소프트웨어의 지원 체계와 능력을 측정한다. 개인 능력에서는 주로 프로그래밍 능력 및 관리 능력을 평가한다.

전체 수식은 위험대상요소 마이스톤을 산출하기 위해서 환경과 개인능력이 상이지만 가중치가 낮게 부여된다. 따라서 위험대상요소에 의한 마일스톤의 정량화에서 환경과 개인능력이 높으면 위험요소가 낮게 산출되게 된다.

i는 위험대상요소 마일스톤의 3단계(위험요소식별, 위험요소구조화, 위험요소제거)를 표현하기 위한 색인이다.

j는 위험대상요소 마일스톤의 각 단계에서 전이되는 4단계(0~3단계)를 표현하기 위한 색인이다. 4단계는 <표 3>에서 제시하였다.

3.3 절에서 제시된 수식을 기반으로 위험대상요소 마일스톤을 정량적으로 분석하고자 한다.

4. 사례 연구

3장에서 제시된 위험대상요소를 위한 마일스톤의 정량적인 산출을 위하여 실제 과제에 적용하였다. 적용분야는 시스템 통합과 연관된 과제 개발에 적용하였다. 적용 과제는 개발 생명주기를 반복(iteration)의 방법을 사용하고 있었다. 즉, 동일한 생명주기를 반복적으로 수행해서 위험요소를 줄이고자 했다. 반복적으로 수행된 생명주기를 네 부분으로 나누어서 산출하였으며, 연산 과정에서 발생하는 소수점은 첫째 자리에서 반올림하였다.

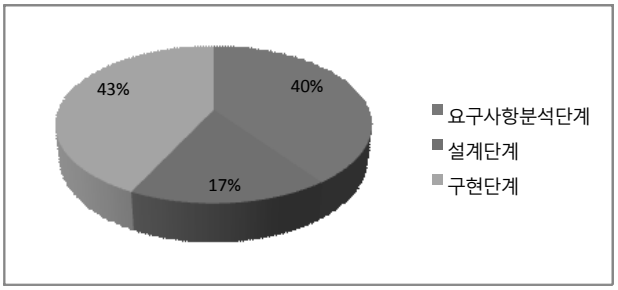
산술식은 3장에서 제시하였으며, 각 항목과 식은 다음과 같다.

$$RM(t,p)=\prod_{i=1}^{3}C(i)+\prod_{j=1}^{4}D(j)\bullet EF(t,p)$$

RM : 위험대상요소 마일스톤 산출
t : 형태
p : 단계
C( ) : 위험대상요소별 개발진척 영향정도
D( ) : 위험대상요소별 영향정도
EF : 위험대상요소를 해결하기 위한 노력승수
i : 위험대상요소 마일스톤의 각 단계 색인
j : 위험대상요소 마일스톤 전이단계의 색인

〈표 9〉 위험요소 산출 자료

단계	C( )	D( )	EF	위험대상요소 마일스톤별 가중치(i)	마일스톤의 위험정도에 따른 가중치(j)
요구사항 분석	0.1	0.8	0.6	0.9	0.5
설계	0.5	0.8	0.8	0.5	0.3
구현	0.7	0.5	0.4	0.3	0.1



(그림 2) 단계별 위험요소 분포도

1) 요구사항 분석 단계
(0.1 X 0.8 X 0.9) + (0.6 X 0.5) = 1.02
2) 설계 단계
(0.5 X 0.8 X 0.5) + (0.8 X 0.3) = 0.44
3) 구현 단계
(0.7 X 0.5 X 0.3) + (0.4 X 0.1) = 1.09

<표 7>은 각 항목을 기준으로 프로세스의 성숙도를 측정하였다. 산출된 수치가 높을수록 위험에 노출된 정도가 크며, 산출결과에 의하여 요구사항분석과 구현단계에서 위험요소에 많이 노출이 되어 있는 것으로 분석되었다. 이에 반하여 설계 단계는 0.44로 위험관리를 충분히 수행하는 것으로 분석되었다.

산출결과에 따라서 과제 수행 중 발생하는 위험요소는 요구사항분석과 구현단계에서 발생되는 것으로 설명할 수 있으며, 이에 대한 대안으로는 요구사항분석과 구현단계에서 위험요소를 줄이기 위한 생명주기를 채택하여 추가적인 위험관리를 수행해야 한다.

<표 7>의 결과를 기반으로 단계별 위험요소를 분석하였다. (그림 2)와 같이 전체 단계에서 요구사항분석과 구현단계에서 위험요소를 많이 내포하고 있음을 알 수 있다. 구현단계에서는 구현기술의 경험부족과 요구사항과 설계의 위험요소가 전파된 경우가 원인이 되었다.

5. 결론 및 향후 연구 방향

본 논문에서는 소프트웨어 프로세스 성숙도를 측정하기 위하여 위험요소를 기반으로 분석하였다. 위험요소를 정량적으로 분석하였고, 성숙도의 수준으로 분류를 하였다.

분류된 위험요소에 의하여 소프트웨어 프로세스 성숙도의 수준을 평가할 수 있도록 하였다. 또한 위험요소의 정량적인 평가에 의하여 소프트웨어 프로세스 성숙도를 평가 모델에 의하여 평가를 받지 않고도 수준을 예측하고자 한다. 따라서 수준을 예측하는 경우, 복잡한 과정이 아닌 간략화 된 과정을 통하여 평가 기간과 노력을 줄일 수 있게 된다.

향후 연구로는 현재 제시된 정량화 식을 자동으로 산출할 수 있는 도구를 개발하는 것이다. 또한 산출식의 정확도를 검증하여 신뢰도를 높이고 이를 위한 보정작업과 함께 여러 과제에 적용하는 과정이 요구되며 이를 계획하고 있다.

참 고 문 헌

[1] 최은만, 소프트웨어공학론, 사이텍미디어, January, 2001.
[2] 윤청, 성공적인 소프트웨어 개발 방법론, 생능출판사, August, 1999.
[3] 권기태, 남영광, 소프트웨어공학, 홍릉과학출판사, February, 2008.
[4] Software process improvement forum, KASPA SPI-7, 2002.
[5] 신경애, “결함중요도 단계를 고려한 소프트웨어 신뢰도 성장 모델에 관한 연구”, 컴퓨터 산업교육기술학회 논문지, Vol.3 No.7 pp.0837-0844, 2000.
[6] Robert h. dunn, “Software defect removal,” Mcgraw-hill, 1984.
[7] Ram chillarrege, Kothanda r. prasad, “Test and development process retrospective a case study using ODC triggers,” IEEE computer society, 2002.
[8] Wohlin, Runeson, “Defect content estimations from review data,” Proceedings international conference on software engineering ICSE pp.400-409, 1998.
[9] Gaffney, John, “Some models for software defect analysis,” Lockheed martin, 1996.
[10] B. compton and C. withrow, “Prediction and control of ada software defects,” J. systems and software, Vol.12, pp. 199-207, 1990.
[11] Roger s. pressman “Software engineering” Mcgraw-hill international edition, 1997.
[12] Tim kasse, “Actin focused assessment for software process improvement,” Artech house, 2002.
[13] Paulish, and Carleton, “Case studies of software process improvement measurement,” Computer, Vol.27, No.9, 1994.
[14] 이은서, 위험요소 상태분석에 의한 프로세스 개선에 관한 연구, 제15-D권 제4호, pp.523-530, 정보처리학회논문지D, 2008. 08.

이 은 서

e-mail : eslee@andong.ac.kr
2001년~현 재 ISO/IEC 15504 국제 심사원
2004년 중앙대학교 컴퓨터공학과(박사)
2004년~현 재 임베디드 산업협회 전문위원
2004년~현 재 한국정보통신기술협회 위원
2005년~2007년 숭실대학교 정보미디어기술연구소 연구교수
2008년 3월~현 재 안동대학교 컴퓨터공학과 조교수
관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)