

2013. 5. 27. [제54호]


# 스크럼 개발방법에서의 측정지표에 관한 분석 및 도입방안

소프트웨어공학센터 경영지원TF팀

## C o n t e n t s

- I. 애자일 방법론의 한계 및 대안
- II. 스크럼 및 소프트웨어 측정 프레임워크  
이해와 측정지표 분석
- III. 스크럼 개발방법 도입방안

## I. 애자일 방법론의 한계 및 대안

 늘날 소프트웨어 개발은 시장환경의 변화에 따라 사용자의 요구사항에 빠르게 대응하고 기술 변화에 유연하게 대응할 수 있어야 한다. 이러한 소프트웨어의 새로운 변화에 따라 계획 중심의 전통적인 개발 방법보다는 애자일(Agile)과 같은 가볍고 변화를 수용하는 개발방법론을 활용하여 개발하는 사례가 증가하고 있다. 최근에는 신속한 변화와 반응이 중요한 웹 서비스 분야뿐만 아니라 게임, 임베디드 및 SI 분야에서도 애자일 도입이 확산되는 추세이다.

애자일(Agile)은 전통적 소프트웨어 개발방법론의 대안을 모색해오던 전문가들이 모여 가벼운 프로세스 방법론들의 공통적 특성을 정의하여 지칭하는 말로 사용되었다. 애자일 개발방법 중에서 가장 많이 알려진 것은 Extreme Programming 과 Scrum이다(Marcal et al., 2008). ‘극한 프로그래밍’이라고 불리는 Extreme Programming(XP) 방법론은 설계-구현-테스트 주기를 극한적으로 짧게 함으로써 고객이 원하는 양질의 소프트웨어를 이른 시일 내에 전달하기 위한 접근이다. 이에 비해 Scrum은 일정한 역할을 갖는 구성원이 참여하는 프로세스를 통해 문제를 발견하고 수정하는 작업을 되풀이 하며 개발해 나가는 방법을 의미한다.

애자일 방법론의 공통적 특징은 몇 가지로 요약할 수 있다. 첫째, 경험에서 시작된 방법들이라는 점이다. XP나 Scrum의 경우 프로젝트 경험을 바탕으로 발전하였으며, Crystal은 Cockburn이 여러 프로젝트 참여자들과 인터뷰와 관찰을 통해 시작되었다. 둘째는 가벼우며 실천적이고 실무적이다. 셋째는 의사소통과 협력을 통한 상호작용을 강조한다. 마지막으로 짧은 반복을 통하여 얻은 경험을 최대한 반영하여 개선해 나가는 것을 강조하는 점이다.

최근 들어 조직에서 애자일 방법론을 적용하는 사례가 증가하고 있다. Ambler는 2012년 113개 기업을 대상으로 한 애자일 설문 조사 결과를 Agility at Scale Survey 보고서에서 제시하였는데, 이에 따르면 최소 86% 조직에서 애자일 기법을 적용해 본 것으로 나타났다. 이는 기존의 무거운 방법론의 한계점을 극복하기 위한 대안으로 애자일 방법론의 가능성을 잘 보여주고 있는 사례이다.

애자일 방법론의 한계점도 여러 연구보고서에서 나타나고 있다. 무거운 기존 방법론과 비교하여 애자일 방법론은 대규모 조직에 적용하기 어려우며 고품질 소프트웨어 개

발에는 적합하지 않으며, 요즘과 같이 경쟁이 치열한 환경하에서는 소프트웨어의 성능과 품질이 강화되어야 하는데, 애자일의 경우 개발 속도를 높이는데만 치중하고 있다고 지적하였다. 따라서 애자일을 적용할 때는 성능과 품질에 대한 정량적 측정이 수반되어야 함을 강조하였다.

Mar al et al.(2008)은 세부 프로세스 관점에서 비교를 시도하였다. 무거운 프로세스로 잘 알려진 CMMI(Capability Maturity Model Integration) 품질개선 프레임워크와 스크럼을 일대일로 비교 평가하여 애자일의 한계점을 제시하였다. CMMI의 한 영역인 프로젝트 관리 차원에서 보면, 스크럼이 Level 3 이상의 프로세스 성숙도를 충족시키기 어렵다고 주장하였다. 이러한 차이의 원인으로는 프로젝트 산정과 비용 예측, 위험관리, 비용 수립과 통제, 프로젝트 데이터 관리, 프로젝트 통합 관리 등에 대한 프랙티스의 부족을 들었다.

애자일의 한계점을 다룬 이들 연구는 구체성 측면에서 부족하다고 말할 수 있다. 예를 들면 애자일은 측정이 약하다는 주장만 하였지, 구체적으로 어떤 관점과 기준에서 부족한지를 제시하지 못하였다. 아울러 무거운 방법론의 시각에서 애자일을 비교평가하다 보니 애자일 고유의 특성과 가능성이 간과되기도 하였다. 이런 이유로 인해 애자일 한계점을 지적한 이들 연구는 애자일 방법을 고려하는 기업에게 혼돈을 주고 있다고 본다. 즉, 애자일은 약점이 많을 뿐만 아니라 취약점을 해결하기 위한 대안이나 방안도 존재하지도 않는다는 잘못된 인식을 초래할 수 있기 때문이다.

이에 본 보고서에서는 구체적인 측정 기준에 입각하여 애자일을 분석함으로써 애자일의 강점, 약점 그리고 이를 극복하기 위한 대안을 모색하고자 한다. 즉, 소프트웨어 품질 관점에서 애자일을 평가하고 나아가서 품질 개선 방안을 제시하고자 한다.

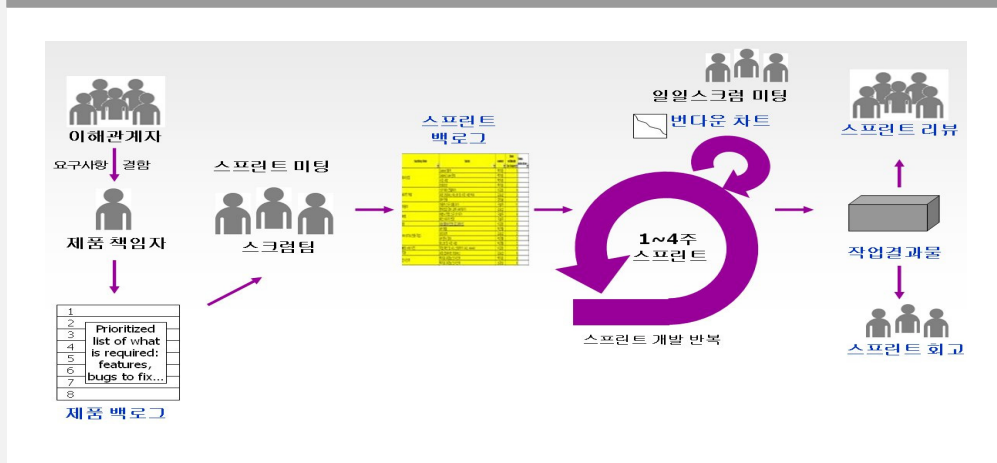
## II. 스크럼 및 소프트웨어 측정 프레임워크 이해와 측정지표 분석

**본** 보고서에서는 애자일 개발방법의 특징을 분석하고 이를 적용하는 방법을 이해하기 위하여 현업에서 적용하기 쉽고 실용적이며 가장 폭넓게 사용되고 있는 스크럼 개발방법을 선택하여 이에 대한 측정프레임 워크와 지표를 제시하였다.

## 2.1 스크럼 소개

스크럼(Scrum)이란 용어는 럭비 경기에서 반칙 등으로 인해 게임이 일시 중단 이후 경기를 다시 시작할 때 양팀 선수들이 대형을 짜는 것을 의미한다. 스크럼은 1986년 Takeuchi와 Nonaka가 일본에서부터 사용하기 시작한 제품 개발 프로세스 유형을 가리키는 용어로 사용되었다. 즉, 생산성이 높았던 일부 기업들은 공통적으로 제품 개발 프로세스를 반복적이고 점진적인 프로세스를 밟아나가며 동시에 구성원들의 창의적이고 자율적인 참여와 협업을 강조하고 있음을 발견하였다. 스크럼의 가장 중요한 특징은 요구사항을 수렴한 후에 가장 중요한 요구사항 기능의 개발 및 테스트를 위해 매우 짧은 반복주기(이를 Sprint라고 하며 길어야 2~4주)를 가동한다. 이 Sprint를 통해 나온 결과물을 사용자와 함께 검토한 피드백을 다시 요구사항에 반영하며 이러한 프로세스를 요구사항이 충족될 때까지 반복하여 수행한다. 이 스크럼 프로세스는 [그림 1]에 묘사되어 있다.

그림 1\_스크럼 프로세스



스크럼의 산출물을 이해하기 위하여 개발 프로세스에 대하여 보다 상세하게 설명하고자 한다. 스크럼 프로젝트는 처음에 제품 책임자가 요구사항 목록에 우선순위를 매겨 제품 백로그(backlog)를 작성함으로써 시작된다. 스프린트 미팅에서는 이 제품 백로그로부터 수행할 개발 항목을 결정하고 스프린트 백로그를 작성한다. 이 과정에서 스크럼 마스터(master)는 팀원들을 도와서 스크럼이 잘 수행될 수 있도록 프로젝트의 장애물을 제거하고 결정을 내릴 수 있도록 도와준다. 스프린트 개발 주기 동안 매일 스크럼 미팅

을 갖는다. 이 과정에서 스크럼 마스터는 개별 팀원에 대한 진척 상태를 확인하고 번다운 차트(Burn-down Chart)에 진척도와 남은 작업, 진행속도를 업데이트 한다. 스프린트가 종료되면 전체 팀원들과 스프린트 리뷰(review) 및 스프린트 회고(retrospective)을 통해 결과물을 확인하고 문제점과 다음 스프린트를 위한 개선 사항 등을 점검한다. 이러한 짧은 반복 개발 주기를 통하여 사용자의 변경에 민첩하게 대응할 수 있게 하고 프로젝트의 유연성을 높일 수 있도록 만들어 준다.

스크럼 활동은 몇 가지 주요한 산출물을 만들어 낸다. 우선 제품 백로그는 사용자 요구 기능을 우선순위로 정리한 내용을 담고 있다. [표 1]에서 보는 바와 같이 해당 요구사항의 내역과 대략적 작업 규모 등이 포함된다.

표 1\_제품 백로그

| 우선 순위 | 요구사항  | 요구사항 내역                   | 작성일   | 작성자 | 작업크기 | 비고                           |
|-------|-------|---------------------------|-------|-----|------|------------------------------|
| 1     | 예약 전송 | 사용자가 예약한 시간에 메시지 자동 전송 수행 | 10/21 | 황순삼 | 2일   | 예약시간은 HH:MM체계로 표현 (예: 15:30) |

스프린트 백로그는 [표 2]에서 보는 바와 같이 스프린트에서 수행할 업무 목록, 이를 구현하기 위한 작업 항목, 작업자, 그리고 예상 작업 시간을 포함하고 있다. 이를 통해 일정과 자원, 그리고 진행 상황을 파악할 수 있다.

표 2\_스프린트 백로그

| 백로그 항목 | 작업 항목                            | 작업자 | 작업산정 (시간) |
|--------|----------------------------------|-----|-----------|
| 예약 전송  | 1. 예약 시간 입력 화면 개발                | 홍길동 | 0.5 시간    |
|        | 2. 예약 시간 완료 메시지 창 개발             | 홍길동 | 0.5 시간    |
|        | 3. 예약 시간 입력 error checking 로직 개발 | 홍길동 | 1 시간      |

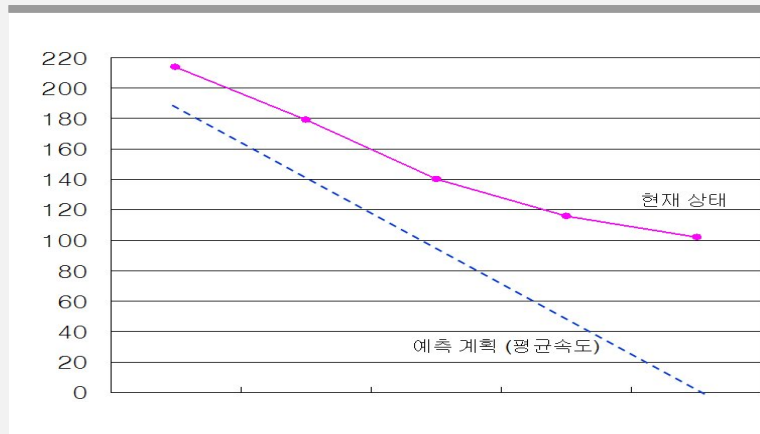
이러한 스프린트 백로그는 스프린트 기간 내의 진행 사항 관리에 기초가 된다. [표 3]에서 보는 바와 같이, 작업항목에 대하여 매일 남은 작업 시간을 기록하여 관리한다.

표 3\_스프린트 백로그 진척 관리

| 작업항목                | 작업자 | 1일  | 2일  | 3일  | 4일 |
|---------------------|-----|-----|-----|-----|----|
| 1. 예약시간 입력 화면 개발    | 홍길동 | 0.5 | 1   | 0.5 | 0  |
| 2. 예약시간 완료 메시지 창 개발 | 홍길동 | 0.5 | 0.5 | 0   | 0  |
| 남은 총 작업             |     | 1   | 1.5 | 0.5 | 0  |

스프린트를 통하여 시각적으로 보여주기 위해 번다운 차트를 만들어 낸다. 일일 단위로 팀원의 남은 업무량을 계산하여 진척 현황을 정량적으로 파악할 수 있도록 해준다. 이를 통해 다음과 같은 정보를 알 수 있다: 첫째, Y축을 통해 남은 업무량과 프로젝트 진행 상태를 알 수 있다. 둘째, 그래프의 기울기를 통해 프로젝트의 수행 속도(Velocity)를 알 수 있다. 기울기가 급하면 그만큼 프로젝트가 빠르게 진행됐다는 것이고 기울기가 적으면 프로젝트 진행 속도가 느리다고 볼 수 있다. 셋째, 점선으로 표시된 예상 업무량과 현재 남은 업무량을 비교하여 현재 진척이 뒤쳐지고 있는지 아닌지를 파악할 수 있다.

그림 2\_일일 진척도 측정 및 번다운 차트 측정



마지막 또 하나의 중요한 산출물은 릴리즈 백로그(Release backlog)이다. 릴리즈 백로그는 제품을 릴리즈 하기 위하여 개발 범위와 일정을 수립하기 위한 목적으로 작성된다. [그림 2]와 같이 릴리즈 백로그는 사용자에게 제품을 언제, 어떤 기능을 담아 제공할 것인지를 결정하게 된다. [표 4]과 같이 릴리즈 백로그에는 릴리즈를 위한 기능 내역과 시간, 스프린트 주기, 릴리즈 일정을 포함하고 있다.

표 4 릴리즈 백로그

| 기능 및 특성(feature) | 내역                        | 시간 | 스프린트 주기     |
|------------------|---------------------------|----|-------------|
| 예약 전송            | 사용자가 예약한 시간에 메시지 자동 전송 수행 | 2h | Sprint 1    |
| 예약 취소            | 사용자의 예약 전송을 취소            | 1h | Sprint 2    |
|                  | 총 남은 시간                   | 3h |             |
|                  | 릴리즈 스프린트                  |    | Sprint 3    |
|                  | 릴리즈 날짜                    |    | 11월 14일 (금) |

## 2.2 소프트웨어 측정 프레임워크

뭔가를 새롭게 개발한다는 것은 창의적인 작업이다. 이 창의적인 작업을 수행해가는 과정에 결과물이 원래 목표를 달성했는지의 여부를 체계적으로 측정하고 평가하는 노력이 필요하다.

소프트웨어의 경우도 마찬가지이다. 소프트웨어를 개발한다는 것은 궁극적으로 조직의 목표 달성을 위하여 제품이나 서비스를 개발한다는 것이다. 따라서 소프트웨어 개발 노력도 목표 달성 여부를 측정하고 평가하는 프레임워크가 필요하다.

실제 소프트웨어 측정 프레임워크는 활용 관점에 따라 다양하게 존재한다. 기존 연구에서는 소프트웨어 측정 프레임워크를 프로세스, 프로덕트 그리고 리소스로 구분하였다. 이에 따르면 프로세스는 비용 효율성 관점에서 투입된 시간과 노력, 오류들을 주로 측정하고, 프로덕트는 신뢰성, 안정성과 같은 제품의 품질 차원에서 규모, 기능성, 재사용성 등을 측정한다. 리소스에서는 생산성 관점에서 투입된 리소스의 비용과 크기 등을 측정한다. 이러한 소프트웨어 측정 프레임워크를 바탕으로 소프트웨어 측정은 목표 지향적인 접근 방법을 취하도록 하는데, 조직의 비즈니스 목표와 우선순위를 수립하고 이를 측정할 수 있는 체계를 갖추는 것이 소프트웨어 측정의 핵심이다. 소프트웨어 측정 프레임워크를 프로젝트 관리(제품과 리소스)와 프로세스 관리와 구분하고 소프트웨어 개발에서 기능 개선, 비용 절감, 적기 출시, 제품 품질 개선을 핵심적인 비즈니스 목표로 보았다. [표 5]에서 비즈니스 목표 달성 측정을 위한 소프트웨어 측정 프레임워크를 보여주고 있다.

표 5\_소프트웨어 측정 프레임워크

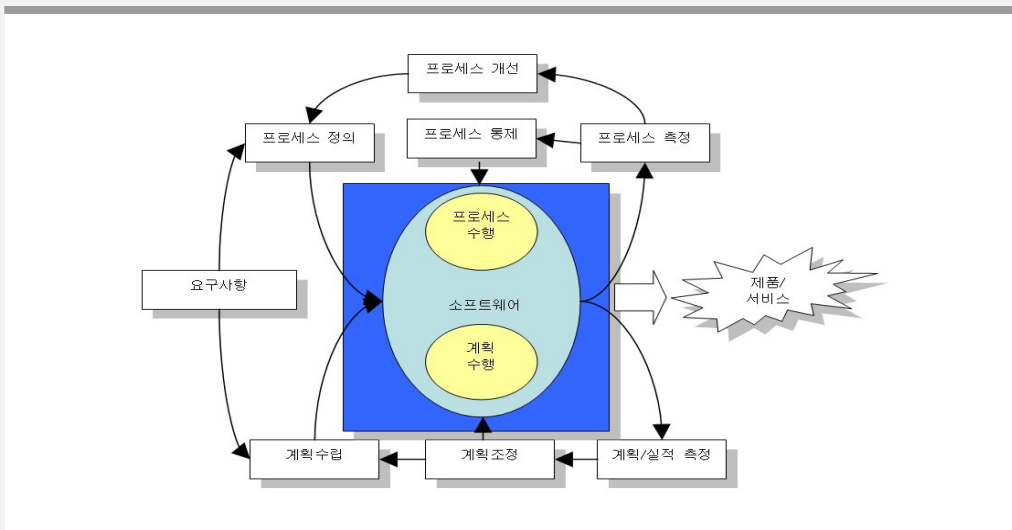
| 비즈니스<br>목표   | 프로젝트 관리<br>측정 항목  | 프로세스 관리<br>측정 항목   | 측정지표  |
|--|---|--|---|
| 기능 개선<br>(Increase<br>function)                      | 요구사항 성장률<br>(Requirement growth)<br>요구사항 안정성<br>(Requirement stability) | 제품 적합도<br>(Product conformance)<br>프로세스 적합도<br>(Process conformance) | 요구사항 수 (Number of<br>Requirements)<br>제품 크기 (Product size)<br>요구사항 변경률 (Rate of<br>requirement change)<br>부적합 비율 (Percentage of<br>nonconforming) |
| 비용 절감<br>(Reduce<br>cost)                            | 예산(Budgets)<br>지출비율(Expenditure rates)                                  | 효율성<br>(Effectiveness)<br>생산성<br>(Productivity)<br>재작업률<br>(Rework)  | 제품 크기 (Product size)<br>제품 복잡도 (Product complexity)<br>투입 공수 (Effort)<br>변경 요구사항 수 (Number of<br>change)  |
| 적기 출시<br>(Reduce<br>time to<br>market)               | 개발 일정<br>(Schedule)<br>진척도<br>(Progress)                                | 생산율<br>(Production rate)   | 투입 시간 (Elapsed time)<br>완성된 작업 수 (# of work<br>completed)   |
| 고객만족도<br>개선<br>(Improve<br>Customer<br>Satisfaction) | 제품 품질<br>(Product quality)  | 프로세스 품질<br>(Process quality)   | 결함밀도 (Defect density)<br>단계별 유입된 결함 수 (Number of<br>defects introduced)<br>결함 발견 효과성 (Effectiveness of<br>defect detection activities)            |

요구사항으로부터 제품이나 서비스로 나오는 소프트웨어 개발 과정은 크게 프로세스와 프로젝트 관리로 나누어 볼 수 있다 즉, 프로세스를 수행하고 측정 및 통제하여 개선을 이루어나가는 프로세스 관리와 전체 계획을 세우고 실적을 계획과 비교하여 조정해 나가는 프로젝트 관리를 통해 소프트웨어 개발이 이루어지게 된다.

프로젝트 관리는 프로젝트의 비용, 일정, 품질, 그리고 기능에 대한 목표 수준을 수립하고 이를 달성하는 것을 목표로 한다. 프로젝트 관리는 목표 달성을 위해 이를 저해하는 위험이나 방해요인들을 찾아내어 제거하거나 감소시키는 것이 목적이다. 프로젝트 관리에서는 계획을 수립하고 계획 대비 실적을 파악하여 계획을 조정해 나가는 통제 과정이 매우 중요하다. 반면에 프로세스 관리의 목적은 원하는 결과를 만들어 낼 수 있도록 프로세스 안정화 및 개선을 통하여 이익을 증대시키는 것이다. 프로세스 관리에서의 핵심은 측정을 통하여 지속적으로 개선해 나가는 것이다.



그림 3\_소프트웨어 개발에서의 프로젝트 관리와 프로세스 관리



## 2.3 스크럼 측정 분석

스크럼의 특성을 분석하기 위해서 소프트웨어 측정 프레임워크를 적용해 볼 필요가 있다. 스크럼을 소프트웨어 측정 프레임워크에 적용함으로써 강점과 약점을 구체적으로 파악할 수 있기 때문이다. 본 보고서에서는 방법론을 정량적으로 분석하고 계량화시키기 보다는 정성적 평가를 위하여 평가 척도를 아래 [표 6]과 같이 도출하였다. 평가 방법은 소프트웨어 측정 프레임워크의 각 측정 지표를 스크럼 산출물과 프로세스가 얼마나 만족시키고 있는지를 평가 척도에 따라 판단하였다.

표 6\_평가 척도

| 구분 | 평가 결과 | 척도                                       |
|----|-------|--|
| O  | 만족    | 측정 지표를 정량적으로 만족시킨다.                      |
| △  | 일부 만족 | 측정 가능한 산출물이 존재하거나 측정 지표를 부분적으로 만족한다.     |
| X  | 불만족   | 측정 가능한 산출물이 존재하지 않거나 측정 지표를 전혀 만족하지 못한다. |

### 2.3.1 프로젝트 관리 측정 평가

먼저 소프트웨어 측정 프레임워크에 따라 프로젝트 관리의 측정 지표에 대하여 스

럼을 적용하였다. 기능 개선을 위한 프로젝트 관리 측정 항목에는 요구사항 성장률과 요구사항 안정성이 있다. 요구사항 성장률은 구현된 요구사항으로 파악할 수 있다. 즉, 요구사항 건수, 제품 크기(예, 코드 라인 수) 등으로 측정할 수 있다. 요구사항 안정성은 프로젝트 기간 동안 요구사항이 얼마나 변경 되었는지를 평가하는 것으로 요구사항 변경이 낮을수록 요구사항 안정성이 높다고 볼 수 있다. 스크럼에서 요구사항 성장률은 제품 백로그에 정의된 요구사항이 스프린트 백로그를 통하여 구현된다. 스크럼에서 스프린트 주기 동안에는 요구사항 변경을 허용하지 않도록 하고 있으나 스프린트 리뷰 이후 사용자의 피드백을 수용하여 제품 백로그에 변경사항을 반영한다. 따라서 스크럼에서 두 지표에 대하여 측정은 가능하지만 정량적으로 관리되지 않는다.

비용절감을 위한 측정 항목은 예산과 지출비율이다. 프로젝트가 예산에 맞게 진행되고 있는지 여부를 확인하기 위한 측정 항목으로 프로젝트 전체 예산과 실제 비용을 측정한다. 스크럼은 비용을 통제하기 위하여 예산을 수립하지 않는다. 스크럼은 투입된 시간이 아니라 남은 작업을 측정한다. 스크럼은 결과 지향적으로 고객에게 전달되는 결과물에 초점을 둔다. 따라서 스크럼에서 예산과 지출에 대한 측정은 이루어지지 않는다.

적기 출시(time to market)를 위한 측정 항목에는 개발 일정과 진척도가 있다. 개발 일정은 프로젝트의 일정 계획 수립을 위하여 기간을 산정하고 작업을 할당하는 것이다. 진척도는 수립된 개발 일정을 기준으로 하여 현재까지 완료된 작업 실적을 측정하는 것이다. 스크럼은 스프린트 단위로 개발 일정을 산정하고 일일 단위로 남은 작업을 검토함으로써 진척도를 측정한다. 또한 번다운 차트는 개발 일정과 진척도를 정량적으로 파악할 수 있도록 그래프로 보여준다. 따라서 스크럼은 개발 일정과 진척도를 정량적으로 측정하고 있지만 전체 개발 일정이 아닌 스프린트 단위로 이루어진다.

고객만족 개선을 위한 측정 항목은 제품 품질로 이를 측정하기 위한 지표로는 결함 밀도가 사용된다. 결함 밀도란 단위 크기당 발견되는 결함 수로 정의될 수 있다. 일반적으로 소프트웨어에서는 1,000 소스 코드당 결함 수를 기준으로 사용한다. 제품 품질 개선은 개발 과정에서 발견되는 최대한 제거함으로써 달성될 수 있다. 스크럼은 스프린트 주기마다 테스트를 수행하여 결함을 발견하고 제거한다. 또한 스프린트 리뷰를 통해 구현된 기능을 데모하는데 이는 일종의 사용자 인수 테스트와 같은 역할을 수행한다. 프로젝트 관리의 측정 항목에 대한 측정 지표를 적용한 결과를 정리하면 아래 표 7과 같다.

표 7\_프로젝트 측정 지표 분석 결과

| 프로젝트 측정 변수       | 측정 가능한 스크럼 산출물     | 충족 수준 |
|------------------|--------------------|-------|
| 제품 성장을<br>제품 안정성 | 제품 백로그<br>스프린트 백로그 | △     |
| 예산, 지출률          | N/A                | X     |
| 개발 일정            | 스프린트 백로그           | O     |
| 진척도              | 번다운 차트             | O     |
| 프로덕트 품질          | 스프린트 데모<br>테스트 결과  | △     |

### 2.3.2 프로세스 관리 측정 평가

위의 프로젝트 관리와 마찬가지로 프로세스 관리의 측정 지표에 대하여 스크럼을 적용하였다. 기능 개선을 위한 프로세스 관리 측정 항목에는 제품 적합도와 프로세스 적합도가 있다. 먼저 제품 적합도는 구현된 기능이 요구사항에 얼마나 부합하는가를 의미하며 측정 지표로는 구현된 요구사항 기능과 제품의 크기로 측정할 수 있다. 스크럼에서는 스프린트 백로그를 통하여 구현된 기능을 파악할 수 있다. 또한 스프린트 리뷰에서 개발된 기능을 데모하여 요구사항이 얼마나 부합하는지를 검토한다. 프로세스 적합도는 개발 공정이 프로세스에 따라 수행된 정도로 파악할 수 있으며 감사(audit)과 같은 활동으로 통하여 수행할 수 있다. 스크럼에서는 감사(audit)과 같은 활동을 수행하지 않는다. 다만 매 스프린트 주기마다 회고를 통해 팀이 정한 작업 룰(working rule)이나 표준을 준수했는지 검토하고 고찰하는 시간을 갖는다. 그러나 측정을 위한 목적은 아니다.

비용 절감을 위한 측정항목으로는 효율성, 생산성, 재작업률을 포함하고 있다. 효율성은 프로세스가 투입 비용 대비 결과의 효율성을 평가하는 것이다. 가령, 결함 발견 후 해결까지 걸리는 시간을 결함 제거 효율성으로 측정할 수 있다. 스크럼은 투입 공수에 대해 측정하지 않기 때문에 얼마나 작업이 효율적으로 수행되었는지 측정하지 않는다. 생산성은 단위 시간 당 만들어지는 결과물로 측정할 수 있다. 스크럼은 스프린트 백로그에 매일 시간 단위로 남은 작업을 측정하여 완료된 작업량을 정량적으로 파악할 수 있다. 재작업률은 개발된 기능에 대하여 결함이나 변경에 따라 다시 수행하는 작업이다. 이는 변경 요구사항 수, 재작업에 투입된 공수 등으로 측정할 수 있다. 스크럼에서는 제품 백로그에 사용자의 요구사항 변경을 반영하고 스프린트 백로그에 결함 수정과

변경 작업 내역을 기록하지만 정량적 측정은 할 수 없다.

적기 출시를 위한 측정항목은 생산율이다. 생산율은 제품의 개발 주기로 제품이 완료되기까지 걸리는 시간으로 파악할 수 있다. 스크럼에서는 스프린트 개발 주기로 기능을 개발하여 릴리즈 백로그로 제품의 출시 시기를 파악하게 된다.

고객만족도 향상을 위한 측정항목은 프로세스 품질이다. 프로세스 품질은 프로세스의 역량(capability)과 관련되어 프로세스의 예측 가능성을 높이고 지속적인 프로세스 개선을 수행하는 것이다. 예를 들어, 각 단계별로 유입되는 결함 수를 통하여 각 단계의 결함 수준을 평가할 수 있다. 스크럼에서는 스프린트 회고를 수행하여 스프린트에서 수행된 활동과 개선될 사항을 검토하고 있으나 프로세스 품질에 대한 측정은 하지 않는다. 프로세스 관리에 속하는 측정 항목에 대하여 스크럼을 적용한 결과를 [표 8]에 정리하였다.

표 8\_프로세스 측정 지표 분석 결과

| 프로세스 측정 변수         | 측정 가능한 스크럼 산출물                   | 충족 수준 |
|--------------------|----------------------------------|-------|
| 제품 적합도<br>프로세스 적합도 | 스프린트 데모<br>테스트 결과<br>스프린트 회고     | △     |
| 효율성                | N/A                              | X     |
| 생산성                | 번다운 차트<br>수행 속도(Velocity)        | O     |
| 재작업률               | 스프린트 백로그                         | △     |
| 생산율                | 스프린트 주기(Sprint cycle)<br>릴리즈 백로그 | △     |
| 프로세스 품질            | 스프린트 회고                          | △     |

### 2.3.3 스크럼 측정 분석 결과

위에서 제시한 소프트웨어 측정 프레임워크를 적용한 스크럼 평가 결과를 살펴보고자 한다. 측정 항목 중에서 만족한 평가를 받은 항목은 개발 일정과 진척도 그리고 생산성이 있다. 이중에서 개발일정과 진척도는 프로젝트 관리 측정 항목이며 생산성은 프로세스 측정 항목이다.

스크럼이 프로젝트 관리 측정 항목에서 개발 일정과 진척도를 만족시킴으로써 프로젝트 일정 관리를 정량적으로 수행한다는 것을 알 수 있다. 일반적으로 프로젝트 기간이 길수록 일정을 정확하게 산정하고 관리하기는 더욱 어렵게 마련이다. 스크럼은 2~4주라는 스프린트 짧은 개발 주기를 반복하여 일정을 보다 정확하게 산정할 수 있다. 또

한 일일 진척도를 시간 단위로 파악하여 현재의 진척 상태를 파악하고 생산성을 측정할 수 있도록 만든다. 스크럼을 도입한 조직에서 보다 빠르게 제품을 개발하고 생산성 향상을 시키고 있는 것은 스크럼이 개발 일정과 진척도를 관리하고 생산성을 정량적으로 측정하는 것과 관련이 있다고 생각된다.

반면에 스크럼은 예산과 지출율, 효율성을 전혀 만족시키지 못하였다. 예산과 지출율은 프로젝트 관리 측정 항목에 속하며 효율성은 프로세스 관리 측정 항목에 해당된다. 계획 중심의 전통적 방법론은 프로젝트 관리에서 비용을 중심에 놓고 있다. 전체 비용을 관리하기 위하여 보다 상세한 계획을 수립하고 계획 대비 실적을 분석함으로써 비용을 통제하였다. 그러나 스크럼은 탑다운(top-down) 방식으로 전체 계획을 수립하고 세부 계획을 구체화하는 것은 현실적이지 못하다고 본다. 또한 측정을 위하여 많은 산출물을 만들어 내는 것 역시 가치가 없는 활동으로 여기고 있다. 즉, 스크럼은 소프트웨어 개발 프로세스를 경험적인 관점으로 바라보고 있는데 이는 소프트웨어 개발은 예측 불가능하고 불규칙적이며 비선형적인 복잡한 프로세스라는 것이다. 이에 따라 스크럼에서는 소프트웨어 개발 프로세스를 관리하기 위하여 자주 반복적으로 검토하고 변화에 유연하게 대처할 수 있는 경험적 프로세스 통제 (empirical process control)을 적용하고 있다. 스크럼에서 일일 회의, 스프린트 데모 및 스프린트 회고는 바로 경험적 프로세스를 위한 통제 도구이다. 지금까지 살펴본 스크럼 측정 분석 결과를 아래 [표 9]에서 정리하였다.

표 9\_스크럼 측정 분석 결과

| 비즈니스 목표  | 프로젝트 측정변수            | 충족 수준 | 프로세스 측정 변수         | 충족 수준 |
|----------|----------------------|-------|--------------------|-------|
| 기능 개선    | 요구사항 성장률<br>요구사항 안정성 | △     | 제품 적합도<br>프로세스 적합도 | △     |
| 비용 절감    | 예산<br>지출률            | X     | 효율성                | X     |
|          |                      |       | 생산성                | O     |
|          |                      |       | 재작업률               | △     |
| 적시 출시    | 개발 일정                | O     | 생산율                | △     |
|          | 진척도                  | O     |                    |       |
| 고객만족도 향상 | 제품 품질                | △     | 프로세스 품질            | △     |

### Ⅲ. 스크럼 개발방법 도입방안

**위**의 평가 결과를 바탕으로 스크럼을 도입하는 조직에서 고려할 사항을 살펴볼 도록 하자. 첫째, 비용 절감의 목적으로 스크럼을 도입하는 조직에서는 전체 예산과 실적을 관리할 수 있는 별도의 방법이나 활동을 고려할 필요가 있다. 제품 백로그로부터 제품을 릴리즈하는데 필요한 범위를 결정하고 필요한 스프린트를 산정해 본다. 첫 스프린트는 개발보다는 전체 개발 일정과 시스템 청사진(blueprint) 기획에 초점을 두고 전체 스프린트 횟수와 투입될 비용을 산정해보는 것이 필요하다. 전체 투입 예산이 산정되면 이를 기준선(baseline)으로 삼고 스프린트 종료 시점에서 전체 예산 대비 실적을 분석해 볼 수 있다. 일정과 비용을 작업 성과에 통합하여 관리하는 획득가치(earned value) 관리는 스크럼의 비용 통제에 적용할 수 있는 도구 중 하나이다. 둘째, 스크럼은 소프트웨어 개발보다는 팀의 개선과 프로젝트 관리에 초점을 맞추고 있다. 따라서 스크럼에서는 소프트웨어 개발 프로세스를 구체적으로 제시하지 않고 있다. 따라서 현재 조직에서 소프트웨어 개발 프로세스가 없는 경우라면 스크럼과 함께 사용될 개발 방법론을 고려하는 것이 바람직할 수 있다. 셋째, 앞에서 언급하였듯이 스크럼은 프로젝트 위주의 방법론으로 조직의 프로세스를 지속적으로 개선하고 제품과 프로세스 품질을 다루는 활동에는 취약하다. 조직의 개발 기술 능력과 성숙도가 낮은 경우, 애자일보다는 계획 중심의 무거운 방법론이 더 적합할 수도 있다. 일반적으로 프로세스 성숙도가 높은 기업들은 새로운 기술이나 방법론 도입을 조직 상황에 적합하게 조정하고 수용할 수 있는 반면 프로세스 성숙도가 낮은 기업은 이를 그대로 적용하여 혼란을 초래할 수 있는 위험이 크다. 따라서 프로세스 성숙도 낮은 조직에서는 스크럼을 소규모나 파일럿을 통하여 먼저 검증해보고 이를 점차 조직에 맞도록 확산해 나가는 것이 필요하다. 또한 프로세스를 지속적으로 개선하기 위하여 핵심 프로세스의 성과 지표를 수립하고 성과를 검토하여 프로세스와 품질을 향상시킬 수 있도록 추진해야 한다.

소프트웨어 개발 환경의 변화에 따라 소프트웨어를 개발하고 구축하기 위한 방법론에도 변화가 요구되고 있다. 계획 중심의 무거운 방법론의 한계점을 극복할 수 있는 대안으로 빠른 변화를 수용하고 생산성을 높일 수 있는 가벼운 프로세스인 애자일이 대두되고 있다. 최근 선진국을 중심으로 애자일을 도입했거나 도입을 고려중인 기업들이 확산되고 있기는 하지만 국내에서 애자일에 대한 연구는 거의 이루어지지 못하고 있는 실

정이다. 국내에서 아직까지 애자일을 도입하려는 조직들에게 구체적이고 실무적인 가이드도 부족한 편이다.

분석 결과, 스크럼은 프로젝트 측정 변수에서는 일정과 진척률을 만족시키고 있지만 예산과 실적은 만족시키지 못하였다. 프로세스 측정 변수에서는 생산성과 생산율을 만족시키고 있는 반면 효율성에 대한 측정 지표는 만족시키지 못하였다. 이는 스크럼의 짧고 반복적인 개발 주기를 통하여 일정과 진척 상태를 정량적으로 관리하는 반면 전체 계획을 수립하고 산출물 측정하거나 관리하지 못하고 있다는 것을 알 수 있다. 아울러 이런 결과를 바탕으로 스크럼을 도입하려는 조직에게 이를 보완하기 위한 구체적인 대안들을 제시하였다.

향후 지속적으로 이를 보완하는 연구를 수행한다면 국내 IT 기업들에게 애자일을 보다 효율적으로 도입하고 측정할 수 있는 구체적 가이드라인을 제공해줄 수 있을 것으로 기대한다.

#### 참고 자료

1. 강규영, “개발환경에서 본 애자일: 오픈마루 개발자의 도입 사례 소개”, 마이크로 소프트웨어, 통권 281호, 2007, pp. 178-183.
2. 강석천, “애자일이란 무엇인가?: 방법론으로 보는 애자일의 의미와 실체”, 마이크로 소프트웨어, 통권 281호, 2007, pp. 172-177.
3. 박지강, “애자일 입문서 스크럼(Scrum): 애자일 방법론 측정의 시발점”, 마이크로 소프트웨어, 통권 289호, 2007, pp. 158-161.
4. 이기호, “애자일(Agile) 방법론을 활용한 제품디자인 프로세스에 관한 연구”, 홍익대학원 석사 학위 논문, 2007.
5. Ambler, Scott W. “Dr Dobb’s 2007 Agile Adoption Survey”, [www.amsbysoft.com/surveys](http://www.amsbysoft.com/surveys), 2007.
6. Abrahamsson, Pekka, Warsta, Juhani, Siponen, Mikko and Ronkainen, Jussi, “New Directions on Agile Methods: A comparative Analysis”, Proceedings of the International Conference on Software Engineering, May 2003.
7. Awad, M.A. “A Comparison between Agile and Traditional Software Development Methodologies”, University of Western Australia, 2005.
8. Beck, Kent and Andres, Cynthia., Extreme Programming Explained, Addison-Wesley Publishers Inc., 2000
9. Boehm, Barry “Value-Based Software Engineering”, ACM SIGSOFT Software Engineering Note, 2003
10. Boehm, Barry and Turner., Richard Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley Publishers Inc., 2003
11. Cockburn, Alistair “Selecting a project’s Methodology”, IEEE Software, July/August, 2000.



12. Cockburn, Alistair and Highsmith, James "Agile Software Development: The People Factor," Computer, pp. 131-133, Nov. 2001.
13. Cockburn, Alistair "Learning From Agile Software Development", The Journal of Defense Software Engineering, 2002.
14. Cohn, Mike and Ford, Doris "Introducing an Agile process to an organization", IEEE Computer, 2003.
15. Cohn, Mike "Agile Estimating and Planning" The Journal of Product Innovation Management, 2006.
16. Fenton, N. and Bush M. E. "Software measurement: a conceptual framework", Journal of Systems and Software, Vol. 12, July 1990.
17. Florac, William. A. and Carleton, Anita, D. "Measuring the Software Process: Statistical Process Control for Software Process Improvement", Addison-Wesley Publishers Inc., 1999.
18. Gilb, Tom "Adding Stakeholder Metrics to Agile Projects", The Journal of Information Technology Management, Vol. 17, No.7, pp. 31~35, July 2004.
19. Hartmann, Deborah and Dymond, Robin "Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value", The proceeding of Agile Conference, July 2006.
20. Highsmith, Jim Agile Project Management: Creating Innovative Products, Addison-Wesley Publishers Inc., 2004.
21. Kitchenham, Barbara Software Metrics: Measurement for Software Process Improvement, Blackwell Publishers Inc., 1996.
22. Law, Amy and Charron, Raylene "Effects of Agile Practices on Social Factors", ACM SIGSOFT Software Engineering Note, July 2005.
23. Layman, Lucas, William, Laurie and Cunningham, Lynn "Exploring Extreme Programming in Context: An Industrial Case Study", Proceedings of the Agile Development Conference, p. 32~41, June 2004.
24. Marcal, Ana, Freitas, Bruno, Soares, Felipe, Furtado, Maria, Maciel, Teresa and Belchior, Arnaldo "Blending Scrum practices and CMMI project management process areas", Innovations in Systems and Software Engineering, Vol 4, No. 1, 2008.
25. Poppendieck, Tom and Poppendieck, Mary Implementing Lean Software Development, Addison-Wesley Publishers Inc., 2006.
26. Rising, Linda and Janoff, Norman "The Scrum Software Development Process for Small Teams", IEEE Software, July 2000.
27. Schwaber, Carey, and Fichera, Richard "Enterprise Agile Adaption in 2007", Forrester Research (<http://www.forrester.com/Research/Workbook/0,9126,45015,00.html>), 2008.
28. Schwaber, Ken "SCRUM Development Process", Proceeding of the 10th Annual ACM Conference on OOPSLA, 1995.
29. Schwaber, Ken "Controlled Chaos: Living on the Edge", American Programmer, April 1996.
30. Schwaber, Ken, Beedle, Mike and Martin, Robert Agile Software Development with SCRUM, Prentice Hall Publishers Inc., 2001.



31. Schwaber, Ken Agile Project Management with Scrum, Microsoft Press, 2004.
  32. Sutherland, Jeff "Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies", Cutter IT Journal, Vol. 14, p. 5~11, 2001.
  33. Subramaniam, Venkat and Hunt, Andy Practices of an Agile Developer, The Pragmatic Bookshelf Publishers Inc., 2005.
  34. Takenchi H. and Nonaka I. "The New New Product Development Game", Harvard Business Review, January-February, 1986
-