

국내외 현황 분석

# SW중소기업을 위한 경량 개발 방법론



과학기술정보통신부  
Ministry of Science and ICT



정보통신산업진흥원  
National IT Industry Promotion Agency

# 경량 개발 방법론

## 국내외 현황

## 발간사

2016년 세계경제포럼(WEF, World Economic Forum)은 전 세계가 당면하게 될 주요 이슈로 4차 산업혁명에 주목했으며, 최근 빅데이터, 클라우드, 인공지능, 사물인터넷 등 융합과 혁신이 빠르게 진행되고 있습니다.

이러한 흐름의 중심에는 소프트웨어가 있습니다. 최근의 소프트웨어는 이전과는 비교할 수 없는 빠른 속도로 변화하고 있습니다. 그러나 전통적 소프트웨어 개발 방법론은 이러한 변화를 빠르게 수용하기 어렵고, 이에 대한 대안으로 경량 개발 방법론이 주목 받고 있습니다.

한 조사에 따르면 해외의 소프트웨어 프로젝트 상당수가 경량 개발 방법론을 활용하고 있으며, 프로젝트 관리 지식 체계를 다루고 있는 PMBOK (Project Management Body of Knowledge)는 2018년 개정판에 경량 개발 방법론을 신규로 포함 시킬 것임을 발표하였습니다.

그러나 기존의 경량 개발 방법론 가이드들은 개념의 이해 위주여서 실제 적용 시 구체적인 실행 방안이 부족하며, 또한 방법론이 추구하는 수평적 조직 문화와 국내 조직 문화와의 이질성 등의 한계로 성공 사례가 적고, 경량 개발 방법론의 대표적인 애자일 및 데브옵스 도입률은 아시아태평양 지역 국가 가운데 가장 낮다는 조사결과도 발표되었습니다.

본 가이드는 기존 경량 개발 방법론의 장점을 모으고 국내 SW 기업의 특성을 반영한 경량 개발 방법론이며, 활용도를 높이기 위해 상세한 설명과, 다양한 사례 및 팁을 수록하였습니다.

아무쪼록 SW 기업들이 본 가이드를 활용하여 비즈니스의 변화에 민첩히 대응하고, 나아가 4차 산업혁명이라는 변화의 중심에 설 수 있기를 바랍니다. 끝으로 본 가이드를 개발하기까지 수고해 주신 관계자 분들께 감사드립니다.

정보통신산업진흥원

원장 윤 종 록

# Contents

|                   |                                   |           |
|-------------------|-----------------------------------|-----------|
| <b>CHAPTER 01</b> | <b>서론</b>                         | <b>05</b> |
| 1.1               | 연구 배경 및 필요성                       | 06        |
| 1.2               | 연구 방법 및 범위                        | 06        |
| <b>CHAPTER 02</b> | <b>애자일 방법론 개요 및 종류 개괄</b>         | <b>07</b> |
| 2.1               | 전통적 소프트웨어 개발 방법의 한계               | 08        |
| 2.2               | 애자일 방법론 개념 및 주요 원리                | 10        |
| 2.3               | 스크럼(Scrum)                        | 14        |
| 2.4               | 익스트림 프로그래밍(eXtreme Programming)   | 17        |
| 2.5               | 린 (Lean) 소프트웨어 개발                 | 19        |
| 2.6               | 칸반(Kanban)                        | 21        |
| <b>CHAPTER 03</b> | <b>전통적 방법론과 애자일 방법론의 장단점</b>      | <b>25</b> |
| 3.1               | 전통적 vs. 애자일 개발 방법론의 차이점           | 26        |
| 3.2               | 전통적 vs. 애자일 개발 방법론의 장단점           | 27        |
| <b>CHAPTER 04</b> | <b>애자일 방법론의 국내외 도입 추세</b>         | <b>29</b> |
| 4.1               | 해외 애자일 방법론 도입 추세                  | 30        |
| 4.2               | 국내 애자일 방법론 도입 추세                  | 37        |
| <b>CHAPTER 05</b> | <b>경량 개발 방법론 도입의 국내 성공, 실패 사례</b> | <b>41</b> |
| 5.1               | 성공 및 실패의 기준                       | 42        |
| 5.2               | 국내 성공 사례                          | 42        |
| 5.3               | 애자일 실패 사례                         | 51        |
| <b>CHAPTER 06</b> | <b>국내 소프트웨어 기업의 애자일 적용 한계점 분석</b> | <b>53</b> |
| 6.1               | 애자일 적용상의 어려움과 한계점                 | 54        |
| 6.2               | 애자일 활성화를 위한 제언                    | 55        |
| <b>부록 1</b>       | <b>경량(애자일)방법론 설문 조사 결과</b>        | <b>57</b> |
| 7.1               | 설문 조사 개요                          | 58        |
| 7.2               | 설문 조사 결과                          | 58        |
| <b>참고문헌</b>       |                                   | <b>65</b> |
| 8                 | 참고 문헌                             | 66        |



CHAPTER

01

서론



|                 |    |
|-----------------|----|
| 1.1 연구 배경 및 필요성 | 06 |
| 1.2 연구 방법 및 범위  | 06 |

# CHAPTER 01 서론

## 1.1 연구 배경 및 필요성

최근의 비즈니스 환경은 빅데이터, 클라우드, 인공지능(AI), 사물인터넷(IoT) 등과 같은 4차 산업 혁명으로 인하여 산업간의 경계가 무너지면서 비즈니스의 불확실성이 한층 높아지고 있다. 이런 환경에서 소프트웨어는 그 중요성이 한층 더해가고 있으며 기업들은 소프트웨어 기술을 기반으로 자신의 하드웨어나 서비스를 더욱 경쟁력 있게 만들려고 최선의 노력을 경주하고 있다. 이런 변화된 비즈니스 환경을 뒷받침하기 위해서 나온 소프트웨어 개발 방법론이 애자일이다. 애자일은 비즈니스 변화에 대한 적응을 강조하다 보니 문서화된 프로세스보다는 사람들간의 수평적 소통과 협력, 자기 주도적인 업무 수행 등 기존의 개발문화와는 다른 접근 방식을 추구하고 있다. 이런 개발문화는 수직적 계층구조와 상명하달에 익숙한 우리의 개발문화와 많은 차이가 나는 것이 사실이다. 그러다 보니 국내에 애자일이 도입된 지 10년이 넘었지만 일부 스타트업을 제외하고는 조직차원에서 성공적으로 정착된 곳이 많지 않다. 인터넷 공간이나 주변에 애자일을 시도했던 개발자들 중의 상당수는 성공사례보다는 실패사례를 얘기하는 경우가 훨씬 많은 현실이다. 2016년 전세계 비즈니스 및 IT 경영진을 대상으로 조사되었던 자료에 따르면 한국기업이 아시아태평양 지역 중에서 가장 낮은 애자일 성숙도를 보였다고 한다. (참고문헌 1)

반면에 해외에서는 소프트웨어 기업을 넘어 일반 제조업이나 서비스업까지 애자일 방식이 확산되고 있으며 단순히 생산성이나 속도를 높이는 차원을 넘어 혁신을 위한 방법론으로 자리매김하고 있다. 본 연구는 국내 중소기업들이 애자일 방법론을 활용하여 빠른 비즈니스 변화에 대응할 수 있는 소프트웨어 개발 기반을 조성하고자 하는 것이 목적이며 이를 위하여 애자일 적용상의 국내 문제점과 한계를 분석하고 이에 대안을 제시하고자 한다.

## 1.2 연구 방법 및 범위

본 연구는 먼저 전통적 소프트웨어 개발의 문제점과 한계를 살펴보고 애자일 방법론 별 특징, 전통적 방법론과 애자일 방법론의 차이점과 장단점을 비교 분석한다. 이를 통하여 애자일 방법론이 효과적인 업무환경을 제시한다. 다음에는 애자일 방법론에 대한 국내외 트렌드와 도입 추세를 살펴보고 성공 및 실패 사례를 분석한다. 마지막으로 국내 기업의 애자일 적용상의 어려움과 한계점을 검토하고 이를 극복할 수 있는 대안을 제시한다.

## CHAPTER 02

### 애자일 방법론 개요 및 종류 개괄



|     |                                 |    |
|-----|---------------------------------|----|
| 2.1 | 전통적 소프트웨어 개발 방법의 한계             | 08 |
| 2.2 | 애자일 방법론 개념 및 주요 원리              | 10 |
| 2.3 | 스크럼(Scrum)                      | 14 |
| 2.4 | 익스트림 프로그래밍(eXtreme Programming) | 17 |
| 2.5 | 린 (Lean) 소프트웨어 개발               | 19 |
| 2.6 | 칸반(Kanban)                      | 21 |

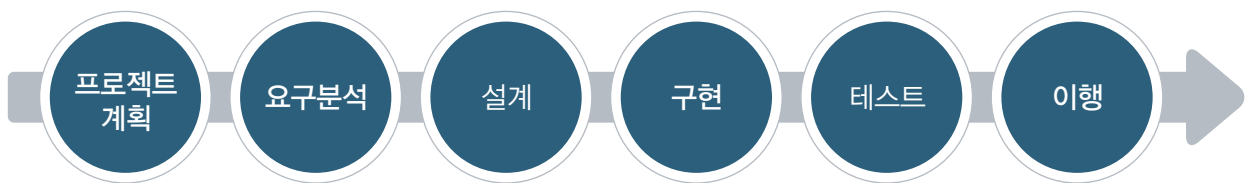
## CHAPTER

## 02

## 애자일 방법론 개요 및 종류 개괄

## 2.1 전통적 소프트웨어 개발 방법의 한계

1960년대부터 소프트웨어 개발의 규모 및 복잡도가 증가함에 따라 구성원들간의 효과적인 의사소통과 개발 공정의 효율성, 품질 확보 등 체계적인 개발의 필요성이 대두되면서 개발방법론들이 등장하기 시작했다. 초기에 나온 구조적 방법론에서부터 정보공학, CBD 개발 방법론들과 CMMI, PMBOK과 같은 프로젝트 관리 방법론들이 나오면서 소프트웨어 개발은 장인적 접근방법에서 보다 공학적인 지식체계로 발전하게 된다. 전통적인 소프트웨어 개발은 이러한 방법론들 바탕 하에 대부분 폭포수(Waterfall) 라이프사이클을 근간으로 개발되고 있다.



〈 폭포수 기반의 전통적인 소프트웨어 개발 형태〉

폭포수 모델은 초기에 사용자 요구사항을 최대한 도출하고 이를 기반으로 계획 및 분석, 설계, 구현이 순차적으로 수행되는 형태로 각 단계마다 리뷰단계가 있어서 전 단계에서 완료되지 못한 것이 있으면 다음 단계로 진행되지 못하는 것이 특징이다. 이러한 단계별 접근은 업무전개가 매우 논리적이면서 개발인력의 효율적인 관리를 도모할 수가 있어서 효과적인 관리 모델로 인식되어왔다. 하지만 이 모델은 요구사항이 명확하고 사업적/기술적 리스크가 크지 않은 제품 개발에 적절한 방식이다. 하지만 현실에서는 그렇지 않은 경우에도 많이 적용되다 보니 여러 가지 문제점들을 야기해온 것이 사실이다. 전통적인 소프트웨어 개발은 이러한 방법론들과 폭포수 모델로 인하여 여러 가지 문제점들이 나타나게 되는 데 주요 내용은 아래와 같다.

첫째, 많은 프로젝트에서 초기 요구 사항을 명확하게 도출하기 어렵다 보니 프로젝트 예산과 일정 예측의 신뢰성이 떨어진다는 점이다. 프로젝트 성격에 따라 초기 요구사항이 명확한 제품 개발도 있지만 비즈니스의 불확실성이 높아진 요즘은 요구사항이 명확하지 않은 경우가 훨씬 많이 발생된다. 폭포수 모델에서는 초기에 요구 사항이 충분히 도출되지 않으면 전체 비용이나 일정 예측이 어렵고 이후 단계가 진행되더라도 많은 요구변경이 발생되기 쉬운 것이 현실이다. 특히 새로운 제품이나 서비스를 만드는 프로젝트는 초기에 개략적인 요구사항만 도출할 수 있을 뿐 구체적인 요구 사항을 도출하기가 매우 어렵다. 이런 상황에서 결정된 초기 범위와 예산과 비용은 불확실성이 높을 수 밖에 없지만 프로젝트 기간 내에 좀처럼 변경되기 어려운 것이 현실이다. 실제 개발과정에서는 요구변경에 따른 업무범위의 변동이 계속 발생하고 있지만 이에 따른 예산, 일정의 변경은 거의 이루어지지 않는다.

둘째, 폭포수 모델의 특성상 프로젝트 후반부에 변경이 많이 발생하다 보니 일정 준수 및 품질 확보가 어렵다는 점이다. 폭포수 모델에서는 제품이 가시화되는 테스트 단계에서 고객의 피드백이 주로 이루어지게 되다 보니 요구 사항의 변경이 이때 많이 발생한다. 물론 분석 단계에서 다양한 분석 기법(유즈케이스 및 프로토타입 등)을 활용하여 고객의 요구 사항을 파악하려고 노력하지만, 이것만으로는 상호간에 최종 모습을 이해하기는 어렵다. 더군다나 새로운 제품인 경우에는 프로젝트 초기에 고객 또한 아이디어가 없기 때문에 어느 정도 완성된 시스템을 보고서야 구체적인 요구 사항이 나오기 마련이다. 전통적 프로젝트 관리에서는 분석단계 이후에 발생하는 추가 요구 및 변경에 대하여 엄격한 변경통제 절차를 통하여 선택적으로 수용할 것을 제시하고 있다. 하지만 한번 결정된 프로젝트 예산과 일정에 대한 조정이 어렵고 고객의 비즈니스 요구 또한 무시할 수 없다 보니 개발팀은 이를 수용할 수 밖에 없는 것이 현실이다. 결국 이러한 요구변경은 대량의 재 작업을 동반하므로 주어진 일정과 품질을 준수하기 어렵게 만든다.

셋째, 개발 프로젝트를 수행하는 과정에서 만들게 되는 산출물이 많다는 점이다. 전통적인 방법론들은 다양한 규모의 프로젝트들을 효과적으로 수행하기 위한 방안으로 만들어지다 보니 많은 관리 절차와 산출물이 존재한다. 이런 산출물들은 개별적으로 타당성이 있고 필요해 보이지만 현장에 적용할 경우에는 효과적인 테일러링(tailoring)이 필요하다. 그렇지 않으면 개발팀에게 불필요한 업무로드로 작용하기 쉽다. 또한 폭포수 모델의 특성상 해당 단계를 완료하지 못하면 다음 단계로 진행되지 못하다 보니 단계별 산출물을 모두 작성해야 한다. 하지만 개발과정에서 설계단계와 구현은 엄격히 분리되기 어렵다 보니 구현이 진행됨에 따라 설계변경이 많이 발생하는 것이 현실이다. 이러한 산출물 변경이나 불충분한 테일러링은 그만큼 개발팀으로 하여금 산출물 작업에 불필요한 노력을 기울이게 만드는 요소가 되고 있다.

넷째, 고객과 개발자간의 의사소통 부족으로 인하여 의도하지 않은 요구변경이 자주 발생된다는 점이다. 폭포수 모델에서는 분석, 설계, 구현, 테스트가 단계별로 진행되다 보니 각 단계별로 적절한 인력을 배치할 수가 있어서 인력을 효율적으로 활용할 수 있는 측면이 있다. 하지만 이러한 인력 운용 방식은 산출물 중심으로 정보전달을 하게 만들어 구현단계에 참여한 개발자로 하여금 사용자와의 직접적인 소통을 저해하는 요소로 작용한다. 이런 상황에서 개발자는 오직 분석, 설계 산출물을 기반으로 작업할 수 밖에 없다. 사용자의 생각이 산출물에 충분히 담긴다면 문제가 없을 수도 있지만 불행히도 몇 사람을 통해서 전달된 정보는 상당 부분 왜곡될 수 밖에 없다.(참고 문헌 2) 사용자의 생각은 분석가와 설계자를 거치면서 상당부분 변질되어 산출물에 기록되기 쉬우며 개발자는 이런 산출물을 보고 코딩을 하다 보니 사용자의 본래 의도와는 많은 차이가 발생하는 것이다.

다섯째, 리더 중심의 명령과 통제 방식으로 인하여 팀원들간에 소통과 협력이 원활하게 일어나지 않는다는 점이다. 전통적인 프로젝트에서는 소수의 개발리더들이 업무를 계획하고, 팀원에게 업무를 배분하는 역할을 수행하는 수직적 의사전달 체계로 일이 진행된다. 팀원은 그저 개발리더에게 할당 받은 업무를 의무적으로 수행할 뿐이다. 그러다 보니 팀원 간에 업무적인 의사소통이 잘 일어나지 않게 되고 상대방을 도와주고 싶어도 무슨 일을 하고 있는지 잘 모르다 보니 상호 협력이 제한적일 수 밖에 없다. 프로젝트 팀원들끼리 얘기하면 간단히 풀릴 문제가 관리자 레벨로 올라가다 보니 문제 해결이 늦어지는 경우도 많이 발생된다. 이러한 수직적 업무관리 방식과 팀원들 간에 소통 부재는 창의적인 제품 개발을 위한 환경과는 거리가 멀다.

## 2.2 애자일 방법론 개념 및 주요 원리

1990년대 중반 인터넷이 대중화되기 시작하면서 산업 구조 및 비즈니스 환경의 급격한 변화가 나타나기 시작했다. 인터넷 기술의 발전으로 구글과 아마존 같은 새로운 비즈니스 모델을 가진 기업들이 출현하면서 기업의 비즈니스 환경은 예측하기 어려울 정도로 빠르게 바뀌게 된다. 특히 인터넷이나 모바일 기반의 소프트웨어 제품들은 경쟁이 치열하다 보니 과거보다 신속히 개발되어 시장에 출시할 필요성이 점점 높아지게 되었다. 전통적인 방법론들은 훌륭한 제품은 좋은 프로세스를 통해서 나온다는 가정 하에 다소 규범적인 프로세스를 통해 제품을 개발할 것을 권장해왔다. 하지만 현실에서는 이러한 방법들이 변화된 비즈니스 환경에 적절하지 않아 많은 사람들이 새로운 소프트웨어 개발 방법을 모색하게 된다. 애자일 방법론은 앞 절에서 언급했던 전통적 소프트웨어 개발에서 나타나는 문제점들을 해소하기 위한 방안으로 시작되었다.

짐 하이스미스(Jim Highsmith), 켄 슈와버(Ken Schwaber)와 같은 사람들은 요구사항의 불확실성을 관리하면서 고객에게 가치 있는 기능을 우선적으로 개발하는 프로젝트 관리 방법론을 주장하였으며 켄트백(Kent Beck) 역시 사람들 간의 빈번한 의사소통과 산출물을 최소화하는 개발 방법론을 발표한다. 다음은 1990년대 중반부터 지금까지 발표된 주요 애자일 개발 방법론들이다.

- DSDM(Dynamic Systems Development Methods), 1994
- Scrum, 1995
- Crystal Clear, 1996
- XP(eXtreme Programming), 1996
- FDD(Feature Driven Development), 1997
- ASD(Adaptive Software Development), 2000
- Lean SW Development, 2003
- Kanban, 2006

이러한 방법론들은 기존 방법론들이 가지고 있던 다소 무겁고(Heavyweight) 규범적인 프로세스의 문제점들을 극복하면서 소프트웨어를 좀더 가볍고(Lightweight) 효율적으로 개발하는 일련의 방법들을 제시하였다. 전통적인 개발 방법은 프로젝트 초기에 요구사항을 기반으로 완벽한 계획과 분석/설계/구현이 순차적으로 이루어지는 폭포수 형태의 개발인데 반하여 애자일은 초기 요구사항의 불확실성을 가정하고 프로젝트 초기에 개략적인 릴리즈 계획만 세우고 주기적으로 이터레이션 계획을 수립하면서 점진적/반복적 개발을 추구한다.



〈일반적인 애자일 개발 형태〉

2001년에는 일련의 애자일 방법론을 발표한 애자일 실천가들이 한 장소에 모여 이러한 방법론들이 가지고 있는 공통적인 개발 철학을 정리하게 되는 데 이것이 유명한 애자일 개발 선언문(agile development manifesto)이다. 이 선언문은 다양한 애자일 방법론들이 가지고 있는 근본 철학을 기술한 문서로 기존의 소프트웨어 개발 방식의 패러다임을 바꾸는 혁신적인 사상을 담고 있다.

## 애자일 개발 선언문

우리는 SW를 개발하는 더 나은 방법을 찾는 과정에서 다음과 같이 생각하게 되었다.

1. 프로세스와 도구보다는 개인과 개인 간의 상호 작용에 더 큰 가치를 둔다.  
(Individuals and interactions over processes and tools)
2. 포괄적 문서화보다는 동작하는 소프트웨어에 더 큰 가치를 둔다.  
(Working software over comprehensive documentation)
3. 계약 협상보다는 고객과의 협력에 더 큰 가치를 둔다.  
(Customer collaboration over contract negotiation)
4. 계획을 따르기보다는 변화에 대응하는 것에 더 큰 가치를 둔다.  
(Responding to change over following a plan)

### 2.2.1 프로세스와 도구보다는 개인과 개인 간의 상호 작용에 더 큰 가치를 둔다.

프로세스와 도구, 사람은 제품개발을 성공적으로 수행하기 위한 세가지 요소이다. 전통적인 방법론들은 대부분 프로세스 개선에 초점을 맞추는 경향이 있다. 반면에 애자일은 아무리 좋은 프로세스와 도구가 있다 하더라도 뛰어난 개인들과 이들간의 협력이 없다면 훌륭한 제품을 만들기 어렵다고 생각한다. 즉, 훌륭한 제품은 도구나 프로세스에서 나오는 것이 아니라 사람에게서 나온다는 것이다. 예를 들어 좋은 프로세스와 도구가 있지만 사람들 간의 소통과 협력이 낮은 팀보다 프로세스와 도구는 부족하지만 열정 있는 사람들간의 소통과 협력이 높은 팀이 압도적인 성과를 내는 경향이 있다.

### 2.2.2 포괄적 문서화보다는 동작하는 소프트웨어에 더 큰 가치를 둔다.

규모가 큰 프로젝트에서 문서화된 개발 산출물은 어느 정도 필요하다. 이런 문서가 없으면 나중에 재 작업을 발생시키는 요소가 될 수 있기 때문이다. 하지만 개발 산출물은 어디까지나 최종 시스템을 잘 만들기 위해서 필요한 것이지 그 자체가 목적이 되어서는 안 된다. 왜냐하면 소프트웨어 개발에서 고객에게 가치가 있는 것은 동작하는 시스템이지 문서화된 산출물이 아니기 때문이다. 산출물은 프로젝트 팀에게 필요한 만큼만 작성되어야지 여기에 시간을 많이 빼앗겨서는 안 된다. 그보다는 고객에게 시스템을 빨리 보여주고 피드백을 받는 것이 좀더 가치 있는 제품을 개발하는 방법이다.

### 2.2.3 계약 협상보다는 고객과의 협력에 더 큰 가치를 둔다.

프로젝트 후반부에서 발생하는 요구사항 변경은 고객 관점에서는 필요한 것일 수 있지만 개발팀에게는 부담스런 업무로 작용한다. 제한된 일정과 예산 하에서 이런 요구 변경은 개발팀으로 하여금 고객과의 협력보다는 계약 협상에 더욱 치중하게 만드는 요소로 작용하기 때문이다. 하지만 요구사항의 불확실성을 처음부터 인정하고 변경되는 요구사항들은 기존 요구사항과의 우선순위화 과정을 통하여 프로젝트에 주기적으로 반영해나가면 고객과 협력하기가 수월하다. 우선순위가 떨어지는 요구사항들은 주어진 일정과 예산 내에서 최대한 반영한다.

### 2.2.4 계획을 따르기보다는 변화에 대응하는 것에 더 큰 가치를 둔다.

새로운 제품이나 서비스를 만드는 경우 프로젝트 초기에 요구사항을 구체적으로 정의하기도 어렵지만 설사 정의하였다 하더라도 비즈니스 환경 변화에 따라 업무 진행 중에 변경되기 쉽다. 이런 상황에서 초기에 설정된 업무 범위와 비용, 일정은 하나의 기준일뿐이지 절대적인 목표가 되어서는 안 된다. 무조건 초기 계획을 준수하는 것보다는 비즈니스 환경 변화에 적절히 대응하면서 실질적인 고객 가치를 높일 수 있도록 목표를 갱신해 가는 것이 효과적이다.

애자일 개발 선언문 참가자들은 이런 선언문을 실질적으로 지원하기 위하여 아래와 같은 12가지 핵심 원리를 제시한다. 이 원리들은 애자일 방법론들의 다양한 프랙티스들과 밀접하게 연결되어 있으며 조직에서 애자일을 적용할 때는 이러한 원리를 따르면서 방법을 유연하게 가져가는 것이 필요하다.

- 1) 소프트웨어 개발을 수행하는 최우선 목표는 빠르고 지속적으로 가치 있는 소프트웨어를 전달함으로써 고객을 만족시키는 것이다.

(Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.)

- 2) 애자일 프로세스는 고객의 경쟁력을 위해서라면 비록 개발 후반일지라도 요구 사항의 변경을 환영한다.

(Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.)

- 3) 가능하면 2주에서 2개월까지 짧은 간격으로 작동하는 소프트웨어를 자주 전달하라.

(Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.)

- 4) 요구 사항을 내는 고객과 개발자는 프로젝트 전체에 걸쳐 날마다 함께 일해야 한다.

(Business people and developers must work together daily throughout the project.)

- 5) 동기가 부여된 개인들 중심으로 프로젝트를 구성하며 그들이 필요로 하는 환경과 지원을 주고 그들이 일을 끝내리라고 신뢰하라.

(Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.)

6) 개발팀 내부에서 정보를 전하는 가장 효율적이고 효과적인 방법은 면대면 대화이다.

(The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.)

7) 작동하는 소프트웨어가 진척의 주된 척도다.

(Working software is the primary measure of progress.)

8) 애자일 프로세스는 지속 가능한 개발을 장려한다. 경영자, 개발자, 고객은 일정한 속도가 계속 유지할 수 있어야 한다.

(Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.)

9) 기술적 탁월성과 좋은 설계에 대한 지속적 관심이 민첩성을 높인다.

(Continuous attention to technical excellence and good design enhances agility.)

10) 단순함이--하지 않는 일의 양을 최대화하는 기술이--필수적이다.

(Simplicity--the art of maximizing the amount of work not done--is essential)

11) 최고의 아키텍처, 요구사항, 설계는 자기 조직화된 팀에서 창발한다.

(The best architectures, requirements, and designs emerge from self-organizing teams.)

12) 정기적으로 어떤 방법이 팀에 더 효과적일지 숙고하고, 이에 따라 팀의 행동을 조율하고 조정한다.

(At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.)

## 2.3 스크럼(Scrum)

### 2.3.1 스크럼 개요

스크럼은 1990년대 초반부터 켄 슈와버와 제프 서덜랜드에 의해서 만들어진 애자일 프로젝트 관리 방법론중의 하나이며 현재는 전 세계적으로 가장 많이 사용되고 있는 애자일 방법론이다. 스크럼은 경험적(empirical) 프로세스 관리 이론에 기반하고 있으며 요구사항의 불확실성을 관리하기 위하여 점진적/반복적 접근 방법을 사용한다.

### 2.3.2 스크럼 프로세스의 역할

스크럼 팀의 역할은 모두 수평적 관계로 누구를 지시하거나 통제할 수는 없다.

#### • 제품책임자

- 제품의 가치와 개발팀의 결과물을 최상으로 만들 책임을 가짐
- 사용자를 대표하여 해당 업무 요구사항을 책임지고 의사결정
- 주기적으로 업무 미팅에 참여하여 개발팀에게 피드백
- 업무의 우선순위 결정 및 테스트 참여

#### • 스크럼마스터

- 구성원들이 스크럼을 제대로 수행하도록 지원
- 제품책임자와 개발팀의 가교역할로써 서번트 리더를 지향
- 개발팀이 자기 조직화 할 수 있도록 개발팀 코칭
- 타 부서와의 관계 원활화와 업무수행 장애요인을 제거

#### • 개발팀

- 보통 5~9명으로 구성되며 제품 개발에 책임이 있음
- 공동책임 및 상호협력을 통한 출시 가능한 제품을 개발
- 제품을 만드는 데 필요한 모든 기술이 있는 기능 혼합(Cross-functional) 팀으로 구성됨

### 2.3.3 스크럼 원리

경험적 프로세스 관리를 위해서는 다음 3가지가 필요함

#### •투명성 (Transparency)

- 업무 진행 상황과 같이 프로세스의 중요한 부분들은 이해 관계자들이 이해할 수 있도록 가시화되어야 되어야 한다.

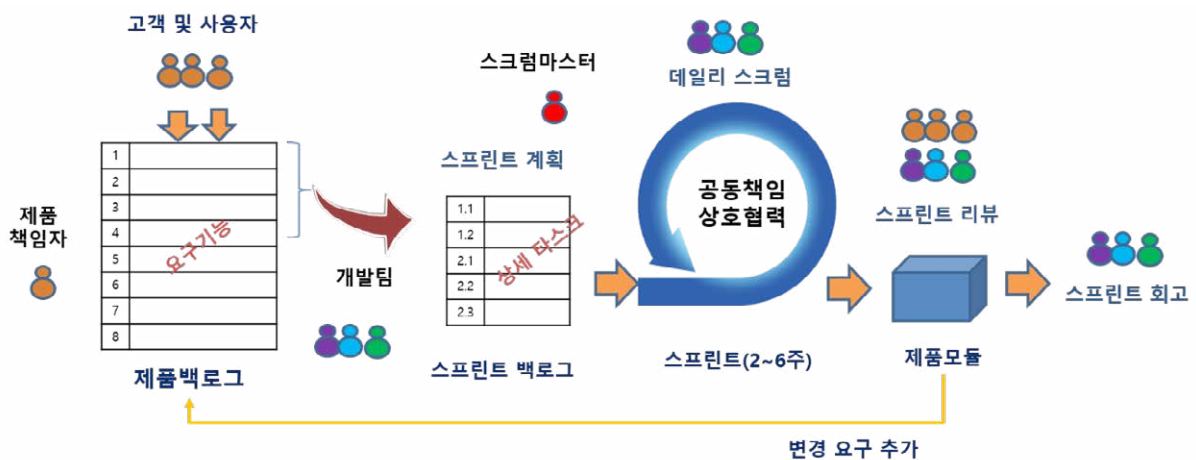
#### •검토 (Inspection)

- 스크럼 사용자들은 주기적으로 스크럼 산출물들을 검토하여 변경 사항들을 찾아내야 한다. 산출물을 검토하는 것은 작업이 진행되는 시점에 숙련된 검토자에 의해 성실하게 이루어 질 때에 가장 효과적이다.

#### •적응 (Adaptation)

- 사용되는 프로세스가 허용 기준 한도를 넘어 결과 제품이 승인되지 않을 것을 검토자가 확인했다면, 그 프로세스나 진행 중인 작업들을 조정하는 과정이 필요하다.

### 2.3.4 스크럼의 주요 활동



〈스크럼 개발 프로세스〉

#### •제품백로그

- 제품개발에 필요한 모든 업무를 우선순위화한 목록으로 기능 및 비 기능 요구사항이나 기술적 업무 등이 모두 포함됨
- 제품백로그에 정의된 기능을 사용자 스토리(User Story)라고 하며 제품책임자가 책임을 가지고 주기적으로 업데이트함

#### •스프린트 백로그

- 개발팀에서 하나의 스프린트 동안 수행할 작업목록으로 제품백로그의 사용자 스토리를 완료하기 위한 상세 태스크들이 포함됨

**• 스프린트 계획**

- 개발팀은 제품백로그의 사용자 스토리를 이해하고 팀에서 수행할 만큼의 사용자 스토리를 가져와서 상세 태스크로 분할함
- 개발팀원은 자발적으로 수행할 상세 태스크를 선택함

**• 데일리 스크럼(Daily Scrum)**

- 매일 같이 15~20분씩 모든 개발팀원들이 한 장소에 모여서 업무 조율 및 협력 방안을 논의하는 회의
- 참석자들은 각자 어제 한 일, 오늘 할 일, 업무 수행 시 장애요인을 얘기함

**• 스프린트 리뷰**

- 스프린트 동안에 수행된 업무를 제품책임자 및 사용자들에게 데모하고 설명함으로써 제품에 대한 폭넓은 이해와 피드백을 받는 회의

**• 스프린트 회고**

- 스프린트 종료 후 개발팀들이 모두 모여서 개발기간 동안 수행했던 활동들을 돌아보고 비효율적인 프로세스를 개선하는 활동
- 스프린트 동안 잘 수행한 사항과 잘 수행되지 못한 사항, 다음 스프린트에서 개선할 사항들을 팀원들과 함께 도출하고 개선 방안을 모색하는 회의

## 2.4 익스트림 프로그래밍(eXtreme Programming)

### 2.4.1 XP 개요

XP는 eXtreme Programming의 약어로 켄트 벡을 중심으로 한 몇몇 엔지니어들에 의해서 만들어진 애자일 방법론 중의 하나이다. 처음 시작은 불확실한 요구사항을 개발하는 10명 이하의 소규모 개발팀을 위한 경량 방법론으로 소개되었으나 지금은 스크럼과 함께 다양한 규모의 프로젝트에서 활용되고 있다. 스크럼이 주로 프로젝트 관리 방법에 초점을 맞추고 있다면 XP는 엔지니어링 기법 등이 많이 포함되어 있어 서로가 잘 맞다고 할 수 있다. XP는 페어 프로그래밍이나 테스트 주도 개발 같은 기존의 개발 방식과는 판이하게 다른 부분이 있다 보니 다른 애자일 방법론에 비해서 논쟁이 많이 되어온 것이 사실이다. 아직도 아키텍처나 설계가 중요시되는 프로젝트에서는 수용되기 어려운 측면이 있다.

### 2.4.2 XP의 핵심 가치

#### • 의사소통(Communication)

- 소프트웨어 개발 과정에서 가장 중요한 것 중의 하나가 의사소통이다. 일부 사람들은 의사소통보다는 코딩에 집중하는 것을 선호할 수 있지만 개발과정에서 문제가 생겼을 때 누군가는 이미 그 문제의 해결책을 알고 있는 경우가 많다. 짧게 소통함으로써 보다 많은 시간을 절약할 수가 있는 것이다. 소통에는 개발자 간 소통과 개발자와 사용자 간의 소통, 개발자와 관리자 간의 소통 등이 모두 필요하다.

#### • 단순성(Simplicity)

- 소프트웨어 개발은 현실에서 보다 어려운 문제를 해결할 것을 요구 받고 있다. 그러다 보니 소프트웨어 역시 점점 복잡해지고 있다. 복잡한 소프트웨어는 만들기도 어렵지만 유지보수 또한 쉽지 않다. 현 시점에서 꼭 필요한 것만 수행하는 것이 보다 간결한 설계와 불필요한 노력을 예방한다.

#### • 피드백(Feedback)

- 소프트웨어 개발이 시작되면 가급적 빠른 시일 내에 고객에게 피드백을 받는 것이 필요하다. 고객과의 빠른 피드백은 사업목적에 이해하고 궁극적으로 올바른 요구사항을 찾는 방법이다.

#### • 용기(Courage)

- 소프트웨어 개발은 예측하기 어려운 문제들을 해결하는 과정이므로 모른다는 것을 창피해하지 않는 용기, 새로운 방법에 도전하는 용기, 나만 생각하지 않고 팀을 위하여 헌신하는 용기 등이 필요하다.

#### • 존중(Respect)

- 탁월한 팀을 만들기 위해서 반드시 필요한 것이 있다면 상호간의 존중이다. 구성원들이 원활히 협력하기 위해서는 모두가 중요하고 존중 받아야 한다는 생각이 있어야 한다.

### 2.4.3 XP의 주요 활동

- 개발 구성원 전체가 모여서 공동의 작업 공간에서 일하면 의사소통과 피드백이 빠르게 이루어진다.
- 제품 개발에 필요한 스킬을 가진 사람들을 한 팀에 포함시켜서 일한다.

- 제품 개발의 진행상황을 한 눈에 볼 수 있도록 개발팀 벽면에 개발 현황에 대한 정보를 부착한다. (Informative Workspace)
- 소프트웨어 개발은 집중을 요하므로 하루 8시간 이상 일하는 것은 지속적인 활력을 떨어뜨린다.
- 페어 프로그래밍(Pair programming) 은 두 사람이 하나의 컴퓨터와 키보드를 공유하면서 일하는 기법으로 상호간에 코드 리뷰를 함으로써 보다 좋은 품질을 얻을 수 있다.
- 고객에게 가치를 줄 수 있는 최소한의 기능(user story) 단위로 계획한다.
- 일주일 단위 및 분기별로 계획한다.
- 10분안에 전체 시스템을 빌드하고 테스트 할 수 있는 지속적인 통합 환경을 구축한다. (Continuous Integration)
- 코딩을 하기 전에 자동화된 테스트 케이스를 먼저 작성한다. (Test First programming)
- 시스템의 설계에 매일 조금씩 투자한다. (Incremental Design)

## 2.5 린 (Lean) 소프트웨어 개발

### 2.5.1 린 개요

린(lean)은 세계적으로 유명한 도요타 자동차의 효율적인 생산방식에 근거하여 만들어진 하나의 경영기법이다. 이미 자동차 같은 제조 산업분야에는 많이 알려져 있는 개념이나 소프트웨어 개발에 적용된 것은 톰 포펜딕과 메리 포펜딕 부부에 의해서 널리 알려졌다. '린'이란 의미는 군살 없이 일체의 낭비가 최소화된 생산방식을 의미한다. 린 소프트웨어 개발의 핵심은 개발 프로세스에 있는 일체의 낭비를 찾아서 제거하고 어떻게 고객에게 빠르게 가치를 제공할 수 있을 것인가에 대한 사고방식이라고 할 수 있다. 여기서 낭비란 고객에게 가치를 제공하지 않는 활동을 말하며 생산성과 품질을 저해하는 요소라고 할 수 있다.

스크럼과 XP, 린은 요구사항의 변화를 수용하고 고객에게 제품을 빠르게 전달하는 등 여러 공통점이 있는 반면에 차이점도 있다. 스크럼이 구성원들과의 소통과 협력에 중점을 두었다면 린은 전체 프로세스 상의 낭비를 제거함으로써 고객 가치를 높이는 것에 우선순위를 두고 있으며 경영시스템까지 폭넓은 시각을 제공한다. 현실에서는 스크럼과 XP, 린을 모두 활용하여 소프트웨어 개발에 적용하면 더 큰 성과를 얻을 수 있는 것으로 알려져 있다. 린 소프트웨어 개발은 아래 원칙에 따라 개발한다.(참고문헌 2)

### 2.5.2 린 원칙

- 낭비를 제거하라 (Eliminate Waste)

소프트웨어 개발 과정에는 보이지 않는 낭비요소들이 많으며 가치흐름도(Value Stream)를 통하여 낭비가 발생하는 부분을 지속적으로 제거해야 한다.

- 과잉기능

소프트웨어 전체 기능 중 80%의 가치를 제공하는 기능은 20% 정도이며 나머지는 과잉기능이므로 20%를 먼저 개발하고 나머지는 점진적으로 개발

- 미완성 작업

부분적으로 완료된 기능은 고객에게 아무런 가치를 제공하지 않음

- 이동

다른 사람과의 커뮤니케이션을 위한 이동 시간이나 산출물에 의한 정보전달

- 업무전환

한 사람이 동시에 여러 프로젝트에 투입하게 될 경우 업무전환 시간이 자주 발생됨

- 품질을 내재화 하라

결함을 기록하는 데 집중하지 말고 결함이 처음부터 생기지 않도록 해야 한다.

- 테스트 주도 개발을 통하여 코드의 실수를 방지한다.
- 테스터의 직무는 결함을 발견하는 것이 아니라 예방하는 것이다.
- 빅뱅통합 대신 지속적 통합을 사용한다.

### • 지식을 창출하라

- 표준은 무조건 따라야 할 프로세스가 아니라 도전 받고 개선되어야 한다.
- 예측 가능한 결과보다는 미래의 변화에 빠르게 대응하는 것이 중요하다.
- 다양한 실험을 통하여 최선의 대안을 찾는 과학적 방법을 사용한다.

### • 확정을 늦춰라

- 완벽한 명세서를 가지고 개발을 시작하는 것이 좋은 생각이라는 것을 버린다.
- 시스템 아키텍처는 언제 어떤 기능이 추가되더라도 그것을 수용할 수 있도록 의존성이 없어야 한다.
- 코드를 실험으로 생각하고 변화를 수용할 수 있게 작성한다.
- 돌이킬 수 없는 결정을 내리기 전까지 가능한 한 많이 학습하고 마지막 순간에 한다.

### • 빨리 인도하라.

- 품질을 내재화하고 낭비를 줄이면 고품질로 빠르게 인도할 수 있다.
- 진행중인 작업의 크기와 작업량을 최소화하면 사이클타임을 줄일 수 있다.
- 개발자의 가동률이 높아지면 오히려 정체를 일으켜 일이 지연된다.
- 일의 양을 할 수 있는 만큼으로 제한하고 당김(Pull)시스템을 사용한다.

### • 사람을 존중하라.

- 자기 주도적으로 일하는 사람이 최고의 지속가능한 우위를 제공한다.
- 도요타의 진정한 혁신은 평범한 직원으로부터 지성을 끌어내어 활용하는 능력이다.
- 팀원들에게 신뢰, 칭찬, 자부심, 책임감을 주어야 한다.
- 마감기한을 맞추라고 사람들에게 압력을 행사하기 보다는 시스템을 검토한다.
- 팀은 공동의 책임을 가지고 상호 협력해서 일해야 한다.
- 팀 룸에 시각적 관리 방법을 사용하여 작업 현황판과 대시보드 등을 관리한다.
- 사람들의 협력을 떨어뜨리는 인센티브를 줄이고 상대평가 체계를 없앤다.
- 협력업체를 존중하고 신뢰를 바탕으로 일한다.
- 범위기반의 고정가 계약 방식은 서로 간의 신뢰를 떨어뜨리며 목표 비용과 일정을 가지고 서로가 협력적으로 일하는 것이 최상의 결과를 얻는다.

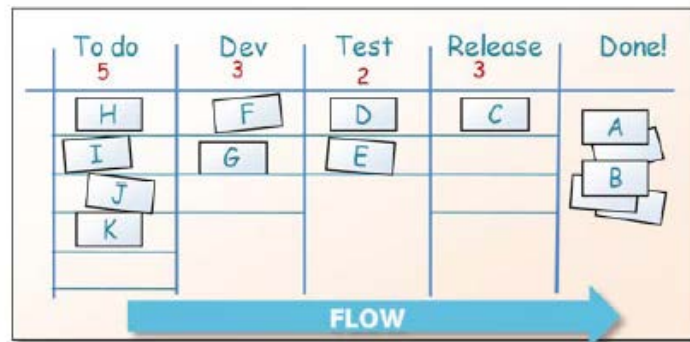
### • 전체를 최적화하라.

- 고객요구에서 배포까지의 전체 흐름을 관리해야 한다.
- 부분최적화가 전체 시스템 성능을 떨어뜨린다.
- 부분적으로 측정하지 말고 전체 시스템을 측정하라.
- 소프트웨어만이 아닌 완전한 제품을 개발한다.
- 완전한 제품은 기능혼합(Cross-functional)팀에 의해 만들어진다.

## 2.6 칸반(Kanban)

### 2.6.1 칸반 개요

칸반이란 전문 서비스 및 소프트웨어 개발 등과 같이 지식 업무를 제공하는 서비스를 효율적으로 개선하는 방법으로 린 원리 중의 하나인 당김 시스템과 적시생산(Just in Time)에 근거를 두고 있다. 칸반 기법은 원래 제조 공정에서 재고를 최소화 하는 데 사용되던 기법이었으나 2000년대 초 데이비드 앤더슨이 소프트웨어 개발에 적용하면서 널리 퍼지게 되었다. 칸반이란 하나의 신호 체계로 칸반보드에 진행 중 업무(Work In Process)에 제한을 둬으로써 고객에게 가는 가치를 높일 수가 있다. 아래 그림의 상단 숫자는 WIP에 대한 제한을 설정한다. 즉 'To Do' 열에는 5개의 업무만 들어올 수 있다는 의미이다. 이 중에서 'Dev'로 1개가 이동하면 그때서야 새로운 업무가 'To Do' 열로 들어올 수가 있다.



칸반보드 샘플

칸반은 스크럼과 달리 기존 프로세스와 직책 등을 그대로 인정하고 현재의 프로세스를 가시화하는 것을 기본으로 하고 있다. 그래서 새로운 변화에 대한 조직의 저항이 다소 적다는 장점이 있다. 칸반은 프로젝트 보다는 소프트웨어 유지보수나 운영, 일반적인 지식 서비스를 제공하는 곳에서 많이 활용되고 있는 편이다.

### 2.6.2 칸반의 핵심 가치와 기본 원칙

칸반의 근간에 깔려있는 기본적인 철학은 조직에 기여하는 모든 사람에 대한 존중이라고 할 수 있으며 다음과 같은 가치를 추구하고 있다.(참고문헌 3)

- 업무에 대한 투명성(Transparency)
- 업무량에 대한 균형(Balance)
- 구성원들간의 협업 (Collaboration)
- 고객 중심 (Customer Focus)
- 흐름(Flow) 중시
- 개인 및 조직에 대한 이해(Understanding)
- 상호간의 합의 (Agreement)
- 사람에 대한 존중 (Respect)

칸반은 다음과 같은 기본 원칙을 가지고 있다.

- 현재 실행되고 있는 프로세스를 그대로 파악하고 기존의 역할, 책임, 직함을 존중한다.
- 개선은 점진적 변화를 통해 추구한다는 데에 합의한다.
- 각 개인부터 고위경영진까지, 모든 계층의 리더십 행동을 장려한다.
- 고객의 요구와 기대를 이해하고 거기에 집중한다.
- 업무를 관리하여, 사람들이 스스로 조직화 할 수 있도록 한다.
- 고객 및 비즈니스 성과를 개선하기 위해 정책을 발전시킨다.

### 2.6.3 칸반의 주요 활동

- 업무 흐름을 시각화 한다.

고객요구에서 배포까지의 수행되는 업무 항목을 먼저 이해하고 칸반 보드를 사용하여 업무를 시각화 한다. 칸반보드의 열은 워크프로우의 단계를 의미하며 행은 서비스 유형(신규기능, 버그 수정, 단순요청 등)을 나타낸다.

- 진행 중 업무(Work In Process)를 제한한다.

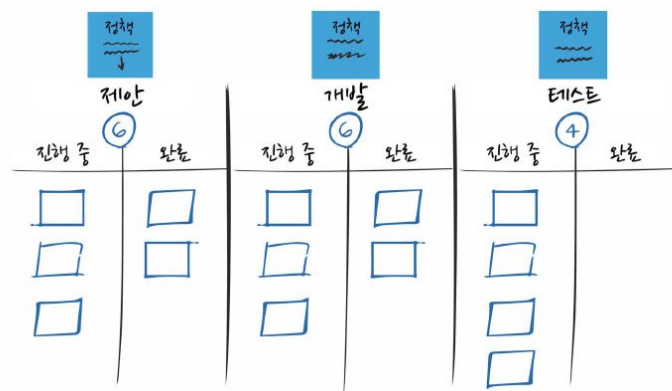
칸반을 성공시키려면 진행 중 업무의 양을 관찰하고 제한해서 최적화하는 것이 필수적이다. 그렇게 하면 서비스 리드 타임이 개선되고, 품질이 향상되며, 산출물을 내놓는 속도가 더 빨라진다. 진행 중 업무(WIP)에 제한을 도입하고 그 제한을 준수하면 당김 시스템으로 바뀌게 된다. 만약 이런 제한이 없다면 오히려 사람은 지치면서 리드타임이 늘어난다.

- 흐름을 관리한다.

칸반 시스템에서 업무 흐름은 가치 제공을 최대화하고, 리드 타임을 최소화해야 한다. 때로는 이러한 목표들이 서로 충돌하기도 하는데, 대개는 제품이 복잡하기 때문이며, 그래서 투명성, 검토, 적용을 통한 경험적 제어가 필요하다. 프로세스 상의 버블넉을 찾아내어 지속적으로 해소해야 한다.

- 정책을 명시화한다.

명시적 정책이란 칸반보드의 상단에 있는 워크플로우 항목에 대한 정책을 의미하며 예를 들어 한 열에서 다음 열로 넘어갈 때 명확한 완료 정의(예: 테스트의 종료 기준)나 어떻게 보충할 것인지를 결정 하는 것을 말한다.



출처 : 에센셜 칸반 요약 가이드, 2016

- 피드백 고리를 정한다.

프로세스의 점진적 변화와 효과적인 서비스 제공을 위하여 주기적 리뷰 미팅을 수립한다. 예를 들어 제품백 로그에 대한 우선순위 미팅이나 데일리 스탠드업 미팅 같은 것을 정의한다.

- 함께 개선하고, 실험을 통해 발전시킨다

칸반은 근본적으로 프로세스 개선을 위한 방법으로 조직의 현재 상태에서 시작하며 지속적이고 점진적인 개선을 추구한다. 칸반은 조직 내에서 유익한 변화가 일어나도록 하고, 자연스러운 또 다른 진화 프로세스, 즉 멸종으로부터 그 변화를 보호하기 위해서 진화 프로세스를 활용한다.



## CHAPTER 03

### 전통적 방법론과 애자일 방법론의 장단점



- |     |                         |    |
|-----|-------------------------|----|
| 3.1 | 전통적 vs. 애자일 개발 방법론의 차이점 | 26 |
| 3.2 | 전통적 vs. 애자일 개발 방법론의 장단점 | 27 |

## CHAPTER 03

## 전통적 방법론과 애자일 방법론의 장단점

## 3.1 전통적 vs. 애자일 방법론의 차이점

비즈니스 환경의 빠른 변화에 대응하기 위하여 등장한 것이 애자일이다 보니 전통적 방법론과는 일부 공통적인 요소도 있지만 여러 가지 면에서 차이점이 있는 것이 사실이다. 특히 제품개발의 성공이 전통적 방법론에서는 체계적인 프로세스 수행에 초점을 두었다면 애자일에서는 사람들간의 소통과 협력에 있다고 보는 점이 다르다. 그래서 애자일 방법론은 사람들의 소통과 협력을 강화 할 수 있는 프랙티스를 많이 제공하는 반면에 전통적 방법론은 표준화 및 정형화된 프랙티스를 강조하는 경향이 있다.(참고문헌 4) 아래 표는 전통적 개발방식과 애자일이 추구하는 목표와 관리 방법의 차이를 설명하였다.(참고문헌 4,5,6)

| 구 분     | 전통적 개발                         | 애자일 개발                           |
|---------|--------------------------------|----------------------------------|
| 적용환경    | 요구사항이 비교적 명확하고 리스크가 낮은 경우      | 요구사항이 불확실성과 리스크가 높은 경우           |
| 통제방식    | Process - driven               | People - driven                  |
| 프로세스 모델 | 정의된 (Defined) 프로세스             | 경험적(Empirical) 프로세스              |
| 계획수립    | 초기에 최대한 요구사항을 확정하고 상세 계획 수립    | 개략적인 범위와 일정을 수립하고 주기적으로 상세 계획 수립 |
| 개발 모델   | 폭포수 중심                         | 점진적/반복적 중심                       |
| 사용자 참여  | 사용자 참여가 제한적                    | 사용자의 지속적인 참여                     |
| 요구변경    | 변경을 가급적 억제                     | 변경을 주기적으로 포용                     |
| 프로젝트 목표 | 계획 준수                          | 고객 가치 증대                         |
| 리더십     | 명령과 통제, 수직적 관계                 | 코칭과 퍼실리테이션, 수평적 관계               |
| 팀원 역할   | 개인 책임으로 일하며 한정된 역할 수행 (I자형 인재) | 팀 책임으로 일하면 포괄적인 역할 수행(T자형 인재)    |
| 조직      | 기능(Functional) 팀               | 기능복합(Cross-functional) 팀         |
| 팀 평가    | 상위관리자 평가, 상대평가                 | 다면 평가, 절대평가                      |

### 3.2 전통적 vs. 애자일 개발 방법론의 장단점

소프트웨어 개발을 성공적으로 이끄는 유일한 방법론은 없으며 개발팀 리더는 프로젝트 상황에 따라 적절한 방법을 찾아야 한다. 그런 측면에서 애자일과 전통적 방법론은 각각의 장단점을 가지고 있으며 현실에서는 이들을 상호보완적으로 활용하는 지혜가 필요하다.(참고문헌 4) 애자일은 소규모 프로젝트에서 태동하다 보니 대규모 개발 프로젝트에 필요한 프로세스를 별로 가지고 있지는 않다. 또한 전통적 개발에서 나타나는 문제점들을 개선하려는 취지에서 발생되다 보니 기존 방법론에서 잘 활용되고 있는 부분에 대해서는 별다른 언급이 없다. 예를 들어 기존 방법론에서 활용되는 각종 분석/설계 기법과 템플릿 등에 대해서 애자일은 거의 언급하고 있지 않다. 애자일 개발에서 산출물 최소화를 권장하는 것은 가치가 없는 활동을 하지 말라는 뜻이지 산출물을 아예 만들지 말라는 뜻은 아니다. 대규모 프로젝트 및 안전성(Safety)이 요구되는 제품 개발 같은 곳에서는 다양한 산출물이 필요하다. 이럴 때는 기존 방법론의 템플릿을 활용하는 것이 도움이 된다. 최근에는 애자일의 장점과 전통적 방법론의 장점을 혼합한 하이브리드형 방법론들이 출현하고 있는데 이런 추세는 앞으로도 이어질 것 같다.(참고문헌 7,8) 아래 표에서는 전통적 방법론과 애자일 방법론의 장단점을 설명하였다.(참고문헌 5, 9, 10)

| 구 분        | 장점                                                                                                                                                                                                                 | 단점                                                                                                                                                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 전통적 개발 방법론 | <ul style="list-style-type: none"> <li>- 체계적인 소프트웨어 개발에 필요한 분석/설계/구현/테스트 프로세스 및 기법, 템플릿 등을 제공</li> <li>- 대규모 프로젝트 수행에 필요한 관리 요소 등을 제공</li> </ul>                                                                   | <ul style="list-style-type: none"> <li>- 과도한 개발 산출물이 발생할 수 있음</li> <li>- 초기 계획 준수에 따라 불필요한 기능 개발이나 일정, 품질 저하가 발생할 수 있음</li> <li>- 분석단계 이후의 요구변화 수용이 어려움</li> <li>- 사람들간의 소통과 협력을 위한 활동이 빈약함</li> </ul>                                                                                          |
| 애자일 개발 방법론 | <ul style="list-style-type: none"> <li>- 요구변화를 주기적으로 포용</li> <li>- 개발 과정의 낭비요소를 줄일 수 있는 효과적인 지침을 제공</li> <li>- 개발 구성원들간의 소통과 협력을 원활하게 하는 다양한 활동을 제공</li> <li>- 수평적인 의사결정 및 팀원들의 사기를 높이기 위한 방법 및 리더십을 제공</li> </ul> | <ul style="list-style-type: none"> <li>- 기존 활동과는 전혀 다른 애자일 활동을 익히기가 어려움</li> <li>- 애자일이 지향하는 개발 문화와 자기 조직화된 팀을 만들기가 어려움</li> <li>- 사용자가 지속적으로 참여하기 어려운 환경에서는 적절하지 않음</li> <li>- 체계적인 소프트웨어 개발에 필요한 분석/설계/구현/테스트 프로세스 및 설계 기법 등을 제공하지 않음</li> <li>- 대규모 프로젝트 수행에 필요한 관리 요소 등을 제공하지 않음</li> </ul> |

애자일은 원칙적으로 모든 개발 환경에 적용할 수 있지만 좀더 효과적으로 작용하는 환경이 존재한다. 다음은 애자일과 전통적 개발에 적합한 환경을 설명한 것이다.(참고문헌 9)

| 구 분     | 애자일 개발에 적합                                                                                                           | 전통적 개발에 적합                                                                                                                            |
|---------|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 시장 환경   | 고객 선호도 및 솔루션 조건이 자주 변경된다.                                                                                            | - 시장 상황은 안정적이며 예측 가능하다.                                                                                                               |
| 고객 참여   | 긴밀한 협력과 신속한 피드백이 가능하다.<br>고객은 프로세스가 진행됨에 따라 원하는 것을 더 잘 알게 된다.                                                        | - 요구 사항은 처음부터 분명하며 안정적이다.<br>- 고객은 개발팀과 지속적인 협력이 어렵다.                                                                                 |
| 혁신 유형   | - 문제가 복잡하고 해결 방법을 알 수 없다.<br>- 범위가 명확하게 정의되어 있지 않다.<br>- 제품 사양이 변경 될 수 있으며 창조적인 혁신과 시장 출시가 중요하다.<br>- 기능 간 협력이 중요하다. | - 이전에도 이와 유사한 작업이 이루어졌으며 솔루션이 명확하다.<br>- 상세한 사양 및 작업 계획은 확신을 가지고 예측 될 수 있다.<br>- 사양을 준수하는 것이 중요하다.<br>- 기능적 사일로에서 문제를 순차적으로 해결할 수 있다. |
| 업무의 모듈화 | - 점진적 개발이 가치가 있으며 고객은 이를 사용할 수 있다.<br>- 작업을 여러 부분으로 분해하고 신속하고 반복적인 주기로 수행 할 수 있다.<br>- 낮은 변경 사항을 관리 할 수 있다.          | - 고객은 모든 것이 완료 될 때까지 제품의 부분을 테스트 할 수 없다.<br>- 낮은 변화는 비싸거나 불가능하다.                                                                      |

〈출처 : Bain & Company〉

## CHAPTER 04

### 애자일 방법론의 국내외 도입 추세



|                      |    |
|----------------------|----|
| 4.1 해외 애자일 방법론 도입 추세 | 30 |
| 4.2 국내 애자일 방법론 도입 추세 | 37 |

## CHAPTER 04

## 애자일 방법론의 국내외 도입 추세

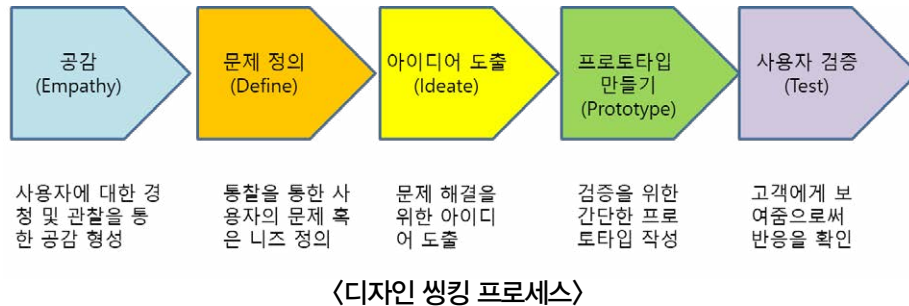
## 4.1 해외 애자일 방법론 도입 추세

2001년 애자일 개발 선언문 이후 애자일은 선도적인 실천가들에 의해 성공적인 사례들이 만들어지면서 미국과 유럽을 중심으로 전세계에 퍼져 나가게 된다. 최근에서 소프트웨어 개발 분야뿐만 아니라 제조, 금융, 교육, 미디어 등 일반 산업에까지 확장되고 있는 추세이다.(참고문헌 8) 하버드 비즈니스 리뷰 최신호에서는 애자일이 지난 25년간 소프트웨어 개발팀에게 동기부여와 생산성을 강화시킴으로써 소프트웨어 개발의 성공률과 속도를 높이고 있다고 설명하면서 비 IT분야의 경영자들도 이러한 애자일 방식을 활용하여 경쟁력을 높일 것을 권장하고 있다.(참고문헌 9) 기업 경영 환경에 불확실성이 커지면서 소프트웨어 개발을 넘어 기업 경영에 애자일 방식을 반영하는 노력이 나타나고 있다. 불확실한 경영 환경에서 완벽한 장기 비즈니스 플랜을 세워두고 달려가는 것보다는 탐색적 실험과 관찰, 적응을 통해서 예측하지 못했던 상황에 대응하는 것이 보다 전략적인 방안이 될 수 있기 때문이다. 이제 애자일은 다양한 기업들이 혁신을 위한 방법론으로 활용하고 있다고 Forbes, CEO Magazine은 말하고 있다.

사실 XP를 제외한 스크럼이나 린, 칸반은 업무성과를 높이기 위한 프로세스 개선 방법론이기 때문에 굳이 소프트웨어 산업에만 적용될 이유는 없다. 개인 재능보다는 사람들간의 소통과 협업을 중시하는 애자일 개발 문화는 다른 분야에서도 충분히 가치를 제공할 수 있기 때문이다. 최근 조사한 자료에 따르면 스크럼을 사용하는 프로젝트의 약 20%가 IT 분야가 아니라 마케팅이나 재무, 인사 등에 활용되고 있다고 한다.(참고문헌 8) IT가 아닌 타 부서에서 애자일을 사용하게 되는 계기는 대부분 IT조직에 적용된 성과를 보고 이를 활용하려고 하는 의도에서 비롯되는 경우가 많다. 애자일에서 제시하는 조직 문화는 직원들에게 동기를 부여하면서 업무 효율성을 높이고 더 나은 관리자가 될 수 있도록 지원하기 때문이다. 최근에는 애자일 방법론을 그대로 활용하기 보다는 다른 방법론들과 결합하면서 그 영역을 더욱 넓히고 있다. 디자인 씽킹(Design Thinking)이나 린 스타트업(Lean Startup), 데브옵스(DevOps)는 이런 측면에서 기존 애자일 방법론들을 보완해주는 좋은 방법론들이다.

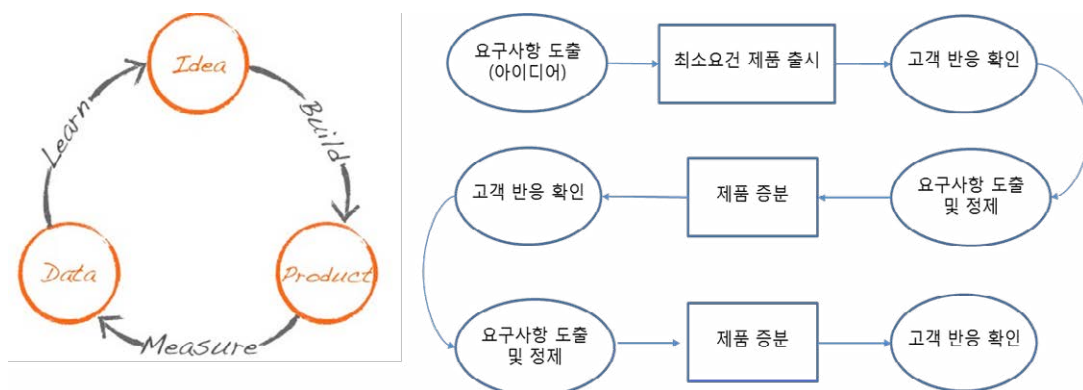
#### 4.1.1 디자인 씽킹과 린 스타트업, 애자일의 결합

디자인 씽킹(Design Thinking)은 세계적 디자인 기업인 IDEO에서 나온 디자인 방법론으로 스탠포드 D스쿨을 통해서 널리 알려졌다. 이 방법은 최종 사용자가 경험하게 될 해결책에 집중해 문제를 풀어나가는 방법을 의미하며 다음과 같은 프로세스를 가지고 있다.



전통적으로 스크럼과 린 같은 애자일 방법론은 문제를 알고 있다는 가정하에서 어떻게 하면 효과적으로 개발할 것인지에 초점을 두어 왔다. 하지만 제품 개발의 성공은 개발 효율성 못지 않게 올바른 문제를 발견하는 것 역시 중요하다. 이를 소홀히 하게 되면 사용자에게 외면 받는 제품을 만들 수 있기 때문이다. 디자인 씽킹은 이런 문제점을 해결하기 위하여 문제를 정의하기 전에 ‘공감’이라는 프로세스를 넣어서 전문가의 오류를 방지하고 인간 중심적 사고를 하기 위해 사용자를 심도 있게 관찰할 것은 요구하고 있다. 공감이란 활동에는 직접 현장에 나가 사용자의 행태를 관찰하거나 직접 체험 해보는 일련의 과정들을 포함하고 있는 데 이런 과정을 겪지 않고서는 진정으로 사용자의 문제를 발견하기 어렵기 때문이다. 디자인 씽킹은 애자일 방법론이 본격적으로 수행되기 전인 제품 개발 초기에 컨셉을 잡는 차원에서 활용된다.

디자인 씽킹이 가벼운 프로토타입을 통하여 올바른 요구사항을 발견하는 데 초점을 맞추었다면 린 스타트업은 벤처 기업의 실패로 인한 비용을 최소화하기 위하여 만들어진 것으로 최소 요건만을 갖춘 제품(MVP, Minimal Viable Products)을 신속하게 개발하여 시장에 출시하고 사용자의 피드백을 바탕으로 지속적으로 업그레이드하는 방법론이다. 기존의 벤처기업들은 먼저 핵심 아이디어를 도출한 후 일정을 짜고 필요한 예산을 마련한 다음에 제품개발을 시작했다. 그리고 개발이 끝날 때까지 전 과정을 철저히 비밀에 부치는 게 일반적이었다. 하지만 이런 방식은 제품이 오랜 시간과 많은 비용을 들인 후에야 시장에 나오므로, 반응이 좋지 않으면 돈이 바닥나서 살아남기가 어려웠다.



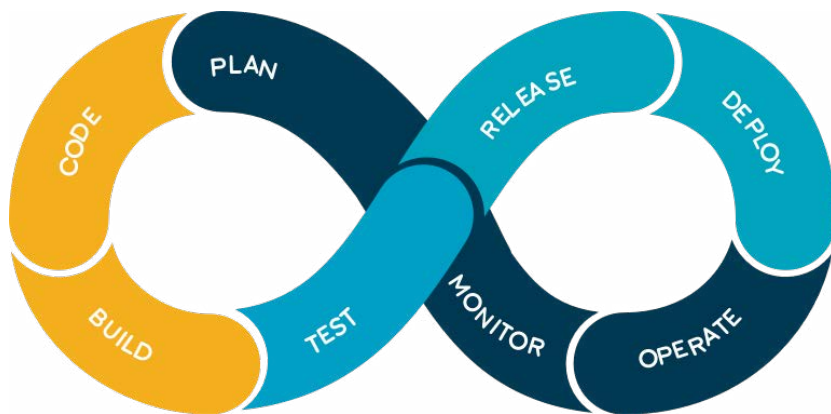
린 스타트업은 이런 장시간의 노력과 비용을 획기적으로 줄이려는 목적으로 만들어졌으며 과도한 기획과 분석을 지양하고 아이디어가 나오면 이를 바탕으로 고객에게 가치가 있는 핵심 기능을 빠르게 출시하는 것을 특징으로 한다. 만약 고객 반응이 좋지 않으면 제품 방향을 전환하거나 새로운 제품을 모색한다.(참고문헌 11) 이런 방식으로 개발된 것이페이팔이나 트위터, 슬랙 같은 제품들이다. 다른 애자일 방법론에 비해서 린 스타트업은 린 비즈니스 캔버스 같은 비즈니스 측면에서의 프랙티스들도 제공하고 있어서 단순히 개발 프로세스를 넘어서고 있다.

최근에는 제너럴 일렉트릭이나 코카콜라, 삼성전자 등 대기업에서 새로운 프로젝트를 수행할 때 린 스타트업을 활용하고 있다. 디자인 씽킹과 린 스타트업, 애자일 방법론이 결합하면 보다 효과적인 제품개발을 진행할 수가 있다. 예를 들어 문제를 발견하는 단계에는 디자인 씽킹이 효과적으로 활용될 수 있으며 이후에는 린 스타트업을 통해서 최소 제품을 만들고 스크럼과 린을 활용하여 효율적으로 개발을 진행하는 방식이다. 이러한 형태는 구글이나 어도비, 페이스북 등 많은 실리콘 밸리 기업들에게서 많이 활용되고 있다.

#### 4.1.2 개발과 운영을 포괄하는 데브옵스의 확산

데브옵스(DevOps)는 소프트웨어의 개발(Development)과 운영(Operations)의 합성어로서, 소프트웨어 개발자와 운영자 간의 소통 및 협업, 통합을 강조하는 개발 환경이나 문화를 말한다. (위키백과) 애자일 개발이 추구하는 중요한 목표 중의 하나는 고객의 요구사항에 빠르게 대응하여 릴리즈를 빈번하게 할 수 있는 환경을 만드는 것이라고 할 수 있다. 그런데 조직적으로 소프트웨어 개발자 그룹과 시스템 운영자(Ops) 그룹으로 나누어져 있다면 이들간의 협업이 문제가 될 수 있다. 개발자는 개발에만 집중하다 보니 시스템의 운영상에서 발생하는 문제를 거의 고려하지 않는 경향이 있다. 반면에 운영자는 시스템을 안정적으로 운영하는 역할을 맡고 있다 보니 안정성을 깨뜨리는 변화를 주고 싶지 않은 경향이 있다.

이런 구조에서는 개발이 아무리 빨리 끝났어도 빈번하게 릴리즈 할 수 없는 상황이 만들어지게 된다. 데브옵스는 이런 상황을 해결하기 위하여 등장한 방법론으로 기존 애자일 방법론들이 주로 사용자와 개발자, 테스터 간의 협업에 초점을 맞추었다면 데브옵스는 주로 개발자와 테스터, 시스템 운영자간의 협업에 초점을 맞추고 있다. 데브옵스 모델에서는 개발 팀과 운영 팀이 더 이상 사일로에 묶여 있지 않고 두 팀이 단일 팀으로 병합될 것을 권장한다. 그래서 엔지니어가 개발에서 테스트, 배포, 운영에 이르기까지 전체 애플리케이션 수명 주기에 걸쳐 작업하며 보안 팀이나 품질보증 팀과도 긴밀하게 협력하는 구조이다.



〈출처 : <https://www.zaizi.com/devops>〉

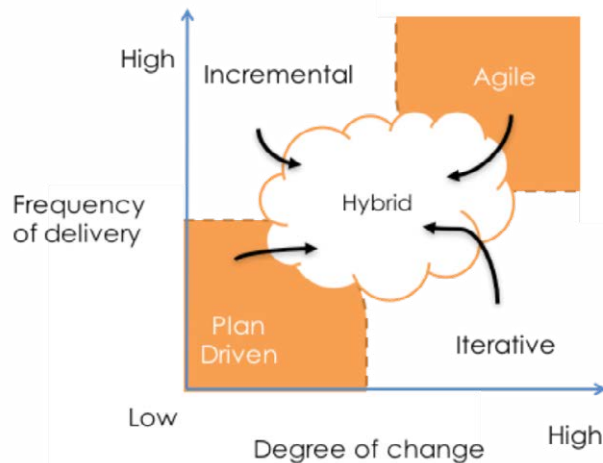
〈데브옵스 모델〉

데브옵스는 주로 빈번한 배포가 필요한 웹 기반의 비즈니스나 클라우드 환경의 제품들에서 많이 도입되고 있으며 높은 안정성과 신뢰성을 요구하는 시스템에서는 필요성을 따져볼 필요가 있다. 데브옵스에서는 다음과 같은 주요 활동과 톨 체인 구축을 권장한다.(참고문헌 12)

- 개발과정에 있어서 운영자의 지속적인 참여
- 자동화된 테스트
- 통합된 형상 관리(Integrated Configuration Management)
- 지속적 통합(Continuous Integration)
- 지속적 출시(Continuous Deployment)
- 애플리케이션 모니터링(Application Monitoring)

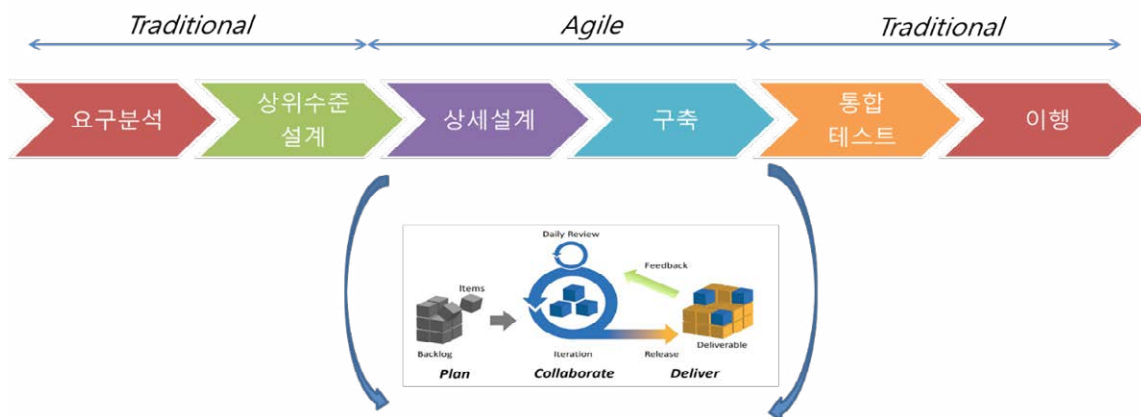
### 4.1.3 하이브리드 애자일 방법론의 확산

애자일 방법론들이 소규모 개발에서 시작되다 보니 대규모 프로젝트나 조직에 적용하기에는 한계가 있는 것이 사실이다. 예를 들어 프로그램 및 포트폴리오 관리와 같은 거버넌스 측면이나 외주 개발이 많은 대규모 프로젝트, 하드웨어 개발이 포함된 프로젝트 등 현실에서 나타나는 다양한 상황을 기존 애자일 방법론만으로 해결하기는 어렵다. (참고문헌 13) 또한 애자일을 적용하려면 경영진의 마인드 셋이나 조직 구조가 바뀔 필요가 있는데 이런 것들은 쉽게 변화하기가 어렵다. 현실적으로 전통적 개발에 익숙한 조직에서는 바로 애자일로 넘어가기 어렵다. (참고문헌 7) SAFe(Scaled Agile Framework)나 DAD(Disciplined Agile Delivery) 등은 이런 문제들을 풀기 위하여 등장한 하이브리드 형태의 확장된 애자일 방법론들이다. 이 방법론들은 기본적으로 스크럼과 린에 기반을 두면서 기존 애자일에서 소홀히 다루었던 아키텍처 및 설계 활동, 프로그램 관리, 통합 테스트 전략 등에 대한 내용들을 다루고 있다.



(출처 : <https://www.agilealliance.org/what-is-hybrid-agile-anyway/>)

또한 요구사항의 불확실성과 프로젝트의 복잡성을 고려하여 워터폴과 애자일을 아래 그림과 같이 혼합한 형태로 적용하기도 한다. 하이브리드 방식은 프로젝트 상황에 따라 다양하게 적용할 수 있지만 기본적으로 애자일 원리를 근간으로 한다.



〈하이브리드 애자일 개발 방식〉

#### 4.1.4 주요 애자일 방법론과 도입 기법

2016년 전세계 소프트웨어 커뮤니티를 대상으로 조사한 자료에 따르면 애자일 방법론 중에서 가장 많이 활용되는 것은 스크럼으로 약 60% 가까이 되는 사람들이 사용하고 있으며 그 다음으로는 스크럼과 XP를 혼합해서 사용하거나 여러 방법론들을 혼합해서 사용하는 하이브리드 애자일 방식으로 사용하는 경우가 많이 나타났다.(참고문헌 14)

- 스크럼 (58%)
- 스크럼과 XP 혼합(10%)
- 하이브리드 애자일 (8%)
- 스크럼과 칸반의 혼합(8%)
- 칸반(5%)
- 기타

애자일 방법론들에 다양한 기법과 테크닉들이 있는데 그 중에서 가장 많이 활용되는 프랙티스들은 아래와 같다.(참고문헌 14)

- 이터레이션 계획 수립
- 데일리 스탠드업 (Daily standup) 미팅
- 회고 (Retrospectives)
- 이터레이션 리뷰(Iteration reviews)
- 릴리즈 계획 수립
- 팀 중심의 추정(Team-based estimation)
- 지속적인 통합(Continuous integration)
- 제품 책임자 역할 정립
- 리팩터링(Refactoring)
- 테스트 주도 개발(Test-driven deployment)

#### 4.1.5 애자일 도입 기업

2016년 전세계 비즈니스 및 IT 경영진 1770명을 대상으로 조사되었던 자료에 따르면 응답자의 88%가 애자일을 도입해서 사용하고 있다고 한다. 이중에서 IT를 넘어 전사차원으로 애자일을 적용하는 기업은 약 30% 정도이다.(참고문헌 15) 이 자료를 보면 전 세계 대다수의 IT 조직들이 애자일을 도입한 상태라고 볼 수 있다. 하지만 도입한 조직들은 애자일을 모든 팀과 프로젝트에 적용하지는 않으며 업무 상황에 따라 유연하게 적용한다. 다음은 애자일을 적용하고 있는 주요 기업들이다.

##### ■ SW기업

- Microsoft, Adobe, SAP, Yahoo, Google, Facebook, Twitter, AOL, BMC
- Intuit, Salesforce.com, VMware, CA Technologies, Autodesk, Citrix etc.

##### ■ 제조업

- GE, IBM, HP, PHILIPS, Dell, Cisco, Nokia, ABB, Saab, John Deere, Boeing
- Raytheon, DaimlerChrysler, Delphi Corp, General Motors, Fujits etc.

## ■ 기타

- Bank of America, City bank, Aviva, Adelaide Bank Australia, US Bank
- Moody's, S&P, AVIVA, NOMURA, NASDAQ, Federal Home Loan Bank Etc.

### 4.1.6 애자일 도입 효과

2016년 전세계 비즈니스 및 IT 경영진 1770명을 대상으로 조사되었던 자료에 따르면 애자일 도입효과를 아래와 같이 얘기하고 있다.(참고문헌 15)

1. 직원 생산성 향상
2. 고객만족도 증가
3. 개발품질 향상
4. 개발 및 운영 효율성 증가
5. 신규 비즈니스 성장
6. IT 비용 감소

2016년 전세계 소프트웨어 커뮤니티를 대상으로 조사한 자료에 따르면 애자일 도입효과를 아래와 같이 얘기하고 있다.(참고문헌 14)

1. 요구사항 우선순위 관리 능력 향상
2. 프로젝트의 투명성 및 가시성 향상
3. 팀의 생산성 향상
4. 시장 출시 기간 단축
5. 팀 사기 향상
6. 소프트웨어 품질 향상
7. 프로젝트 예측력 향상

애자일을 잘 적용하게 되면 먼저 요구사항 우선순위 관리를 통하여 비즈니스와 IT 간의 연결성을 높일 수가 있으며 타스크 보드와 같은 시각적 관리 방식을 이용하여 프로젝트의 투명성을 높일 수가 있다. 이런 투명성을 바탕으로 팀원들간의 소통과 협력을 강화시켜 나가면 생산성과 품질 모두 올라가는 효과가 있다.

## 4.2 국내 애자일 방법론 도입 추세

### 4.2.1 IT 조직을 중심으로 확산 중

국내에 애자일이 도입된 것은 2000년 대 초반으로 사용자에게 민감한 다음이나 네이버 같은 인터넷 기반의 웹 서비스 기업을 중심으로 도입되었으며 점차 게임개발이나 솔루션 개발 기업 등으로 퍼지게 된다. 현재는 IT 유지보수 조직이나 SI 프로젝트에도 애자일이 점차 확산되고 있는 추세에 있다. 하지만 아직까지 공공기관이나 국방, 자동차와 같이 엄격한 규제가 있는 소프트웨어 개발까지는 영향을 미치지 못하고 있다. 해외에서는 IT 가 아닌 조직에서도 애자일이 확산되면서 성공사례가 많이 나타나고 있지만 국내에서는 IT 조직 외에는 이렇다 할 성공사례가 나오지 않고 있다. 애자일을 선도적으로 도입했던 국내 기업들도 혁신적인 제품을 출시하기 위하여 디자인씽킹과 린스타트업, 애자일을 결합하여 활용하고 있으며 최근에는 데브옵스에도 관심을 가지고 활발히 도입하고 있는 추세에 있다.(참고문헌 16)

네이버와 카카오 같은 인터넷 기업들은 애자일이 추구하는 인간 존중이나 수평적 소통 문화를 비교적 빠르게 받아들이고 있으며 조직구조도 신속한 의사결정이 가능한 소규모 셀 조직으로 바꾸고 있다. 국민 메신저로 등극한 '카카오톡' 은 애자일 방식으로 만든 대표적인 제품이라 할 수 있는데 이 제품은 초기에 린 스타트업 방식을 활용하여 2개월만에 최소 제품(MVP)을 만들고 이후에는 애자일 방식으로 지속적으로 개발하고 있다. 카카오의 개발 문화를 보면 애자일이 추구하는 철학과 거의 동일하다.(참고문헌 17)

- 가능한 모든 정보를 구성원들과 공유한다.
- 빠른 의사결정을 위해 필요한 권한을 팀원이라도 책임자에게 위임한다.
- 업무 협의를 할 때에는 팀장과 팀원 구별 없이 수평적인 커뮤니케이션을 추구한다.
- 다른 부서의 일도 우리의 일처럼 관심을 가진다.
- 빠르게 실행하고 만약, 실패한다면 빠르게 개선한다.

한편 수직적인 업무방식에 익숙한 대부분의 국내 기업들은 애자일이 추구하는 개발 문화를 좀처럼 받아들이지 못하고 있는 것도 사실이다. 많은 기업들이 제조업에서 효과적으로 작용했던 수직적 업무방식에 익숙하다 보니 애자일에서 권장하는 서번트 리더십이나 수평적 소통 문화를 잘 받아들이지 못하고 있다. 그러다 보니 애자일 활동 중에서 손 쉬운 도구 및 기술적인 부분만 받아들이고 있는 상황이다.

삼성SDS, 엘지CNS 와 같은 IT서비스 기업을 중심으로 대규모 개발 프로젝트에도 하이브리드 애자일이 지속적으로 시도되고 있다. 교과부 차세대 교육행정정보시스템(NEIS) 구축, 범한 판토스 글로벌 물류시스템 확대 구축, 솔로몬 저축은행 차세대 프로젝트, 현대 카드 차세대 프로젝트 등 여러 대형 프로젝트들에도 일부 애자일 방식이 적용된 것으로 알려져 있다.(참고문헌 18) 하지만 국내 대형 시스템 통합 프로젝트인 경우 고객의 이해 부족 및 개발사의 애자일 수행 역량 부족 등으로 인하여 성공적으로 적용되지는 못하고 있는 상황이다. (참고문헌 19, 20)

### 4.2.2 애자일 도입 기업

국내에서도 많은 기업들이 애자일을 활용하여 제품을 개발하고 있으나 문화적인 차이로 인하여 해외에 비해서는 확산 속도가 매우 느린 편이다. 국내에서 애자일을 도입한 조직들 역시 모든 팀이 애자일을 적용하는 것은 아니며 팀의 자율적인 선택에 따라 적용된다. 예를 들어 어떤 팀들은 애자일의 일부 프랙티스들만 활용할 수도 있다. 다음은 애자일을 적용하고 있는 주요 기업들이다.

- **SW기업**

- 카카오, 네이버, 쿠팡, 티몬, 엔씨소프트, 넥슨, 삼성SDS, 엘지CNS
- SK C&C, 엔키아, 한글과컴퓨터, 다우기술, 티맥스소프트, 핸디소프트 등

- **제조업**

- 삼성전자, 엘지전자, 두산중공업, 하이닉스 반도체, 루트로닉 등

- **기타**

- 신한은행, 국민은행, ING 생명, 하나은행, 한화투자증권, 신한투자금융 등

### 4.2.3 주요 애자일 기법

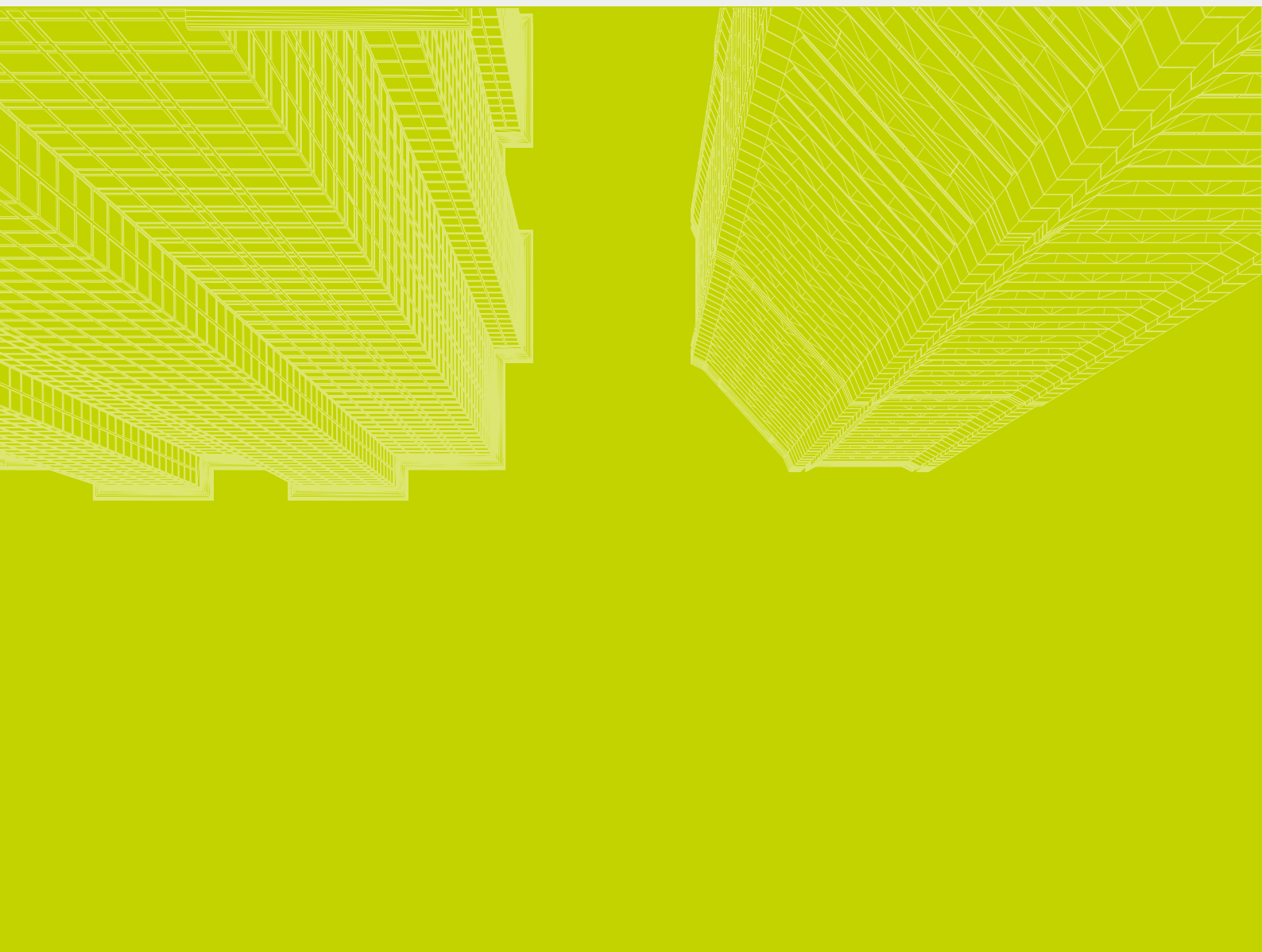
국내에서 가장 많이 활용되는 애자일 프랙티스들은 아래와 같으며 해외와 유사하게 도입되고 있으나 회고 및 데일리 스탠드업 (Daily standup) 미팅을 많이 하는 것으로 나타났다. (부록참조)

- 회고 (Retrospectives)
- 데일리 스탠드업 (Daily standup) 미팅
- 작업 진행 상황판(Task Board)
- 릴리즈 계획 수립
- 스크럼 마스터 역할 정립
- 지속적인 통합(Continuous integration)
- 자동화된 단위 테스트
- 테스트 주도 개발(Test-driven deployment)
- 사용자 스토리
- 리팩터링(Refactoring)

#### 4.2.4 애자일 도입 효과

국내 애자일 도입 현황을 조사한 자료에 따르면 국내에서도 많은 조직들이 애자일 도입 효과를 얘기하고 있는 데 그 중에서 상위 7개를 나열하면 아래와 같다. 국내에서는 해외 조사와는 다르게 팀의 생산성과 업무효율성 향상이 가장 큰 것으로 나타났다. (부록참조)

1. 팀의 생산성 및 업무효율성 향상
2. 변화하는 요구사항에 대한 조절 능력 향상
3. 비즈니스와 IT간의 효과적인 소통
4. 소프트웨어 품질 향상
5. 프로젝트의 투명성 및 가시성 향상
6. 시장 출시 기간 단축
7. 프로젝트 예측력 향상



## CHAPTER 05

### 경량 개발 방법론 도입의 국내 성공, 실패 사례



|     |             |    |
|-----|-------------|----|
| 5.1 | 성공 및 실패의 기준 | 42 |
| 5.2 | 국내 성공 사례    | 42 |
| 5.3 | 애자일 실패 사례   | 51 |

## CHAPTER 05

## 경량 개발 방법론 도입의 국내 성공, 실패 사례

## 5.1 성공 및 실패의 기준

애자일 개발 선언문의 첫 번째 원리를 보면 소프트웨어 개발의 최우선 목표는 고객에게 가치 있는 소프트웨어를 빠르게 지속적으로 전달하는 것이라고 얘기하고 있다. 이런 관점에서 애자일 성공의 기준은 앞 절에서 설명했던 팀의 생산성이나 품질, 시장 출시기간 단축과 같은 비즈니스 효과들이 나타나야 한다.(참고문헌 21) 또한 애자일이 단발적으로 수행되기 보다는 조직 차원에서 애자일 철학과 원리를 받아들이는 모습이 나타나는 것도 필요하다. 따라서 본 연구에서는 애자일 도입의 성공기준을 아래와 같이 정의한다.

- 애자일 적용을 통하여 시장 출시기간 단축이나 생산성 같은 비즈니스 효과가 나타나고 있는가?
- 고객을 포함한 경영진과 직원 모두 만족도가 향상되고 있는가?
- 조직 차원에서 애자일 업무 방식이 지속적으로 유지되고 있는가?

## 5.2 국내 성공 사례

## 5.2.1 (주)엔키아 사례

## 1) 회사 개요

이 회사는 IT운영 관리 솔루션 개발 전문 기업으로, IT서비스를 구성하는 IT인프라의 장애와 성능을 실시간으로 모니터링 하는 제품과 클라우드 운영/관리 솔루션, ITIL 기반의 ITSM 솔루션 등 건강하고 장애 없는 IT서비스 운영/관리를 위한 전반적인 제품을 개발하고 있는 국내 대표적인 IT솔루션 전문기업이다. 또한 최근엔 스마트팩토리, 스마트시티 등을 위한 IoT 플랫폼을 개발하여 신제품 출시를 앞두고 있는 등 IT의 흐름에 대응하는 끊임없는 투자와 연구를 지속하고 있다.

## 2) 애자일 적용 상황

현재 개발 인력은 130여명 정도로 이 중에서 사업지원부서가 아닌 순수 솔루션 개발을 하는 팀에 애자일 방식을 적용하여 개발하고 있다. 처음 애자일 도입은 2015년도에 신제품을 개발하는 1개팀에 먼저 도입 되었으며 그 이후 어느 정도 적용 효과가 나타남에 따라 다른 팀으로 확산되었다.

## 3) 애자일 도입 배경

처음에는 개발자 중에 몇몇이 애자일을 활용하여 개발해보자는 의견이 있어서 스크럼을 먼저 도입하였으며 스크럼을 어느 정도 알고 있는 개발자 중에 1명이 스크럼마스터로 선정되었다. 처음에는 그런대로 진행이 되었으나 이내 난관에 봉착하게 되었다. 왜냐하면 스크럼마스터 역할을 하는 사람이 개발을 병행하다 보니 스크럼을 피상적으로 이해할 수 밖에 없었으며 스프린트 계획 및 리뷰, 회고 등 여러 가지 스크럼 활동이 만

족스럽게 진행 되지를 못했다. 그래서 조직차원에서 애자일 코치의 도움을 받기로 하였으며 부족한 부분을 지원받으며 나름대로 성공적으로 진행하게 되었다.

#### 4) 애자일 적용 초기 문제점

다음은 애자일 적용 초기에 나타났던 문제점들이다.

- 개발팀 리더의 애자일 지식 및 역량 부족

처음에는 개발자 중에서 1명이 스크럼 마스터 역할을 하다 보니 팀 리더는 애자일에 대한 지식이 거의 없었으며 제대로 이해를 하지 못한 상황이었다. 그래서 데일리 스크럼 미팅이나 회고에 거의 참여를 하지 않았으며 팀 코칭에 대한 개념이 거의 없었다. 하지만 애자일 코칭을 받으면서 팀 리더에게 지식과 경험이 쌓여갔으며 나중에는 자연스럽게 스크럼마스터 역할도 수행할 수 있게 되었다.

- 스프린트 계획 및 리뷰 활동의 부실

적용 초기에는 스프린트 계획 수립 시 전체 구성원이 참여하지 않고 전통적 방식대로 소수의 파트리더가 각자의 타스크를 추정하는 방식으로 진행되었다. 그러다 보니 다양한 의견들이 반영되지 못하고 추정의 신뢰성도 떨어졌으며 리뷰 시 이해관계자들간의 갭이 많이 발생하게 되었다. 하지만 애자일 코칭을 받으면서 디자이너를 포함한 전체 구성원들이 스프린트 계획에 참여하여 다양한 의견들을 개진하고 사전에 스토리의 인수기준을 명확하게 함으로써 리뷰 시 발생하는 이해관계자들간의 갭을 많이 줄일 수 있게 되었다.



〈스프린트 계획 모습〉

스프린트 리뷰도 처음에는 애자일에 대한 이해관계자들의 인식부족으로 적극적인 참여가 부족하였으나 제품 책임자 역할 수립과 애자일에 대한 인식확산으로 나중에는 적극적으로 참여하게 되었다.

### • 데일리 스탠드업 미팅 및 회고 활동의 부실

데일리 스탠드업 미팅이 처음에는 개발 구성원이 모두 참여하지 않은 상태에서 진행되었으나 나중에는 디자이너를 포함한 프로젝트와 연관된 인력 등이 모두 참여한 상태에서 활발한 업무 조율과 협력이 이루어졌다. 회고 역시 처음에는 개발 구성원이 모두 참여하지 않은 상태에서 진행되고 전문적인 스킬이 부족하여 실질적인 프로세스 개선이 이루어지지 않았다. 하지만 애자일 코칭을 통하여 팀 리더를 포함한 전체 인력이 참여하여 좀더 효과적으로 진행되었다.

## 5) 현재 애자일 적용 상황

현재는 연구소에서 솔루션을 개발하는 2개팀과 신 제품을 개발하는 1개팀, 총 3개팀에서 애자일을 활용하여 제품을 개발하고 있다. 팀 마다 성향이 다르다 보니 제품백로그를 관리하는 도구나 애자일 활동은 다소 다르게 진행하고 있다. 예를 들어 어떤 팀은 레드마인(Redmine)을 이용하여 제품백로그를 관리하고 있으며 어떤 팀은 엑셀을 사용하여 제품백로그를 관리하고 있다. 개발팀 리더들은 대부분 시니어 엔지니어 출신이며 팀 관리와 함께 제품책임자 역할을 맡고 있다. 전체적으로 아래와 같은 프로세스가 공통적으로 진행되고 있다.

### • 릴리즈 계획 및 제품 백로그 정제 미팅

팀 리더는 개발팀원들과 6~8 개월 정도의 릴리즈 계획을 수립하고 스프린트 단위로 제품백로그 정제 미팅을 수행하고 있다. 아래 그림은 스프린트 별로 수행 할 주요 기능(Future)들을 기술한 릴리즈 계획 샘플이다.

| 분류  | ID             | 업무 범주 | Sub Epic      | 요구사항(스토리)                        | 상세 내용                                                                                                                                                                                                                                                                                                           | 테스트 방법                                                                                                            |
|-----|----------------|-------|---------------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| 기능  | PB_F_02_08_07  | GW    | ThingStar G/W | Metric 데이터 전송 크기 축소              | <ul style="list-style-type: none"> <li>자동 등록 기능을 위해 성능수집 시 Metric데이터 마다 리소스 메타정보를 매번 전송하는 비효율성 개선 필요</li> <li>Centric에 리소스 자동 등록 후 리소스 ID를 Gateway에 전달하는 기능 개발</li> <li>이후 Gateway에서 전송하는 Metric 데이터는 부가적인 리소스 메타정보 대신 리소스 ID를 이용하여 송신하는 기능 개발</li> <li>결과적으로 Metric 데이터 전송 크기를 축소 할 수 있는 기능 개발이 목적임</li> </ul> | - Kafka Topic에 쌓인 metric 데이터 크기 전후                                                                                |
| 비기능 | PB_NF_02_13_04 | GW    | 자체 센서 개발      | 자체센서 연동 규격 정의 및 구현               | <ul style="list-style-type: none"> <li>자체제작 센서와 GW간 MQTT로 주고받을 연동데이터 정의</li> <li>자체제작 센서연동 Plugin 및 Provider 개발</li> </ul>                                                                                                                                                                                      | <ul style="list-style-type: none"> <li>연동규격 정의 문서 리뷰</li> <li>연동데이터 화면 확인</li> </ul>                              |
| 기능  | PB_F_02_13_05  | GW    | 자체 센서 개발      | 스마트 빌딩용 자체센서 제작 및 Embedded SW 개발 | <ul style="list-style-type: none"> <li>환경센서 (온도,습도,조도) * 3 개발 및 설치</li> <li>환경센서 (온도,습도,조도,미세먼지) *1 개발 및 설치</li> </ul>                                                                                                                                                                                          | <ul style="list-style-type: none"> <li>환경 센서 노드 H/W 정상 동작 확인</li> <li>AP를 통한 데이터 조회 (온도, 습도, 조도, 미세먼지)</li> </ul> |
| 기능  | PB_F_02_13_06  | GW    | 자체 센서 개발      | 자체 환경 센서 노드 연동                   | <ul style="list-style-type: none"> <li>자체 환경센서와 AP간 MQTT JSON 타입으로 데이터 송수신 기능 구현 (상세 지표 : 온도, 습도, 조도)</li> </ul>                                                                                                                                                                                                | - Thingstar에서 데이터 확인                                                                                              |
| 기능  | PB_F_02_15_06  | GW    | 스마트 빌딩        | 사무실종합현황 근태 위젯 구현                 | <ul style="list-style-type: none"> <li>사무실 종합 현황 페이지 구성 재설계</li> <li>실내 환경 상세 페이지 재설계</li> <li>CCTV 화면 개선 도출</li> </ul>                                                                                                                                                                                         | - 대시보드 저작물을 이용한 리뷰                                                                                                |

## 〈릴리즈 계획〉

### • 스프린트 계획

제품들이 다양하고 복잡한 기능을 가진 솔루션이다 보니 3주 단위로 스프린트를 진행하고 있다. 어떤 팀에 따라 통합테스트를 1주 정도 진행 후 릴리즈를 수행한다.

연구소의 경우 3주의 개발 및 1주의 통합테스트가 반복되면서 진행되고 통상 1달 단위로 릴리즈가 수행된다. 스프린트 계획 수립 시에는 팀원들이 모두 참여하여 제품백로그를 갱신하며 팀 내 파트별로 스토리를 수행하기 위한 타스크 도출과 투입공수를 추정한다.

| 유니트 | ID             | 업무<br>범주 | Sub Epic      | 요구사항(스토리)                        | 상세 내용                                                                                                                                                                                                                                               | 대요 방법                                                         |
|-----|----------------|----------|---------------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| 기능  | P8_F_02_08_07  | GW       | ThingStar G/W | Metric 데이터 전송 크기 축소              | - 자동 등록 기능을 위해 성능수집 시 Metric데이터 마다 리소스 메타정보를 매번 전송하는 비효율성 개선 필요<br>- Centric에 리소스 자동 등록 후 리소스 ID를 Gateway에 전달하는 기능 개발<br>- 이후 Gateway에서 전송하는 Metric 데이터는 부가적인 리소스 메타정보 대신 리소스 ID를 이용하여 송신하는 기능 개발<br>- 결과적으로 Metric 데이터 전송 크기를 축소 할 수 있는 기능 개발이 목적임 | - Kafka Topic에 쌓인 metric 데이터 크기 전후                            |
| 비기능 | P8_NF_02_13_04 | GW       | 자체 센서 개발      | 자체센서 연동 규격 정의 및 구현               | - 자체제작 센서와 GW간 MQTT로 주고받을 연동데이터 정의<br>- 자체제작 센서연동 Plugin 및 Provider 개발                                                                                                                                                                              | - 연동규격 정의 문서 리뷰<br>- 연동데이터 화면 확인                              |
| 기능  | P8_F_02_13_05  | GW       | 자체 센서 개발      | 스마트 빌딩용 자체센서 제작 및 Embedded SW 개발 | - 환경센서 (온도,습도,조도) * 3 개발 및 설치<br>- 환경센서 (온도,습도,조도,미세먼지) *1 개발 및 설치                                                                                                                                                                                  | - 환경 센서 노드 H/W 정상 동작 확인<br>- AP를 통한 데이터 조회 (온도, 습도, 조도, 미세먼지) |
| 기능  | P8_F_02_13_06  | GW       | 자체 센서 개발      | 자체 환경 센서 노드 연동                   | - 자체 환경센서와 AP간 MQTT JSON 타입으로 데이터 송수신 기능 구현 (상세 지표: 온도, 습도, 조도)                                                                                                                                                                                     | - Thingstar에서 데이터 확인                                          |
| 기능  | P8_F_02_15_06  | GW       | 스마트 빌딩        | 사무실종합현황 근태 위젯 구현                 | - 사무실 종합 현황 페이지 구성 재설계<br>- 실내 환경 상세 페이지 재설계<br>- CCTV 화면 개선 도출                                                                                                                                                                                     | - 대시보드 저작물을 이용한 리뷰                                            |

### 〈스프린트 수행할 제품 백로그 상세 내용 샘플〉

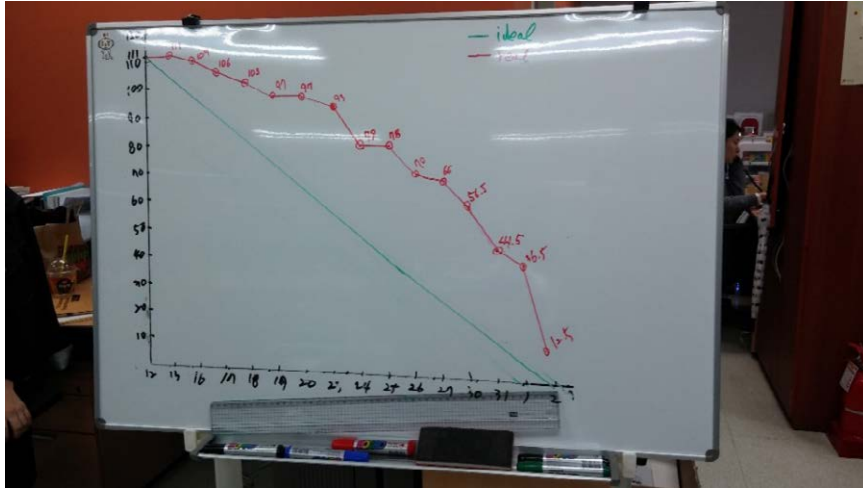
상세 태스크들은 포스트 잇을 사용하여 아래 그림과 같은 ‘작업 진행 상황판’에 붙여진다. 상황판의 상단은 제품백로그, 계획, 진행, 완료로 구성되어 있으며 해당 스프린트에서 수행할 스토리들은 ‘제품백로그’ 열에 놓인다. 스프린트 계획 시 상세한 개발태스크들로 분할되며 이렇게 분할된 태스크들은 ‘계획’ 열에 놓인다. 이후에는 데일리 스탠드업 미팅 시 개발자들에 의해 매일같이 진행, 완료로 이동된다.



### 〈작업 진행 상황판〉

- 데일리 스탠드업 미팅 및 번다운 차트(Burn down chart)

각 팀들은 매일 아침 15 ~20분 정도 모여서 업무 조율과 협력을 위한 미팅을 수행한다. 일부 팀은 아래와 같이 번다운 차트를 작성하여 팀원들이 전체 진도율을 파악할 수 있도록 한다.



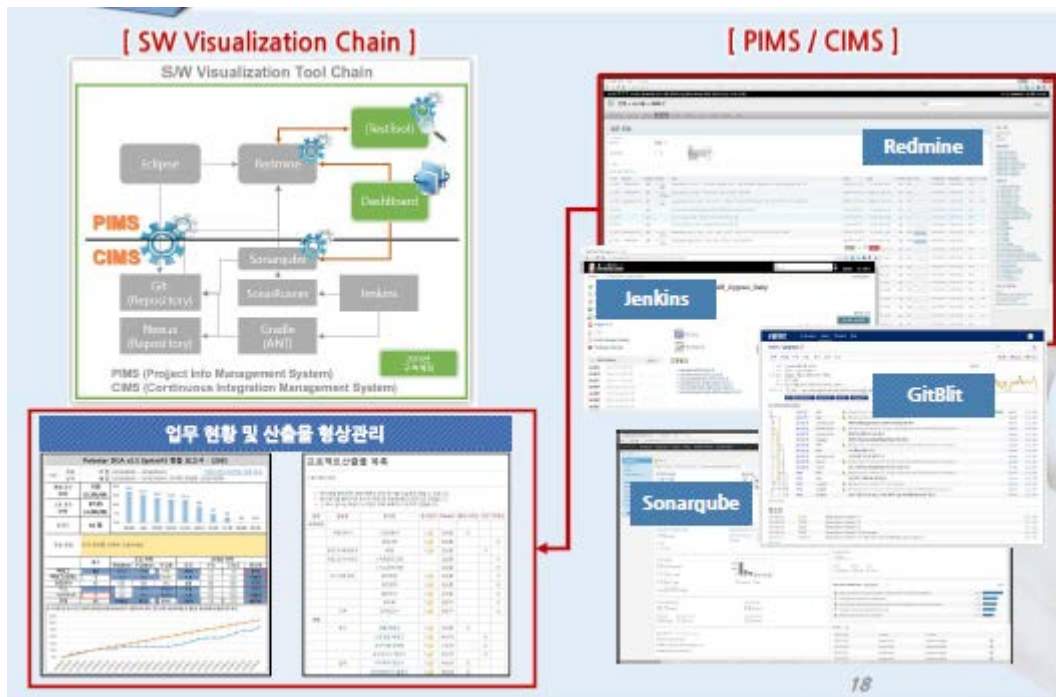
### 〈스프린트 번다운(Burn down) 차트〉

- 스프린트 리뷰 및 회고

각 팀들은 스프린트가 끝나면 이해관계자들이 참여한 상태에서 리뷰 미팅을 진행하고 있으며 팀에 따라 경영진을 대상으로 한 리뷰 미팅도 별도로 진행한다. 신제품 개발팀 같은 경우에는 리뷰 시에 팀원들간에 활발한 토론이 이루어지고 있다. 회고는 스프린트 리뷰가 종료된 후에 1~2시간 정도 진행이 된다.

- 지속적인 통합 구현

J-unit과 Jenkins, Sonacube를 활용하여 정적분석과 자동화된 단위테스트와 빌드 체계를 구축하여 운영하고 있다.



### 〈Tool Chain〉

## 6) 애자일 적용 효과

애자일 적용으로 인한 효과에 대해서 연구소장은 이렇게 얘기하고 있다. ‘과거에는 제품을 릴리즈하는 데 3개월에 1번 정도 할 수 있었는데 지금은 1개월단위로 할 수 있게 되었다. 스프린트 단위로 각 팀들이 개발 목표를 명확히 수립함으로써 요구사항에 대한 우선순위 관리가 수월하고 이를 모두 공유할 수 있게 된 것이다. 앞으로 더욱 릴리즈 주기를 당기려고 노력 중이다.’ 이 외에도 다음과 같은 점에서 효과가 있었다고 구성원들은 얘기한다.

- 프로젝트 진행상황을 투명하게 볼 수 있다.
- 과거에는 개인이 혼자 일하는 형태다 보니 누락된 기능들이 많이 발생하였는데 지금은 다 함께 계획을 세워 추진하다 보니 그런 일들이 많이 줄었다.
- 과거에는 사전 계획 없이 개인 플레이로 일하다 보니 리더 입장에서 아웃풋이 마음에 들지 않는 경우가 많았으나 지금은 많이 좋아졌다.
- 다양하게 발생하는 요구사항에 대한 우선순위 관리가 수월해졌다.
- 이해관계자들이 제품리뷰에 주기적으로 참여하고 피드백을 받을 수 있다.
- 과거에는 팀원들에게 일방적으로 일정이 주어졌는데 지금은 팀원들이 주도적으로 공수와 일정을 수립하게 되어 개발자들의 만족도가 높아졌다.

## 5.2.2 (주)엔비티 사례

### 1) 회사 개요

이 회사는 2012년 설립된 스타트업이다. 국내에 선보인 세계 최초 모바일 잠금화면 플랫폼 '캐시슬라이드(cashslide)'는 2017년 현재 1,800만 명 이상의 사용자를 확보하고 있으며, 국내의 성공을 바탕으로 해외 시장에도 적극 진출하여 중국에서 '쿠화(coohua)'도 성공 궤도에 안착시켰다. 2017년 현재 임직원 100여명의 회사로 성장했다.

### 2) 애자일 도입 배경

애자일 방식은 빠르게 변하는 모바일 앱 시장에 사업적으로 유연하게 대처하기 위하여 2013년에 개발 조직에서 도입하였다. 도입 초기에는 애자일 코치에 적합한 사내 인력을 선발하여 전환 배치하였으며, 이후 경험 있는 애자일 전문가를 채용 배치하며 애자일 방식을 성숙시켰다.

### 3) 애자일 적용 상황

이 회사는 전사적으로 애자일 방식을 사용하여 제품을 개발하고 있다. 3개의 제품개발실과 1개의 연구실이 있으며 각 실은 사업의 특성에 따라 애자일 방식을 다르게 사용한다. B2B와 기술 연구를 다루는 경우 칸반을 주로 사용하고, B2C의 경우 스크럼/XP를 하이브리드하여 주로 사용한다. 주력 사업인 캐시슬라이드 제품개발실은 3개의 개발팀으로 구성되어 있다. 이 중 2개의 팀은 신규 개발 과제를 주로 담당하고, 1개의 팀은 개선 및 운영 개발 과제를 주로 담당한다. 신규 개발 과제를 담당하는 2개의 팀은 전담 애자일 코치와 함께 프론트엔드 앱 및 백엔드 서버 개발자, 제품 디자이너 등 6~7명으로 구성되어 있는 기능 혼합(Cross-functional) 팀이다.

팀의 애자일 코치는 개발팀이 담당한 개발 과제가 성공적으로 완료될 수 있도록 전반적인 개발 활동을 조율하는 엔지니어링 매니저 역할을 한다. 더불어 각종 계획, 일간 스탠드업, 회고, 리뷰 등의 활동을 퍼실리테이션 하면서 개발 팀이 자기조직화 할 수 있도록 코칭하고 이끈다.

개선 및 운영 개발 과제를 주로 담당하고 있는 개발팀은, 전담 애자일 코치가 있지 않으며, 제품의 아키텍트가 엔지니어링 매니저 역할을 한다. 주로 담당하는 과제의 특성상 칸반을 주로 사용한다. 반면 신규 개발 과제를 주로 담당하는 개발팀은 스크럼을 중심으로 관련 방법들을 하이브리드한 방식을 주로 사용한다.

캐시슬라이드 제품개발실에는 제품개발실 소속으로 1명과 각 개발팀 소속 2명, 총 3명의 애자일 코치가 있다. 캐시슬라이드 제품개발실 외 다른 제품개발실에는 애자일 코치가 없으나, 필요시 요청을 하면 해당 제품개발실은 애자일 코칭을 지원 받을 수 있다. 제품 책임자는 제품개발실 별로 1명씩 배치된다.

아래 활동들은 캐시슬라이드 제품개발실에서 진행하는 활동이다.

#### • 제품/파티 백로그 정제(Refinement) 미팅

매월 초 제품 책임자가 제품개발실 인원 전체와 함께 수행할 과제를 논의하는 미팅이다. 제품백로그 정제 미

팅에서 제품 책임자는 경영진과 합의된 과제를 제품개발실에 설명한다. 제품개발실의 각 개발팀들은 과제에 대하여 논의하고 담당할 과제를 직접 선정하고 타 개발팀과 협상한 후 제품 책임자와 확정한다. 확정된 과제들은 개발팀 별로 후속 진행되는 파티 백로그 정제 미팅에서 제품 책임자를 통해 구체적으로 설명된다. (엔비타는 제품개발실 소속 개발팀을 파티라고 부른다.)



〈파티별 백로그 계획 미팅〉

#### • 파티 과제 계획(Planning) 미팅

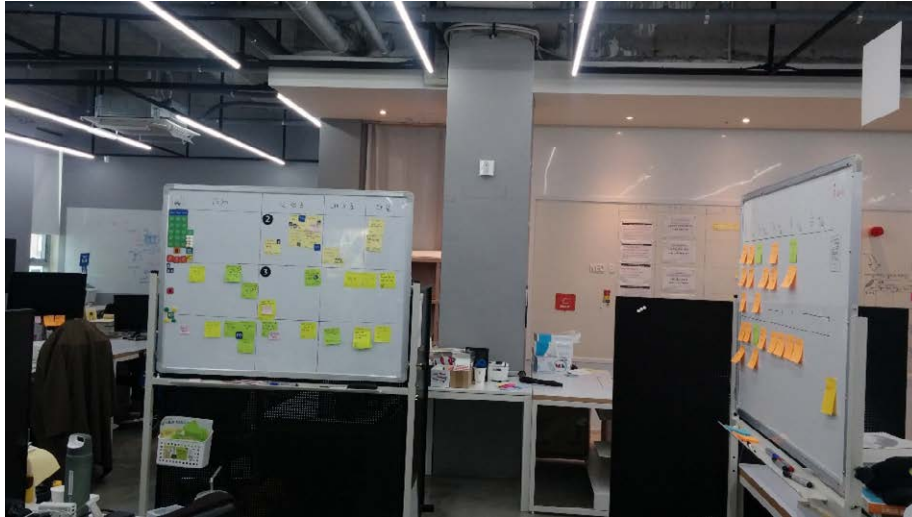
과제가 새로 시작할 때 개발팀 인원 전체 그리고 제품 아키텍트와 함께 과제의 일정 계획을 수립하는 미팅이다. 과제의 불확실 정도에 따라 계획에 사용하는 방법이 달라진다. 과제의 완료 시점까지의 작업들에 대해 예측이 가능하다면 타스크를 모두 도출하여 공수를 추정하고 전체 일정 계획을 확정한다. 이러한 방법을 크리티컬 패스(Critical Path) 스케줄링이라고 부른다. 과제가 어떻게 진행될지 예상하기 어렵다면 1주 단위의 계획을 수립하고 1일 단위로 갱신해간다. 이러한 방법을 타임박스(Timebox) 스케줄링이라고 부른다.



〈파티별 백로그 계획 미팅〉

### • 데일리 스탠드업 미팅

엔비티의 모든 개발팀은 매일 아침 소속 개발팀의 칸반보드 또는 태스크 보드 앞에 모여 미팅을 하고 하루를 시작한다.



〈 팀별 칸반보드/태스크보드〉

### • 제품/파티 회고

엔비티의 모든 개발팀은 매주 금요일 오후에 1시간의 회고를 한다. 제품개발실은 매월 3번째 주 금요일 오후에 2시간의 워크아웃(전체 회고)을 한다.

### • 월별 제품 리뷰

제품개발실은 매월 마지막 주 금요일 오후에 2시간의 제품 리뷰를 한다. 제품개발실 전체와 이해관계자들이 모여서 진행하고 완료된 과제와 관련 사업 지표를 함께 검토한다.

### • 분야별 학습조직

엔비티의 모든 직무자들은 매주 금요일 오후에 2시간의 직무별 학습 모임을 가진다. 프론트엔드 앱 개발 학습모임, 백엔드 서버 개발 학습모임, 제품 디자인 학습모임, 제품 관리 학습모임 그리고 애자일 학습모임이 있다.

### • 기타 사항

가치 있는 신규 제품 탐색 및 개발이 중요한 신제품개발실은 디자인적 사고를 통한 탐색과 린 스타트업을 활용하여 개발이 진행된다. 매주 금요일은 성장을 위한 자기개발, 각종 학습 모임, 리뷰, 회고 등을 진행하며 계획 수립 시 포함하지 않는다.

## 4) 애자일 적용 효과

애자일을 적용하면서 빠르고 유연한 개발문화를 가지게 되었으며, 시장의 변화에 신속하게 대처하며 더 나은 서비스를 사용자에게 제공할 수 있게 된 것이 주요 효과이다. 직원들은 ‘일반적인 회사에서보다 더 많은 의사결정권을 가지게 되면서 업무에 대한 오너십이 높아졌고, 높은 성취감을 느낄 수 있게 되었다.’ 고 얘기한다. 엔비티는 캐시슬라이드 제품 외 새로운 제품들을 지속적으로 출시하며 빠르게 성장하고 있다.

### 5.3 애자일 실패 사례

애자일 방식은 전통적 개발 패러다임과는 많은 차이가 있다 보니 처음 애자일을 접하는 조직이 성공적으로 적용하는 경우보다는 실패하는 경우가 훨씬 많은 것이 사실이다. 애자일에 대한 부정적 경험을 갖고 있는 일부 개발자들은 거부감도 상당하다. 많은 개발팀들이 애자일을 쉽게 생각하고 도입했다가 별다른 효과도 보지 못하면서 개발자들에게 업무량만 늘리는 경우가 많았기 때문이다.

스크럼 프랙티스들이 이해하기 쉽고 적용이 쉬울 것 같지만 막상 적용하려고 하면 많은 현실의 벽에 부딪치게 되는 것이 일반적이다. 애자일에 대한 철학과 원리를 제대로 이해하지 않고 단순히 점진적/반복적 프로세스나 특정 개발 기법을 도입하게 되면 별다른 효과를 못 느끼거나 조직원들의 저항만 불러일으키기 쉽다. 본 연구에서는 소프트웨어 관련 커뮤니티 및 컨퍼런스에서 많이 얘기되는 실패사례들을 몇 가지 패턴으로 분석해 보았다.

#### • 데일리 스탠드업 미팅의 오용

데일리 스탠드업 미팅은 애자일을 처음 접하는 조직에서 가장 쉬워 보이는 애자일 프랙티스로 인식하는 경우가 많다. 왜냐하면 과거에도 이런 미팅을 아침마다 한 적이 있기 때문이다. 그래서 많은 리더들이 전통적인 사고방식을 가지고 데일리 스탠드업 미팅을 수행한다. 즉, 아침에 팀원들로부터 업무 상황을 보고 받고 필요한 지시를 내리는 형태의 미팅 방식이다. 하지만 애자일에서도 이렇게 진행하면 개발자들로부터 심한 저항을 받으면서 애자일 도입은 처음부터 실패하게 된다.

애자일에서 수행하는 데일리 스탠드업 미팅은 과거 미팅의 성격과는 근본적으로 다르다. 팀원들이 중심이 되어 수평적으로 대화하면서 업무를 조율하고 협력하는 미팅이다. 팀 리더는 팀원들이 겪고 있는 장애요인이나 이슈를 해결하는 데 초점을 맞추어야 하며 가급적 지시보다는 팀원들이 자기 주도적으로 협력하여 일할 수 있도록 유도해야 한다. 데일리 스탠드업 미팅은 자칫 개발자들을 매일 같이 압박한다는 느낌을 가질 수 있기 때문에 리더는 개발자들이 그런 압박감을 느끼지 않도록 신경을 많이 써야 한다. 정상적인 궤도에 오르기까지는 1~2달이 걸릴 수가 있다.

#### • 팀 책임의 업무 수행 방식 부족

기존에 팀원들은 대부분 할당된 업무만 수행하는 개인 책임의 업무 수행방식이었다. 문제가 생기면 해당 업무를 수행한 개인이 책임을 지는 구조다. 그러다 보니 자기 업무에만 관심을 가지게 되고 다른 사람의 업무에 관심을 가지는 일이 적고 도와줄 여력도 거의 없었다. 하지만 애자일을 적용하면 팀 책임으로 일하는 방식에 익숙해져야 한다. 즉, 자기 일만 잘해서는 안되며 팀의 일에 대하여 공동의 책임감을 느껴야 한다. 그렇게 되면 다른 사람의 업무에 자연스럽게 관심을 가지게 되고 필요 시 돕게 된다. 애자일 적용 시 팀원들이 이런 사고방식의 전환이 이루어지지 않으면 협력으로 인한 시너지 효과가 전혀 나타나지 않는다. 이런 상태에서는 데일리 스탠드업 미팅도 지속되지 못한다. 서로가 협력하지 않는 상태에서 업무 공유는 무의미하기 때문이다. 리더는 개인적인 미팅을 통하여 팀원들이 사고방식의 전환이 일어날 수 있도록 지원해야 한다.

- 애자일 개발문화를 제외한 프로세스와 도구 중심의 애자일 적용

애자일 도입 시 성공과 실패를 결정하는 요인 중 도구가 차지하는 비율은 10% 미만이며 개발 문화가 차지하는 비중이 90% 이상이다. (22) 해외에서 애자일이 혁신을 위한 방법론으로 자리잡은 것은 프로세스 때문이 아니라 애자일이 추구하는 사고방식의 변화 때문이다. 그런데 많은 국내 조직에서 이런 부분은 현실적으로 어렵다 보니 단순히 애자일 프랙티스와 이를 지원하는 도구만 도입하는 경우가 많다. 단순히 애자일 프랙티스와 특정 애자일 도구를 쓴다고 해서 해당 조직이 애자일 조직이라고 얘기할 수는 없는 이유가 여기에 있다. 물론 특정 도구나 프랙티스가 도움이 될 수 있지만 애자일의 효과를 온전히 누리기에는 어렵다.

- 탑 다운 형태의 애자일 확산

애자일에 대한 효과가 많은 사례를 통해서 증명되다 보니 일부 경영진들은 애자일을 조직 내에 빨리 도입해서 성과를 얻고 싶은 충동이 생기게 된다. 그래서 과거에 했던 방식처럼 파일럿으로 1~2개를 수행해보고 이를 표준화해서 다른 모든 팀에 확대 적용하는 경우가 많다. 문제는 내부에 애자일 스킬과 역량이 부족한 상태인데도 서둘러 확산하는 경우이다. 보통 애자일을 처음 적용할 때는 시행착오가 많이 발생하기 때문에 노련한 애자일 코치가 있으면 도움이 된다. 만약 이런 코치의 도움 없이 애자일을 적용한다면 전담 인력을 지정해서 차근차근 진행하는 것이 바람직하다. 애자일이 퍼지는 가장 좋은 방법은 작은 성공스토리를 만들어 입 소문이 나게 하는 것이다.

## CHAPTER 06

### 국내 소프트웨어 기업의 애자일 적용 한계점 분석



- 6.1 애자일 적용상의 어려움과 한계점 54
- 6.2 애자일 활성화를 위한 제언 55

## CHAPTER 06

## 국내 소프트웨어 기업의 애자일 적용 한계점 분석

## 6.1 애자일 적용상의 어려움과 한계점

소프트웨어 개발 프로세스의 변화는 단순히 도구와 기술만 바꾼다고 해서 변화될 수는 없으며 조직의 구조와 문화, 관리 방식 등의 변화가 수반되어야 한다.(참고문헌 5,24) 애자일은 기존에 존재했던 방법론에 비하여 훨씬 혁신적이다 보니 조직 전반적인 변화가 불가피하다. 2016년 전세계 비즈니스 및 IT 경영진 1770명을 대상으로 조사되었던 자료에 따르면 애자일 방법론의 활용을 가로막는 주요 원인으로 한국 기업들로부터는 다음과 같은 것들이 나타났다.(참고문헌 1)

1. 애자일과 맞지 않는 조직 문화(58%)
2. 사내 기술 및 지식 부족(53%)
3. 적절한 기술 및 도구 통합의 어려움 (52%)
4. 보안문제 (48%)
5. 경영진 개입 및 이해 부족(43%)

최근에 국내에서 소프트웨어 커뮤니티를 대상으로 애자일 도입시의 어려움을 조사한 설문 자료를 살펴보면 경영진과 동일하게 '애자일 철학과 맞지 않는 조직 문화'를 첫 번째 장애요인으로 지적하고 있다. 애자일의 핵심 철학은 팀원에 대한 존중과 신뢰, 수평적인 소통과 협력 문화 라고 할 수 있는데 전통적인 국내 기업들은 대부분 명령과 통제, 개인 중심으로 일하는 업무방식 방식을 가지고 있다 보니 충돌되는 것이다. 아래 내용은 설문조사에 나타난 결과이다. (부록참조)

1. 애자일 철학과 맞지 않는 조직 문화(56%)
2. 경영진의 의지 및 지원 부족(51%)
3. 애자일 관련된 경험 및 스킬 부족(44%)
4. 국내 현실을 반영한 적용 가이드 부족(44%)
5. 구성원들의 새로운 변화에 대한 저항(28%)
6. 불충분한 교육(30%)

두 번째로 지적된 경영진의 의지 및 지원 부족은 애자일 적용을 조직 차원의 변화가 아닌 단순히 팀 차원의 변화로만 받아들이다 보니 생긴 현상이라고 할 수 있다. 경영진들이 애자일을 비즈니스 성과 향상을 위한 강력한 경영기법으로 인식하지 않고 단순히 팀의 기술적 개선 정도로만 접근하는 것이다. 이는 경영진들이 애자일 철학을 제대로 이해하지 못하는 데 기인한다. 애자일이 조직 내에서 성과를 내기 위해서는 그에 걸 맞는 조직 환경 구축, 인사 및 재무 관리 방식의 변화, 애자일 교육이나 컨설팅 제공 등 경영진이 지원해주어야 할 업무가

많이 필요하다.(참고문헌 23)

세 번째로 지적된 애자일 관련된 경험 및 스킬 부족은 애자일 적용 시 나타나는 시행착오를 극복하기가 쉽지 않기 때문에 발생한다. 스크럼이나 칸반 같은 경우 프레임워크가 비교적 단순해서 적용하기 쉬워 보이지만 막상 현실에 적용하게 되면 경우의 수가 많아서 여러 가지 이슈들이 많이 발생하게 된다. 또한 애자일이 기존의 개발 패러다임과는 많이 다르다 보니 현실적인 장애에 부딪치는 경우도 많다. 이런 장애요소들을 극복하기 위해서는 애자일에 대한 경험과 스킬이 있는 필요한데 이런 능력을 단기간에 획득하기는 불가능하다.

네 번째로 지적된 국내 현실을 반영한 적용 가이드 부족은 앞서 경영진으로부터 조사되었던 '적절한 기술 및 도구 통합의 어려움'과 일맥상통한다. 기존에 존재하는 애자일 관련 가이드나 책들이 다소 추상적이고 해외 사례들이 대부분이다 보니 우리 현실에 맞는 애자일 적용 방법을 찾기가 쉽지 않은 것이 현실이다. 또한 애자일에 대한 성공사례 같은 것이 공유되지 않다 보니 어떻게 시작해야 할지도 모르는 경우가 많다. 애자일을 성공적으로 도입하기 위해서는 시행착오를 줄일 수 있도록 적절한 프랙티스 및 도구에 대한 가이드가 필요하다.

다섯 번째로 지적된 구성원들의 새로운 변화에 대한 저항은 익숙한 방식이나 고정관념을 버리지 못하는 데서 발생한다. 사실 우리가 가지고 있는 기존의 소프트웨어 개발 및 관리 방식은 20~30년이 넘게 유지되어온 오래된 관행이다. 이런 방식을 단기간에 변화시키기는 불가능하다. 변화에 저항하는 사람 중에 상당수는 과거 애자일과 관련하여 안 좋은 경험들을 가지고 있거나 애자일에 대한 잘못된 인식을 가지고 있는 경우가 많다.

여섯 번째로 지적된 불충분한 교육은 경영진의 지원부족과 연계되어 있다. 애자일 도입에 대한 경영진의 관심이 저조하다 보니 직원들에게 애자일에 대한 교육이나 멘토링 같은 투자를 하지 않는 것이다. 불충분한 교육은 애자일에 대한 구성원들의 인식을 부실하게 하여 결국은 저항을 키우는 결과를 낳게 되는 경우가 많다.

## 6.2 애자일 활성화를 위한 제언

애자일 적용상의 어려움과 장애요인으로 인하여 국내 기업에서 애자일을 성공적으로 정착시킨 곳은 일부 스타트업을 제외하고서는 별로 없다. 우리나라 IT기업이 지난 30년 동안 괄목할 만한 성장을 해온 것은 사실이나 과거의 성공방정식이 지금의 성공방정식이 될 수 없는 것처럼 애자일 적용에 어려움이 있다 하더라도 이를 극복해야 한 단계 도약할 수 있을 것이다. 이런 관점에서 국내 중소기업이 애자일을 성공적으로 도입하여 활용할 수 있는 방안에 대하여 제시한다.

### 1) 국내 중소 기업 상황을 반영한 애자일 성공 사례와 상세 가이드가 필요하다.

앞 절에서도 언급했지만 기존에 존재하는 애자일 관련 가이드나 책들이 다소 추상적이고 국내 현실을 반영하지 못하다 보니 현업에서 실제 참조할 가이드가 부족한 것이 현실이다. 따라서 국내 중소기업의 현실을 반영하고 국내 애자일 적용 사례 등이 담긴 가이드가 만들어지면 애자일 활성화에 도움이 될 수 있다. 가이드 구성 방향은 애자일 전문가를 확보할 여력이 없는 소프트웨어 중소기업 입장에서 애자일을 효과적으로 활용할 수 있는

형태의 가이드가 바람직하다. 가이드 작성시 고려할 점은 성공사례를 마치 베스트 프랙티스로 오해하지 않게끔 주의해야 한다.

## 2) 애자일 도입을 원하는 중소 소프트웨어 기업에게 최소한의 교육과 코칭이 필요하다.

5.2절에서 언급한 국내 성공사례를 보면 애자일 코치의 도움이 절대적인 것을 알 수 있다. 아무리 현실적이고 상세한 가이드를 배포한다고 해도 기존 개발 방식과는 현격하게 다른 애자일 원리와 프로세스를 실무자들이 제대로 이해하기에는 한계가 있다. 따라서 정보통신산업진흥원과 같은 공공기관에서 애자일 도입을 원하는 중소 소프트웨어 기업에게 최소한의 교육과 컨설팅을 제공한다면 성공적인 안착에 도움이 될 것이다.

## 3) 중소기업 경영자들에게 애자일 시작방법과 성공사례 등에 대한 정보를 제공한다.

앞 절에서 애자일 도입의 가장 큰 어려움은 기존의 조직 문화와 경영진의 의지 부족을 지적한 바 있다. 경영진의 의지와 참여가 있어야 비로서 조직 문화가 바뀔 수 있기 때문이다. 또한 상당수 경영진들이 애자일을 어떻게 도입해야 하는지도 잘 알지 못하는 것으로 나타났다. 많은 글로벌 기업의 경영진들은 애자일을 비즈니스 가치를 높이기 위한 경영도구로 인식하고 있기 때문에 비즈니스 변화에 대응할 수 있는 애자일 조직을 만들기 위하여 조직 구조와 문화를 바꾸는 데 많은 노력을 기울이고 있다. 따라서 중소기업 경영자들이 애자일을 적극적으로 활용할 수 있도록 애자일 도입방법과 성공사례 등에 대한 정보를 주기적으로 제공하는 것이 도움이 된다.

# 부록

## 경량(애자일)방법론 설문 조사 결과



|     |          |    |
|-----|----------|----|
| 7.1 | 설문 조사 개요 | 58 |
| 7.2 | 설문 조사 결과 | 58 |

## 부록

## 경량(애자일)방법론 설문 조사 결과

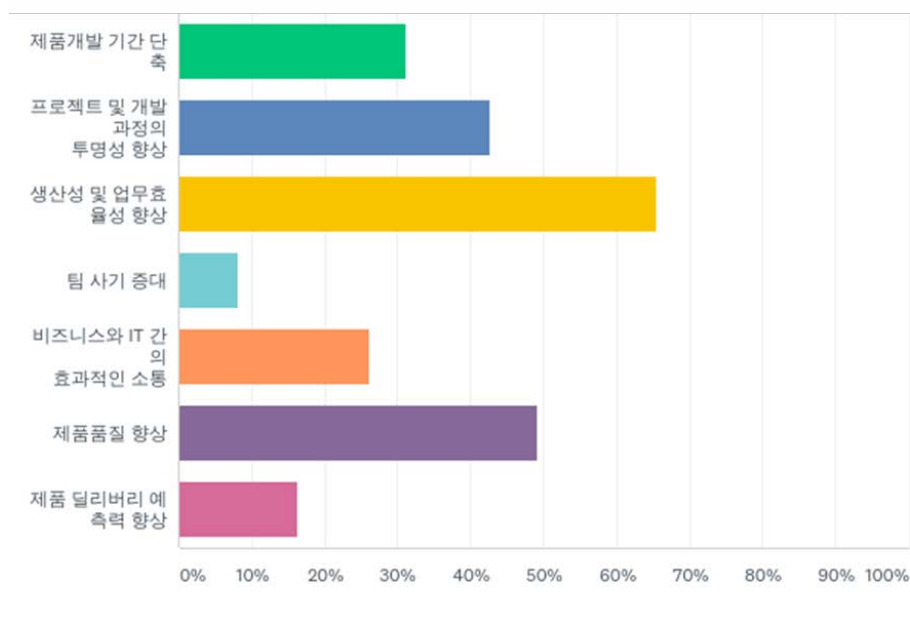
## 7.1 설문 조사 개요

- 설문 기간: 2017년 8월 11일~9월 15일
- 설문 응답자 수: 68명
- 주요 설문 내용
  - 모든 애자일 방법론의 국내 도입 현황을 묻는 질문
  - 애자일 방법론 도입 시 겪는 어려움에 대한 질문
  - 애자일 방법론의 실패 및 성공 요인을 묻는 질문
  - 응답자 배경을 묻는 질문
  - 설문 문항: 12문항

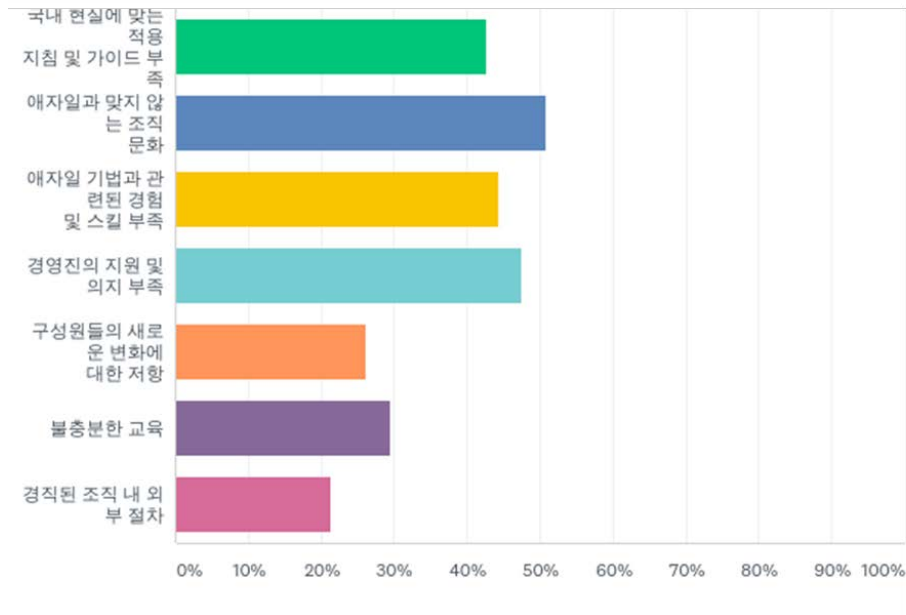
## 7.2 설문 조사 결과

Q1

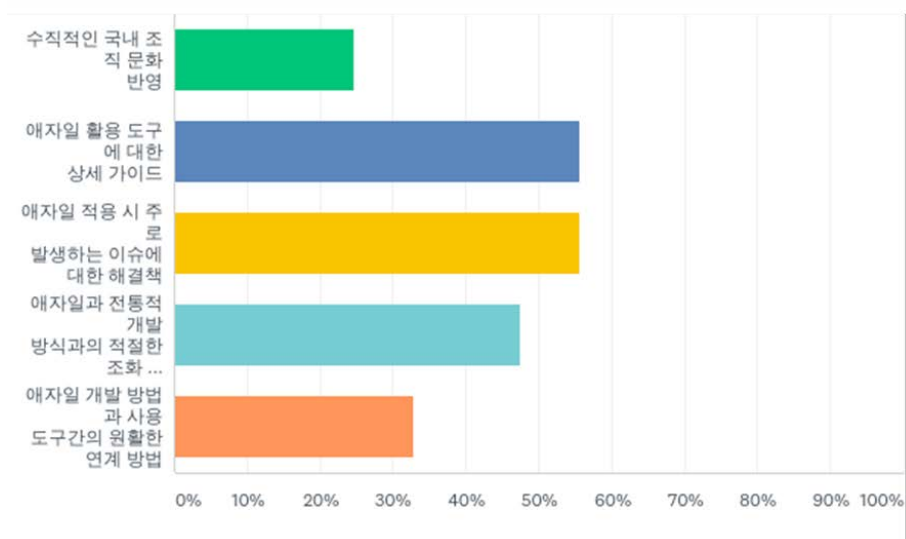
귀하가 애자일 방법론을 도입한다면 어떤 목적으로 도입하려고 하십니까?



## Q2. 귀하가 애자일 개발 방법론을 도입하려고 할 때 겪는 어려움은 무엇인가요?

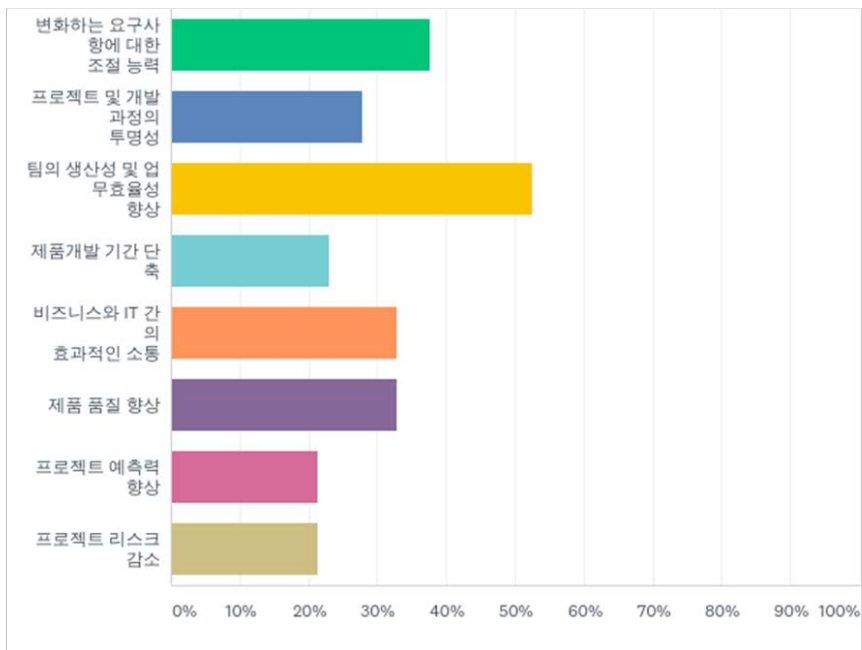


## Q3. 국내 현실을 반영한 애자일 적용가이드가 만들어진다면 어떤 내용이 들어가는 것이 좋은지요?



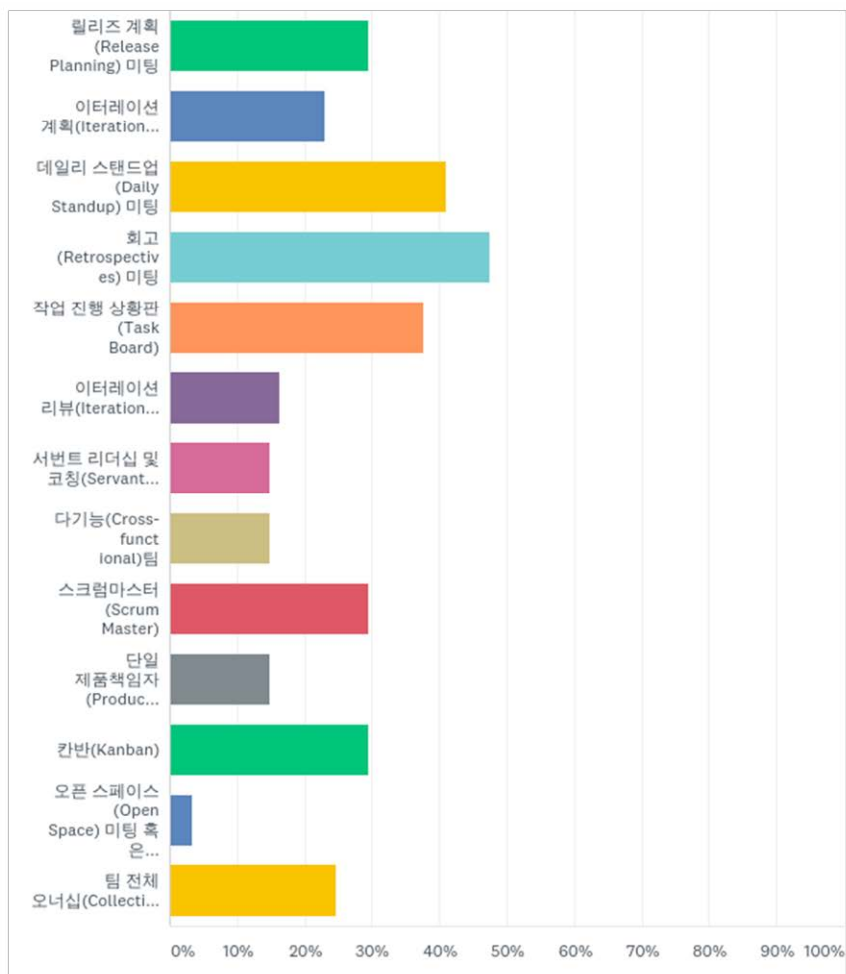
Q4

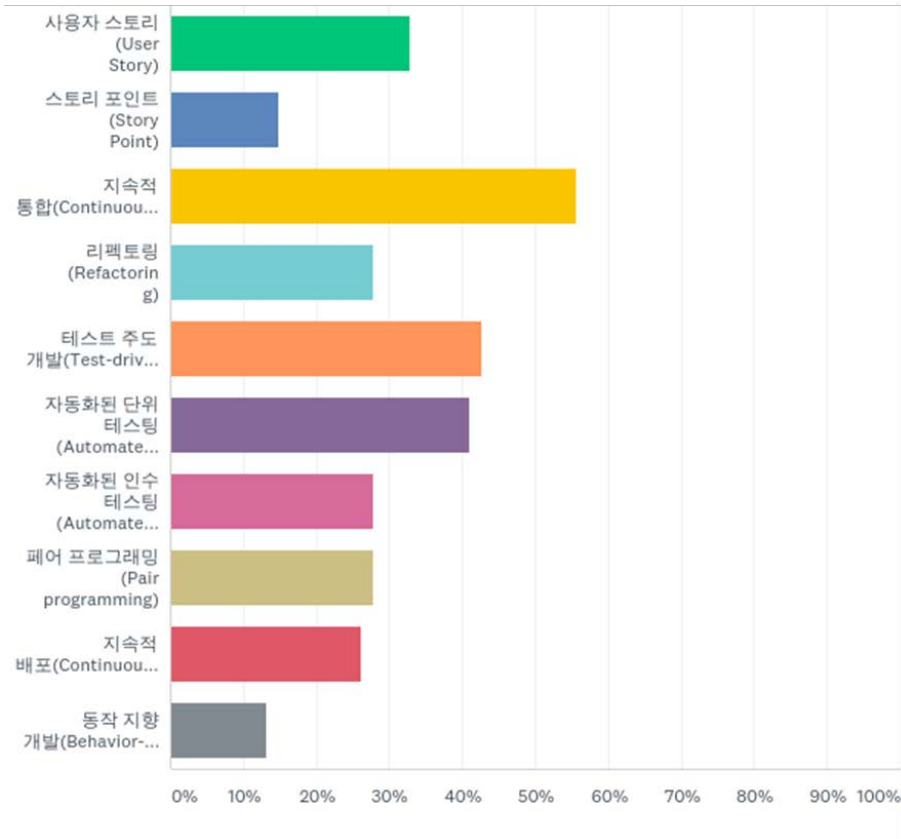
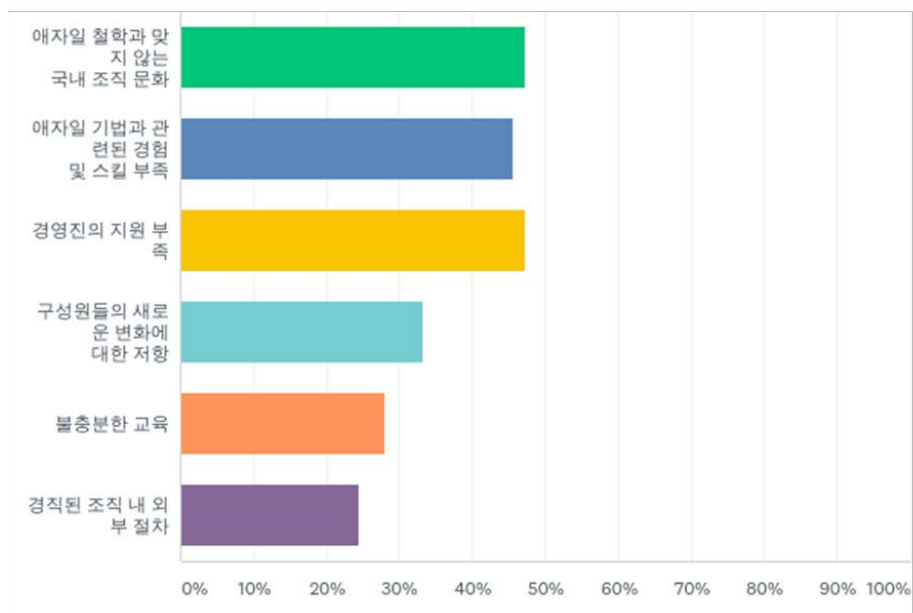
귀하가 애자일을 적용하고 있다면 애자일의 어떤 점이 도움이 되었나요?



Q5

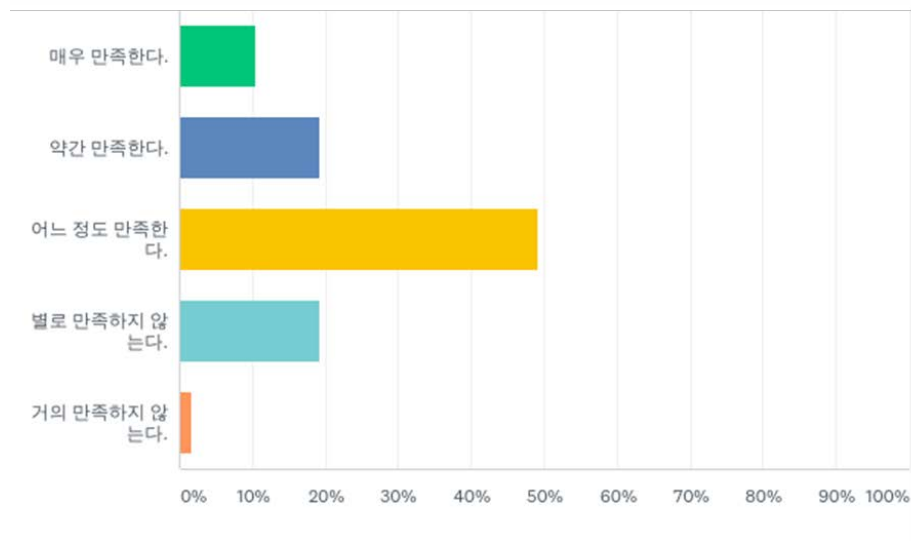
귀하가 도입한 애자일 관리기법은 어떤 것인가요?



**Q6.**
**귀하에게 도움이 되었던 애자일 기술 프랙티스는 어떤 것인가요?**

**Q7.**
**귀하가 애자일을 도입했을 때 부딪혔던 어려움은 무엇인가요?**


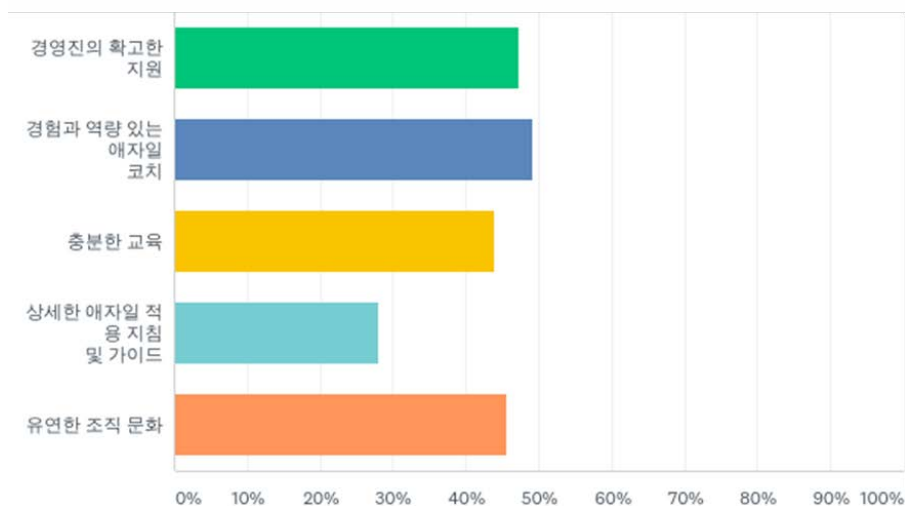
**Q8.**

**애자일 방법론 적용 결과 귀하의 만족도는 어떤가요?**

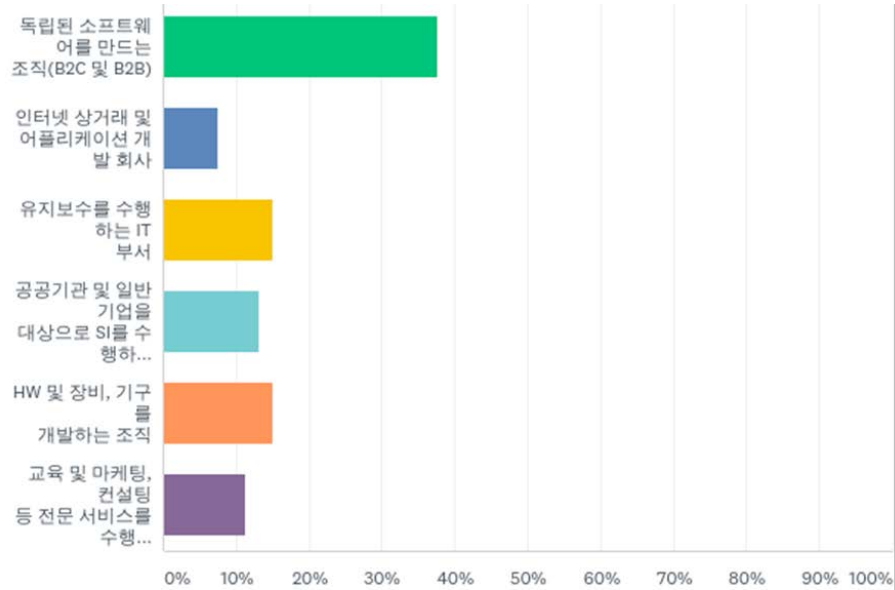


**Q9.**

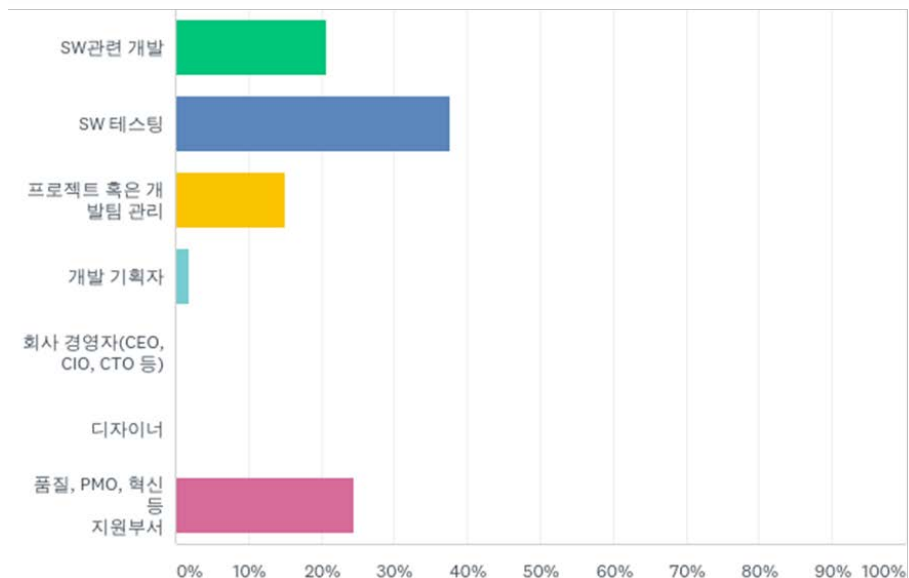
**귀하는 애자일을 도입할 때 성공요인이 무엇이라고 생각하나요?**



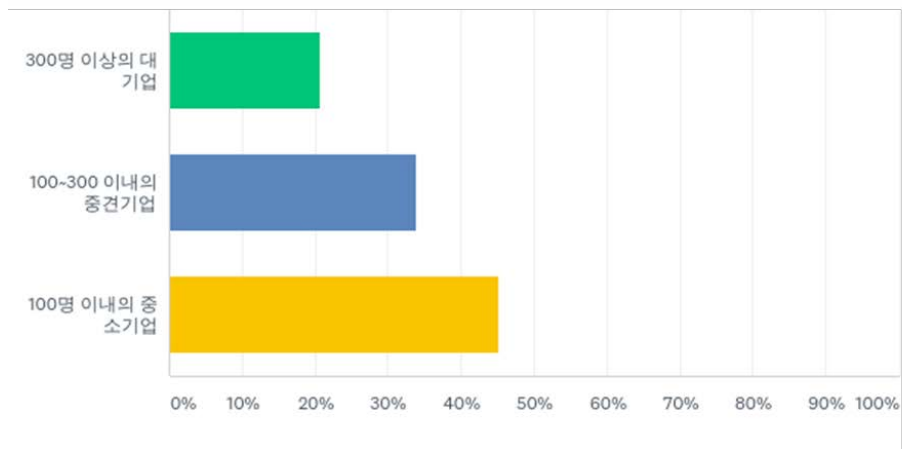
### Q10. 귀하가 일하고 있는 조직은 어떤 종류인가요?



### Q11. 조직에서 귀하의 주된 역할은 무엇인가요?



**Q12.** 귀하가 일하고 있는 회사는 규모가 어디에 속하나요?



# 참고문헌



## CHAPTER 08

## 참고문헌

1. '한국, 데브옵스·애자일 성숙도 낮다', ZDNet Korea, 2017.3.19
2. 메리 포펜딕·톰 포펜딕 지음, '린 소프트웨어 개발의 적용', 엄위상 외 2인 옮김, 위키북스, 2007, pp.86-87
3. 데이비드 J 앤더슨외 1인 지음, '에센셜 칸반 요약 가이드', 조승빈 외 2인 옮김, 2016
4. 이재왕 지음, '애자일 & 스크럼 프로젝트 관리', 길벗출판사, 2016
5. Sridhar Nerur, Radhakanta Mahapatra, George Mangalapa : Challenges of migrating to agile methodologies, Commun. Of ACM 48, 5(May 2005)
6. Daniel J. Fernandez & John D. Fernandez : Agile Project Management —Agilism versus Traditional Approaches, Journal of Computer Information Systems Winter 2008-2009
7. What is Hybrid Agile, Anyway?, <https://www.agilealliance.org/what-is-hybrid-agile-anyway/>
8. The State of Scrum report, 2017 edition, Scrum Alliance
9. Darrell K. Rigby, Jeff Sutherland, Hirotaka Takeuchi : Embracing Agile, May 2016 issue (pp.40-48, 50) of Harvard Business Review.
10. Vishnu Vinekar, Craig W. Slinkman, and Sridhar Nerur : Can agile and traditional systems development approaches coexist?, [www.ism-journal.com](http://www.ism-journal.com), 2006
11. 에릭 리스 지음, '린 스타트업', 이창수외 1인 옮김, 인사이트, 2012
12. 효과적인 데브옵스(DevOps) 적용을 위한 10가지 실행방안, 정보통신산업진흥원 동향브리핑, 제 110호, 2014
13. When-Why-How to use the Hybrid model, <https://intland.com/blog/agile/when-why-how-to-use-the-hybrid-model/>
14. 11th annual State of Agile Report, Versionone, 2017
15. Agile & DevOps Survey Report : Accelerating Velocity and Customer Value with Agile and DevOps, CA Technologies, 2016
16. SK㈜C&C, 클라우드 전환 코칭 서비스 확대, ZDNet Korea, 2017.8.8
17. 카카오의 일하는 방식, [www.kakaocorp.com](http://www.kakaocorp.com)

18. IT프로젝트 애자일 개발방식 확산, 디지털타임스, 2010.10.18
19. 블로그 : '애자일 적용을 올바르게 하기 위해서는' <http://blog.naver.com/clippe/30166572173>
20. 블로그 : '애자일. 한국에서 실패하는 이유' <http://blog.naver.com/PostView.nhn?blogId=cloud2519&logNo=220744431349>
21. MATTHEW BADGLEY : Top 10 Tips for Measuring Agile Success, <https://blog.versionone.com/top-10-tips-agile-success/>
22. 속도와 혁신의 마법, 애자일, 전사·전업무 확산 가이드, IDG Deep Dive
23. 마이크 콘 : '경험과 사례로 풀어낸 성공하는 애자일', 인사이드, 2012, PP 486-509
24. Tsun Chow, Dac-Buu Cao, A survey study of critical success factors in agile software projects, The Journal of Systems and Software 81 (2008) 961-971

