

---

# 정보기술연구소

Agile 보고서

Version 0.1

*Issue Date: 2018/10/11*

*KCC정보통신*

*신 지 영*

## 제.개정 이력

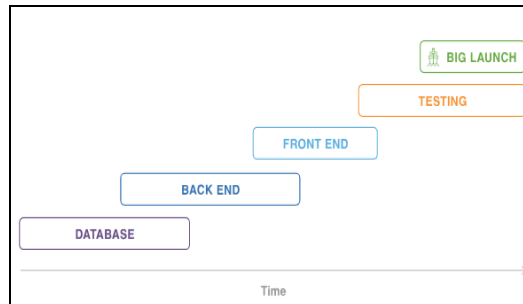
0.1	최초 작성	2018-10-11
개정번호	제.개정 페이지 및 내용	제.개정 일자

## - 목 차 -

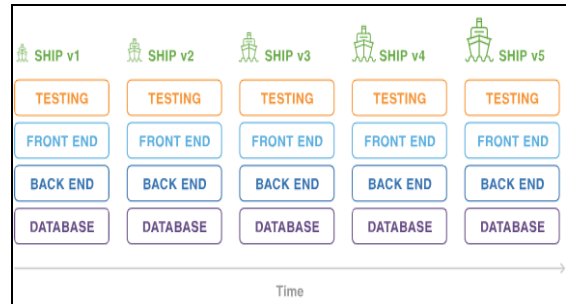
<b>1. 애자일(AGILE) 개발이란</b>	<b>4</b>
1.1. 개념	4
1.2. 등장 배경	4
1.2.1. 애자일 소프트웨어 개발 선언문(AGILE MANIFESTO, 2001)	5
1.2.2. 애자일 소프트웨어의 12가지 원칙	5
1.3. 애자일 프레임워크(AGILE FRAMEWORK)	6
1.3.1. 스크럼(SCRUM)	6
1.3.2. 칸반(KANBAN)	9
1.3.3. 스크럼과 칸반의 비교	12
1.3.4. 칸반을 통한 스크럼의 보완	12
1.4. 애자일의 장·단점	13
1.5. 전통적 및 애자일 개발의 적절한 대상	13
<b>2. JIRA의 AGILE 적용 예시</b>	<b>14</b>
2.1. JIRA AGILE BOARD	14
<b>3. REVIEW</b>	<b>18</b>
<b>4. 참조 문헌</b>	<b>19</b>
<b>5. 용어 정리</b>	<b>20</b>

## 1. 애자일(Agile) 개발이란

### 1.1. 개념



<그림 1. 폭포수(Waterfall) 방식>



<그림 2. 애자일(Agile) 방식>

‘Agile’이라는 단어는 ‘재빠른’, ‘민첩한’의 뜻을 지닌다. 즉, 애자일(Agile) 개발 방식은 폭포수(Waterfall) 개발과 같은 기존 개발방식의 단점을 보완하고자 나온 빠르고 유연한 개발 방식 또는 문화를 의미한다.

폭포수 방식은 ‘요구사항 분석>설계>개발>테스트>배포’의 단계적인 프로젝트 수행 방식이며, 모든 기능이 포함된 프로젝트의 완성을 목표로 한다. 따라서 고객의 피드백은 최종 결과가 나올 때까지 받지 못하고, 디자인 및 코드의 문제점은 출시될 때까지 발견되지 않는다.

애자일 방식도 ‘요구사항 분석>설계>개발>테스트’의 단계적 작업이지만, 프로젝트 완성보다 더 작은 단위의 기능 완성을 목표로 한다. 또한 짧은 주기의 작업을 여러 번 수행하여 고객의 피드백을 지속적으로 받으며 최종 완성까지 달려간다는 차이점이 있다.

### 1.2. 등장 배경

애자일의 등장은 소프트웨어의 개발 프로세스가 다른 공학적 프로세스와는 큰 차이가 있음을 인지하는 것에서 시작된다.

90년대 초반까지 소프트웨어 개발은 장기간에 걸쳐 많은 인력과 충분한 비용을 투자하는 다른 공학적 프로세스와 비슷한 맥락으로 진행되었다. 하지만 급격하게 증가한 컴퓨터 계산 용량과 문제의 복잡성은 기존 개발 방식의 한계를 보여주었다.

소프트웨어는 유동적이며 개방적이다. 요구사항은 개발 중에도 변동 가능성이 높고, 그에 따른 작업량은 예측이 어렵다. 기존의 방식은 초기 설계에 의존적이기 때문에 갑작스러운 변동에 빠르게 대처하기 어렵고, 설계부터 다시 해야하기에 투자한 시간, 비용, 인력을 낭비하게 된다.

이를 극복하고자 다른 개발 방식을 찾게 되었고, 그 중 하나가 애자일 개발 방식이다.

## 1.2.1. 애자일 소프트웨어 개발 선언문(Agile Manifesto, 2001)

우리는 소프트웨어를 개발하고, 또 다른 사람의 개발을 도와주면서 소프트웨어 개발의 더 나은 방법들을 찾아가고 있다. 이 작업을 통해 우리는 다음을 가치 있게 여기게 되었다:

**공정과 도구보다 개인과 상호작용을**  
**포괄적인 문서보다 작동하는 소프트웨어를**  
**계약 협상보다 고객과의 협력을**  
**계획을 따르기보다 변화에 대응하기를**

가치 있게 여긴다. 이 말은, 왼쪽에 있는 것들도 가치가 있지만, 우리는 **오른쪽에 있는 것들에 더 높은 가치를 둔다**는 것이다.

## 1.2.2. 애자일 소프트웨어의 12가지 원칙

- ① 애자일의 최우선 순위는 고품질 소프트웨어를 일찍, 지속적으로 전달해서 고객을 만족시키는 것이다.
- ② 업무의 단순성이 필수적이다.
- ③ 작동하는 소프트웨어가 진척의 주된 척도이다.
- ④ 작동하는 소프트웨어를 자주 전달. 2주~2개월의 간격으로 하되 더 짧은 기간을 선호한다.
- ⑤ 개발의 후반부일지라도 요구사항 변경을 환영한다.
- ⑥ 동기 부여된 개인 위주로 프로젝트를 구성. 그들의 요구를 반영하고, 신뢰하라.
- ⑦ 가장 효율적인 개발 현황 공유 방법은 면대면 대화이다.
- ⑧ 지속 가능한 개발을 장려. 스폰서, 개발자, 사용자는 일정한 속도를 유지한다.
- ⑨ 좋은 기술성과 설계에 대한 지속적인 관심이 기민함을 높인다.
- ⑩ 최고의 아키텍처, 요구사항, 설계는 팀내에서 창발한다. (\*자기조직화)
- ⑪ 운영 담당자와 개발자는 프로젝트 내내 매일 함께 일한다.
- ⑫ 정기적으로 효과적인 방안에 대해 숙고하고, 이에 따라 팀의 행동을 조율한다.

## 1.3. 애자일 프레임워크(Agile Framework)

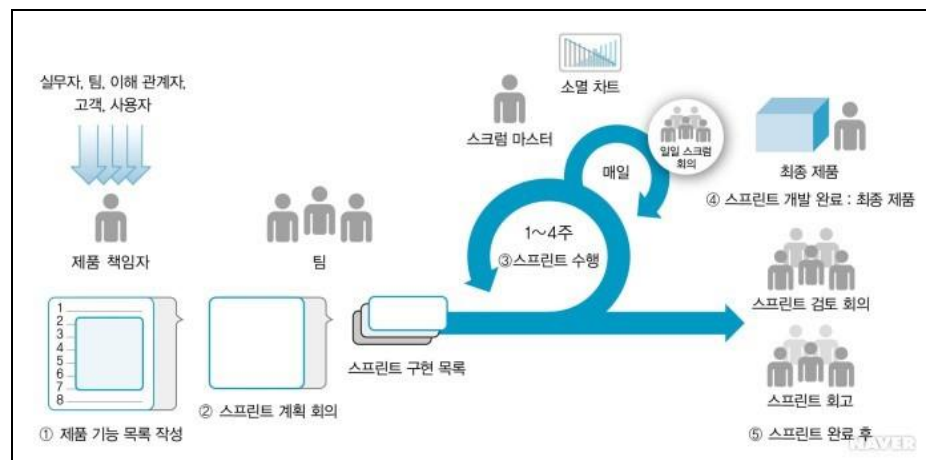
### 1.3.1. 스크럼(Scrum)

#### - 개념

- \*스프린트(Sprint)<sup>ii</sup>를 기반으로 만들어진 주기적 프레임 워크이다.
- 소규모 팀에서 짧은 기간동안 작은 것을 만든다.
- 단, 프로젝트의 전체 모습을 확인할 수 있도록 정기적으로 통합한다.

#### - 스크럼의 특성

- 프로젝트의 기능/개선점에 대한 우선 순위를 부여한다.
- 개발 주기는 한 달로 하며, 주기마다 실제 동작하는 결과를 제공한다.
- 개발 주기마다 적용할 기능 및 개선 목록을 제공한다.
- 항상 팀 단위로 생각한다.
- 날마다 15분의 회의를 가지며, 원활한 의사소통을 유지한다.
- 스크럼의 핵심 요소는 투명성<sup>iii</sup>, 검토<sup>iv</sup>, 적응<sup>v</sup>



<그림 3. 스크럼(Scrum) 진행 플로우>

#### - 스크럼의 진행 과정

- (1) 제품 요구사항을 바탕으로 Task를 최소 단위의 목록(Backlog)으로 만든다.
- (2) 제품 책임자는 Task에 우선순위와 가중치를 부여한다.
- (3) 개발팀을 구성하고, 스크럼 마스터를 지정한다.
- (4) 1번의 스프린트 기간동안 수행할 Task를 지정하여 Sprint Backlog를 만든다.
- (5) 스프린트 동안 일일 스크럼 리뷰를 진행한다.
- (6) 모든 미팅의 시간은 최대한 짧게 반복적으로 나누고, 해당 스프린트 종료시 출시 가능한 코드를 시연 한다.(Code review)
- (7) 해당 스프린트 결과를 보고 및 회고한다.



## - 스크럼의 구성요소

### ① 스크럼 팀(Scrum Team)

- 제품책임자

: 제품 백로그 관리를 담당 / 백로그에 대한 정확한 이해 후 우선순위 지정  
/ 다음 스프린트 작업 제시

→ 최상의 결과물을 만들 수 있는 백로그 구성이 목표

→ 개발팀에 요구사항을 전달하기 위해서는 반드시 제품책임자의 허가를 얻어야 함

- 개발팀

: 매 스프린트에서 완료된 기능을 추가하고, 잠재적으로 출시 가능한 제품 배포가 가능한 전문가들 / 오직 개발만을 담당

→ 제품 기능 구현 방법을 스스로 정하는 자기 조직화 (Self-Organizing) 팀

→ 제품 개발에 필요한 모든 기술이 있는 \*교차기능(Cross-Functional) 팀<sup>vi</sup>

→ 민첩한 대응이 가능한 최적의 개발팀 크기는 4~8명

→ 개발 팀원들은 동등한 직책과 책임을 가짐

- 스크럼 마스터

: 스크럼팀을 도와주는 리더 / 스크럼팀의 스크럼 이론 및 실천내용, 규칙 준수 여부 파악 담당 / 스크럼팀에게 스크럼을 이해시킴

### ② 스크럼 이벤트 모임

- 스프린트

: 최대 한 달 동안 계획한 기능을 완료하여 사용 & 출시가 가능한 제품으로 만들어내는 과정

→ [구성]

: 스프린트 계획 + 일일 스크럼 + 개발 작업 + 스프린트 리뷰 + 스프린트 회고

→ [스프린트 계획시]

1) 이번 스프린트의 작업 결정

2) 이번 스프린트에서 배포 가능한 제품, 배포에 필요한 작업 결정

3) Topic : 개발 가능한 기능 / 기능 구현 방법 / 목표 / 하루 단위 미팅 '일일 스크럼'

→ [스프린트 진행시]

1) 목표에 위협적인 변경을 허용하거나, 제품의 품질 목표를 낮추지 않음

2) 개발 범위가 명확해지면, 개발팀과 제품 책임자는 개발 범위를 설정 및 재협상 가능

→ [스프린트 종료시]

1) 새 스프린트는 직전 스프린트가 끝남과 동시에 시작

→ [스프린트의 취소]

1) 스프린트 기간이 종료되기 전 가능

2) 목표가 불명확해진 경우 가능

3) 제품 책임자만이 스프린트 취소 권한을 가짐

4) 이미 완료된 제품 백로그 검토 후, 해당 작업물의 출시가 가능하면 제품에 포함시킴

5) 미완료 상태의 제품 백로그는 재추정, 필요하다고 판단시 백로그 목록으로 다시 들어감



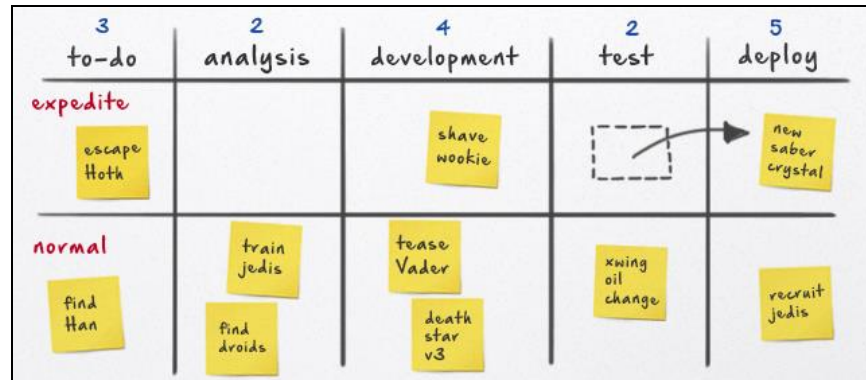
- 일일 스크럼  
: 개발팀만 참여하는 하루 주기의 15분 미팅 / 어제 한 일, 오늘 할 일, 방해요소 논의  
→ 소통 증진, 추가적인 미팅 제거, 개발 방해 요소 제거, 개발팀 건문 증대 효과
- 스프린트 리뷰  
: 스크럼 팀과 제품 이해관계자들이 종료된 스프린트에서 완료된 기능, 변경 사항을 함께 확인하는 시간  
→ 제품을 검토하고, 필요에 따라 백로그를 수정하기 위해 수행  
→ 완료된 작업 시연 후 질문과 답변 시간을 가짐  
→ 변경된 백로그를 고려하여 다음 작업을 의논  
→ 스프린트 동안 일어난 문제와 해결방안을 논의
- 스프린트 회고 미팅  
: 스프린트 리뷰 후, 다음 스프린트 계획 전 스크럼 팀끼리 스스로를 되돌아보는 시간  
→ 스크럼 팀이 스스로를 되돌아보고, 다음 스프린트에서 개선 및 계획 가능한 것 논의  
→ 종료된 스프린트가 사람, 상호관계, 도구 측면에서 어떤 효과를 냈는지 검토  
→ 잘 된 기능과 개선 여지가 있는 항목을 고르고 순위를 매김

### ③ 스크럼 산출물

- 제품 백로그(BackLog)  
: 제품의 요구사항에 우선순위를 매긴 목록  
→ 제품의 다음 릴리즈에 추가될 모든 기능, 요구사항, 개선사항, 수정 버그가 게시됨  
→ 완성된 것이 아닌, 점점 더 나은 기능으로 발전하는 유동적인 요소  
→ 제품 책임자가 제품 백로그의 내용, 가용성, 우선순위에 대한 책임을 가짐
- 스프린트 백로그  
: 스프린트를 위해 선택된 제품 백로그 항목들의 집합  
→ 제품 증분을 배포하고 스프린트 목표를 실현하기 위한 계획서  
→ 다음 제품 증분에 포함될 기능과 기능 구현에 필요한 작업을 예상하여 작성한 목록  
→ 스프린트 백로그 작업들의 진행 현황에 따라 남은 작업량의 추정치를 업데이트  
→ 스프린트 진행 중에는 개발팀만이 변경 가능한 항목  
→ 실시간 진행 상황을 보여줌
- 제품 증분  
: 스프린트에서 완료된 모든 제품 백로그 항목의 집합  
→ 스프린트 종료 시, 생산된 새로운 제품 증분은 반드시 사용 가능한 완료 상태여야 함  
→ 실제 출시 여부에 관계없이 사용 가능해야 함



### 1.3.2. 칸반(Kanban)



<그림 4. 칸반(Kanban) 진행 플로우>

#### - 개념

- 지속적 개선을 추구하는 애자일 프레임 워크이다.
- 각 프로세스마다 이슈를 표시하여 다음 프로세스와의 **연속적인 흐름**을 만들어 **워크플로**를 시각화하고, 전체 프로세스를 유연하게 만든다.
- **To do, Dev, Test, Release, Done** 등의 프로세스로 구분하여 진행한다.

#### - 칸반의 특성

- 현재 상태(기존 프로세스, 역할, 책임, 직함 등) 그대로 시작
- 업무의 진행 상황을 시각적으로 볼 수 있게 만들
- 시각적 신호를 사용하여 진행 중인 업무(WIP, Work in Progress)량을 제한
- WIP를 제한하여 업무의 양을 조절하고, 결과물의 완성도를 높임
- 새로운 업무를 받았을 경우, 기존의 업무를 완료한 후 WIP에 여유가 있을 때 새 업무를 '당겨'오는 '당김 시스템(Pull System)'을 사용
- 칸반 시스템은 스크럼 기법과 함께 사용 가능  
: 스크럼의 단점을 칸반 기법으로 극복하는 것이 효과적
- 칸반의 핵심 요소는 **지속 가능성, 서비스 지향, 생존 가능성**

#### - 칸반의 3가지 규칙

- ① 워크플로우의 시각화 : 일을 작게 분할, 카드에 기록하여 보드에 Flow별로 게시
- ② WiP 제한 : Flow별로 동시에 진행 가능한 일의 개수를 제한
- ③ 리드 타임 측정 : 한 항목을 완료하는데 걸리는 평균시간을 산정.

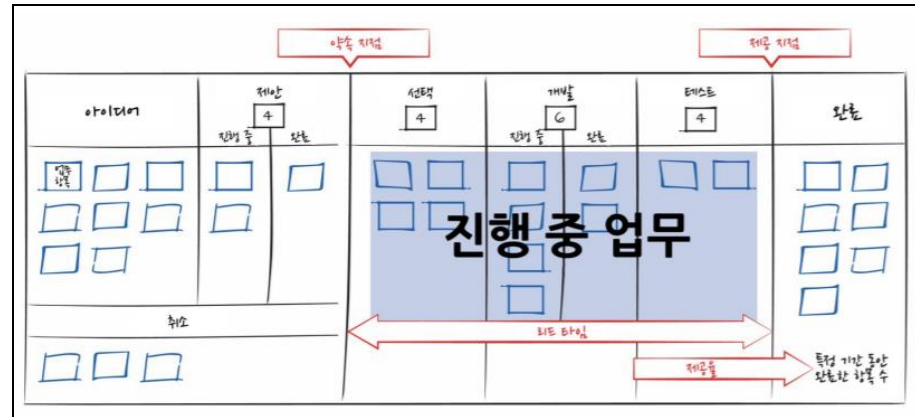
예측 가능하고, 최소화된 리드 타임을 위해 프로세스를 최적화

#### - 칸반의 진행 과정

- (1) 업무를 시각화한다.
- (2) 진행 중인 업무(WIP)의 최대 개수를 제한한다.
- (3) 흐름을 관리한다.
- (4) 정책을 명시화한다.
- (5) 피드백 루프를 실행한다.
- (6) 함께 개선하고, 실험을 통해 발전시킨다.

## - 칸반의 구성요소

### ① 칸반 보드(kanban boards)



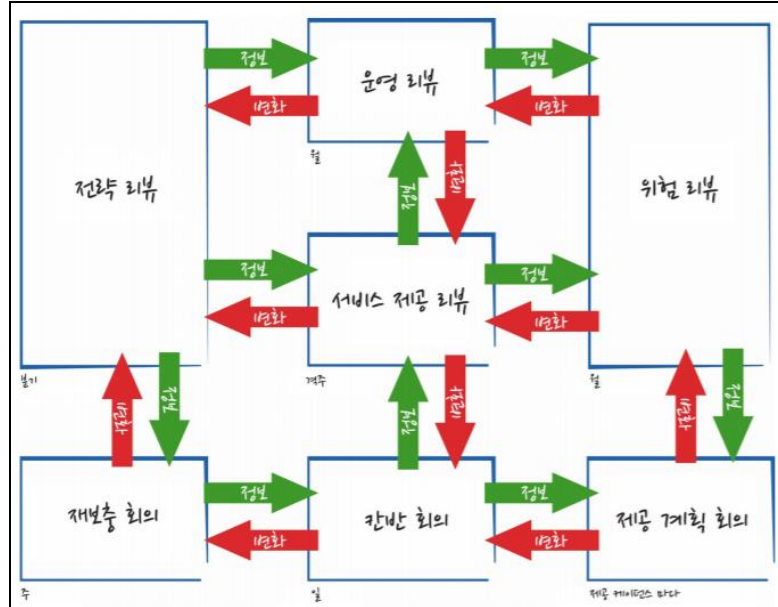
<그림 5. 칸반 보드(kanban board) 예시>

- WiP(Work in Progress / 진행 중 업무)  
: 특정 시점에 진행 중인 작업 항목의 수 또는 집합  
→ 각 단계 별로 WiP의 최대 개수를 제한하여 명시한다.  
→ 너무 많은 WiP는 생산성을 감소시키고, 너무 적은 WiP는 리드 타임을 길어지게 한다.
- 레인(swimlanes)  
: 프로세스의 상태를 나열한 2개 이상의 열
- 약속 지점  
: 업무 항목을 제공할 것이라는 약속이 이루어지는 지점.  
고객에게 제공될 결과물을 위한 업무를 확정하는 작업이 이루어짐.
- 제공 지점  
: 업무 항목을 제공했거나 완료함을 나타내는 지점.
- 리드 타임(Lead Time)  
: 업무 항목이 약속 지점으로부터 제공 지점까지 이동하는 데 걸리는 시간.  
즉, 한 업무 항목이 완료되는 데 소요되는 평균 시간  
→ 리드 타임을 측정하여, 가능한 짧고 예측 가능하게 프로세스를 최적화해야 함
- TiP(Time in Process / 처리 기간)  
: 대상 프로세스에 머물러 있는 시간

### ② 칸반 관련 역할

- 서비스 요청 관리자  
: 고객의 요구를 이해하고, 재보충 회의에서 업무의 선택 및 순위 결정을 촉진하는 역할. 스크럼의 제품 책임자와 비슷한 개념.
- 서비스 제공 관리자  
: 선택한 업무를 고객에게 제공하는 업무 흐름을 책임지고, 칸반 회의 및 제공 계획 수립을 촉진하는 역할. 플로우 마스터라고도 불림.

### ③ 케이던스(Cadences)<sup>vii</sup>



<그림 6. 7개의 케이던스로 구성된 피드백 루프>

- 전략 리뷰(Strategy Review)  
: 제공할 서비스를 정의하고, 서비스와 관련된 외부 환경 변화를 파악
- 운영 리뷰(Operations Review)  
: 서비스 사이의 균형을 이해하고, 고객 만족도를 높일 수 있는 자원을 배치
- 위험 리뷰(Risk Review)  
: 서비스 제공의 위험을 파악하고, 그것에 대응하기 위한 전략을 짚
- 서비스 제공 리뷰(Service Delivery Review)  
: 서비스의 효율성을 측정하고 개선하기 위한 회의
- 재보충 회의(Replenishment Meeting)  
: Task를 약속 지점 이후로 옮긴 후의 준비 상태를 확인하기 위한 회의
- 칸반 회의(The Kanban Meeting)  
: 팀원들이 보통 매일 조정하고 스스로를 조직화하며 리뷰 계획을 수립. 완료된 업무와 이슈 해결에 대해 논의하며, 짧은 회의를 위해 '스탠드업' 형태인 경우가 많음
- 제공 계획 회의(Delivery Planning Meeting)  
: 고객에게 제공하는 산출물을 파악하고, 계획을 수립

### 1.3.3. 스크럼과 칸반의 비교

	스크럼	칸반
주기	정기적 & 고정된 시간의 스프린트(Sprint)	지속적 흐름
릴리즈 방법론	제품 책임자의 승인이 전제된 각 스프린트의 마지막 단계	지속적 배포 또는 팀의 재량
역할	제품 책임자, 스크럼 마스터, 개발 팀	정해진 역할 X, 일부 팀에서 애자일 코치의 도움을 요청
주요 지표	속도	*Lead time
변경 철학	되도록 진행중인 스프린트의 변경은 지양하고, 다음 스프린트에 반영	언제든지 변경 가능
WiP 제한	간접적 (스프린트마다)	직접적 (작업 흐름 단계마다)
보드	스프린트마다 초기화	계속 유지
장점	<ul style="list-style-type: none"> <li>• 일정 주기마다 실행 가능한 제품이 산출되어 고객과의 소통이 활발함</li> <li>• 프로젝트의 시작과 끝이 분명함</li> <li>• 피드백에 유연함</li> </ul>	<ul style="list-style-type: none"> <li>• 프로젝트를 탄력적 &amp; 절차 지향적으로 관리</li> <li>• Dead Line이 없지만 속도에 대한 압박은 존재하여 전체적 생산성 향상</li> <li>• 계획 충족을 위해 개발 범위를 속이는 일이 없음</li> <li>• WiP의 개수를 제한하여 혼란 방지</li> <li>• 다음 진행할 작업의 내용, 병목 현상을 한 눈에 파악 가능하여 프로젝트 관리 오버헤드 감소</li> </ul>
단점	<ul style="list-style-type: none"> <li>• 일일 스크럼, 스프린트 회의에 많은 시간 소요</li> <li>• Dead Line 안에 업무를 완료해야 하는 압박감</li> </ul>	<ul style="list-style-type: none"> <li>• 팀의 특별한 요구에 따라 모델링할 수 있는 칸반 보드를 제공하는 소프트웨어의 부재</li> <li>• 신속함에 대한 조직의 동기부여 상실</li> </ul>

<표 2. 스크럼과 칸반의 차이>

### 1.3.4. 칸반을 통한 스크럼의 보완

보완대상	보완 상세 내용	기대효과
잡은 미팅의 오버헤드	비판적이고 잡은 데일리 스크럼, 회고 미팅으로 불필요한 시간 소요를 최소화함	회의 소요시간 감축 및 긍정적 에너지에 집중
Story Point 추정의 불확실성	예상보다 큰 작업은 스프린트에 오버헤드 초래. 따라서 WiP를 조정하여 불확실성에 대응	프로세스 시각화, 업무예측 용이
스프린트 변경의 경직성	WiP의 조정을 통해 스프린트 주기 내 변경을 유연하게 수용해야 함	업무 프로세스의 유연성 확보

<표 2. 칸반을 통한 스크럼의 보완점>

#### 1.4. 애자일의 장·단점

##### • 장점

- 짧은 주기로 제품을 만들면서 고객의 요구사항을 지속적으로 반영하고, 문제점들을 바로 제거 가능
- 선행 업무가 적다
- 작은 기능을 출시할 때 빠르게 개발 가능
- 점진적인 테스트로 인해 초기에 버그 발견 가능
- 개발 구성원들간의 소통과 협력을 원활하게 함
- 수평적인 의사결정 및 팀원들의 사기를 높이기 위한 방법을 제공

##### • 단점

- 애자일이 지향하는 개발 문화와 자기 조직화된 팀 구성이 어려움
- 사용자가 지속적으로 참여하기 어려운 환경에서는 적절하지 않음
- 체계적인 개발에 필요한 분석/설계/구현/테스트 프로세스 및 설계 기법 등을 제공하지 않음
- 대규모 프로젝트 수행에 필요한 관리 요소 등을 제공하지 않음

#### 1.5. 전통적 및 애자일 개발의 적절한 대상

구분	전통적 개발	애자일 개발
시장 환경	- 시장 상황이 안정적 & 예측 가능	- 고객 선호도 및 솔루션 조건이 자주 변동
고객 참여	- 요구 사항이 분명함 - 고객과 개발팀의 지속적 협력이 어려움	- 고객과 긴밀한 협력 및 신속한 피드백 가능
혁신 유형	- 유사 작업이 선행되었고, 솔루션이 명확함 - 사양 및 작업 계획을 예측 가능 - 사양 준수가 중요	- 문제가 복잡하고 해결 방법을 알 수 없음 - 범위가 불명확 - 제품 사양의 변동 가능 - 기능 혁신과 시장 출시가 중요 - 기능 간 협력이 중요

<표 3. 전통적 및 애자일 개발의 적절한 대상>

## 2. Jira의 Agile 적용 예시

### 2.1. Jira Agile Board

#### Step1. 백로그에 각 기능별로 에픽(Epic)<sup>viii</sup>을 등록

Epic 만들기

이슈 가져오기 필드 구성

프로젝트

스크럼 테스트 (UDEP)

이슈 유형

에픽

호환되지 않는 필드 구성 또는 업무 흐름 연동 때문에 일부 이슈 유형을 사용할 수 없습니다.

Epic Name

1.검색 기능

Provide a short name to identify this epic.

요약

검색하는 기능

다른 항목 생성

만들기

취소

#### Step2. 스프린트 단위로 Story<sup>ix</sup>, 오류 등의 이슈를 생성하여 백로그(Backlog)를 등록

스크럼 테스트

스크럼 테스트 프로젝트

UDEP 보드

보드

백로그

활성 스프린트

보고서

필드

이슈 및 필터

피커

템플릿

할록 추가

프로젝트 설정

UDEP 보드

백로그

Quick filters

Assignee

에픽

에픽 만들기

1.검색 기능

UDEP-19 검색하는 기능

이슈

2

완료

0

예측되지 않음

0

예상

0

에픽에 이슈 생성

2.물작성기능

UDEP-20 물작성하는 기능

이슈

2

완료

0

예측되지 않음

0

예상

0

에픽에 이슈 생성

3.물수정기능

UDEP-21 물수정하는 기능

이슈

2

완료

0

예측되지 않음

0

예상

0

에픽에 이슈 생성

에픽이 없는 이슈

스프린트 2 15 이슈

기능 1 구현 방법 구성

기능 1 구현

기능 1 테스트

기능 1 배포

기능 1 버그

에픽3 이슈

에픽1 이슈

에픽2이슈

기능 1 스토리

검색 할수 구현

검색 정렬 옵션 추가

물 작성 할수 구현

D8 글 타이틀의 등록

물 수정 할수 구현

D8 글 타이틀을 수정

에픽1

에픽2

1.검색 기능

2.물작성기능

3.물수정기능

에픽1

에픽2

에픽3

백로그 5 이슈

기능 2 설계

기능 2 구현 방법 구성

기능 2 구현

기능 2 테스트

기능 2 배포

이슈 생성

### Step3. 이름, 기간, 목표를 설정하여 스프린트 시작

#### 스프린트 시작

이 스프린트에 7개의 이슈가 포함됩니다.

스프린트 이름: \*

기간: \*

시작일: \*

종료일: \*

스프린트 목표:

### Step4. 시작한 스프린트의 스크럼 보드를 관리 (할 일, 진행 중, 완료의 프로세스로 나뉨)

UDEP 보드

#### 스프린트 2

모든 기능 구현

19 days 남은

스프린트 원본

할 일 4

글 작성 함수 구현  
 2.글작성기능  
 UDEP-24

DB 글 테이블에 등록  
 2.글작성기능  
 UDEP-25

글 수정 함수 구현  
 3.글수정기능  
 UDEP-26

DB 글 테이블을 수정  
 3.글수정기능  
 UDEP-27

진행 중 3

기능 1 버그  
 UDEP-11

검색 함수 구현  
 1.검색기능  
 UDEP-22

검색 정렬 옵션 추가  
 1.검색기능  
 UDEP-23

완료 8

기능 1 구현 방법 구상  
 UDEP-2

기능 1 구현  
 UDEP-3

기능 1 테스트  
 UDEP-4

기능 1 배포  
 UDEP-5

에픽3 이슈  
 에픽3  
 UDEP-16

에픽1 이슈  
 에픽1\_기능1  
 UDEP-14

에픽2이슈  
 에픽2  
 UDEP-16

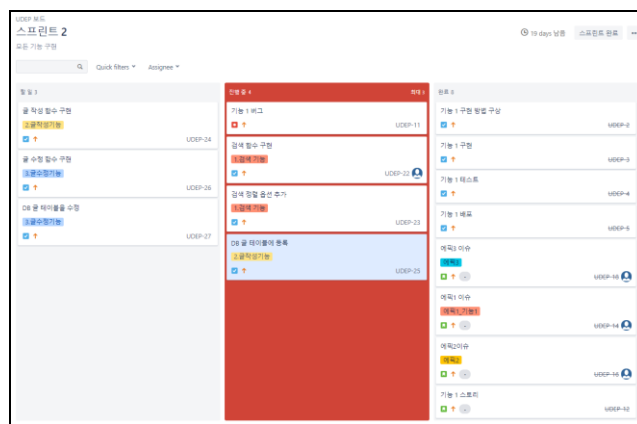
기능 1 스토리  
 UDEP-12

: 스프린트 백로그는 각 에픽별로 레이블이 표시됨

## - 각 작업 흐름당 WiP의 개수 제한이 가능함

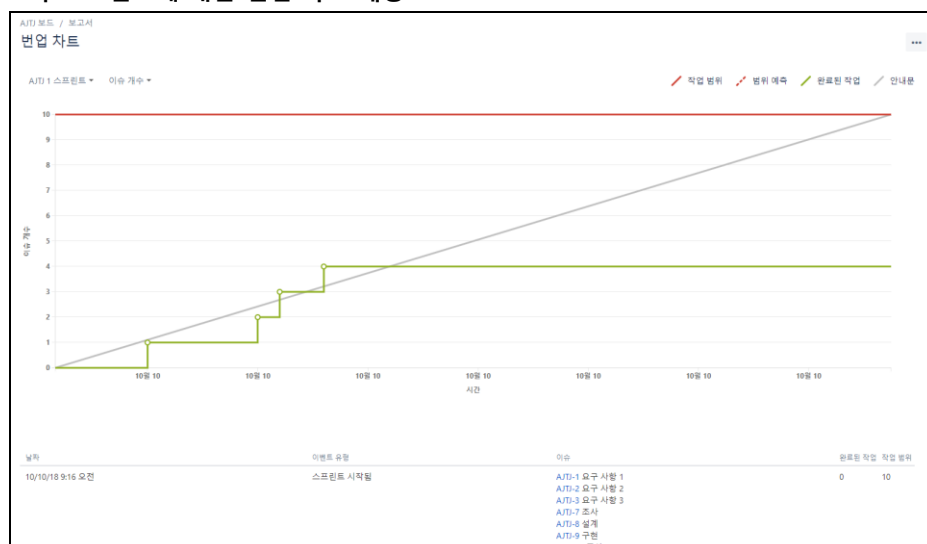


: '진행 중'의 WiP 개수를 3개로 제한



: '진행 중'의 WiP가 제한 개수를 넘으면 하이라이트 표시

## - 각 스프린트에 대한 번업 차트 제공



: 스프린트의 완료된 작업을 전체 범위와 비교하여 보여주는 그래프.

→ 초록색 실선 : 실제 완료한 작업 개수 / 빨간색 실선 : 목표 작업 개수

두 선이 만나면 해당 스프린트의 목표를 충족한 것이며, 간격이 많이 벌어진다면 기간 안에 제품 백로그를 완료하기 어렵다는 것을 의미한다.



### Step5. 스프린트 완료 시, 미완료 이슈를 자동으로 다음 스프린트로 이동

스프린트 완료: 스프린트 2

12 개의 이슈가 완료되었습니다.  
3 개의 미완료 이슈가 해야할 일 목록의 맨 위로 돌아갑니다.

완료되지 않은 모든 이슈를 이동할 장소를 선택하십시오.

다음으로 이동


새 스프린트
 ▼

하위 작업들의 위의 총계에 포함되지 않았습니다. 부모 이슈와 함께 항상 같은 스프린트에 포함됩니다.

완료
취소

### - 스프린트 보고서 제공

UDEP 보드 / 보고서					
스프린트 보고서					
상태 보고서					
완료된 이슈 <span>이슈 탐색기에서 보기</span>					
키	요약	이슈 유형	우선순위	상태	Story Points (-)
UDEP-11	기능 1 버그	버그	↑ Medium	완료	-
UDEP-22	검색 함수 구현	작업	↑ Medium	완료	-
UDEP-23	검색 정렬 옵션 추가	작업	↑ Medium	완료	-
UDEP-25	DB 글 테이블에 등록	작업	↑ Medium	완료	-
미완료 이슈 <span>이슈 탐색기에서 보기</span>					
키	요약	이슈 유형	우선순위	상태	Story Points (-)
UDEP-24	글 작성 함수 구현	작업	↑ Medium	진행 중	-
UDEP-26	글 수정 함수 구현	작업	↑ Medium	할 일	-
UDEP-27	DB 글 테이블을 수정	작업	↑ Medium	할 일	-
이슈가 이 스프린트 외부에서 완료됨 <span>이슈 탐색기에서 보기</span>					
키	요약	이슈 유형	우선순위	상태	Story Points (-)
UDEP-2	기능 1 구현 방법 구상	작업	↑ Medium	완료	-
UDEP-3	기능 1 구현	작업	↑ Medium	완료	-
UDEP-4	기능 1 테스트	작업	↑ Medium	완료	-
UDEP-5	기능 1 배포	작업	↑ Medium	완료	-
UDEP-12	기능 1 스토리	스토리	↑ Medium	완료	-
UDEP-14	예픽1 이슈	스토리	↑ Medium	완료	-
UDEP-16	예픽2 이슈	스토리	↑ Medium	완료	-
UDEP-18	예픽3 이슈	스토리	↑ Medium	완료	-

 <b>KCC정보통신주식회사</b> KCC Information & Communication	Title <b>정보기술연구소 아키텍처 연구</b>	
Document #: 정보기술연구소 Agile 보고서	Version #: 0.1	Issue Date: 2018/10/11

### 3. Review

이번 학기에 졸업작품을 진행할 때, 프로젝트의 초반에 미리 업무를 분배하려고 했지만 정보가 부족한 시점이라 일을 나누기가 어려웠고, 결과적으로 작은 단위의 일을 많은 사람들이 비효율적으로 나누게 되었다.

또한 초기에 분배한 역할은 프로젝트를 진행하면서 그 역할을 더 잘 아는 사람이 맡게 되었고, 한 사람은 역할이 모호해지는 경우가 발생하였다. 이 경험을 바탕으로 일을 잘 나누는 것이 중요한게 아니라, 지속적으로 서로 체크하고 조율하는 방안이 필요하다는 것을 느꼈다.

애자일 개발 방식 중 스크럼의 '데일리 스크럼'과 같이 매일 짧은 회의를 진행하며, 서로의 업무 현황을 파악하면 혼자 결정하기 어려운 문제에 즉각적인 피드백을 받을 수 있어 프로젝트의 방향을 확립하기 수월하다. 또한 자신이 맡은 업무에 대해 매일 얘기해야 하므로 업무에 대한 책임감이 높아진다.


하지만, 스크럼 방식처럼 짧지만 잦은 회의를 해야하는 것은 팀원들에게 Dead line이라는 조급함을 주어 스트레스가 될 수도 있겠다는 생각을 한다. 또한 잦은 회의는 비판적인 피드백을 더 자주 받을 수도 있으므로 팀원의 사기를 저하시킬 수 있고, 불필요한 시간 소요가 될 수 있다.

따라서 스크럼에 칸반 방식을 더해 개발 업무의 Dead line을 만들지 않고, WiP을 스프린트 단위가 아닌 작업 흐름마다 직접적으로 제한시켜 팀원들의 압박감을 줄이는 방안이 필요하다고 생각한다.

애자일 개발 방식은 계획에 의존하지 않고 변화에 대응하는 것을 핵심으로 보지만, 너무 잦은 변화는 프로젝트를 진행하는 팀원들에게 과중 업무를 줄 수 있어 오히려 생산성을 떨어트릴 수도 있다고 생각한다.

애자일의 모든 규칙을 한꺼번에 적용하는 것보다, 현실적으로 가능한 범위 내의 변동을 수용하는 방향으로 조금씩 애자일 방식을 사용한다면 팀원들의 생산성도 높이고, 결과물의 완성도도 높이는 효과를 볼 수 있을 것이라 생각한다.



 <b>KCC정보통신주식회사</b> KCC Information & Communication	Title <b>정보기술연구소 아키텍처 연구</b>	
Document #: 정보기술연구소 Agile 보고서	Version #: 0.1	Issue Date: 2018/10/11

## 5. 용어 정리

### i 자기조직화(*self-organization*)

: 시스템의 구조가 외부로부터의 압력 없이 스스로 혁신적인 방법으로 조직을 꾸려나가는 것

### ii 스프린트(*Sprint*)

: 고정된 시간만큼의 작업을 반복하는 것

### iii 투명성(*Transparency*)

: 프로세스의 중요 부분들을 표준으로 정의함으로써, 결과에 대한 책임이 있는 사람들과 공유하는 것

### iv 검토(*Inspection*)

: 주기적으로 산출물을 확인하여 변경 사항을 찾아내는 것

### v 적응(*Adaptation*)

: 사용되는 프로세스가 허용 기준을 넘은 것을 발견하여, 해당 프로세스에 진행 중인 작업을 조정하는 것

### vi 교차기능팀(*Cross-Functional Team*)

: 하나의 과업을 달성하기 위해 모인, 직위는 같지만 작업 분야는 각기 다른 조직원들로 구성된 팀

### vii 케이던스(*Cadences*)

: 칸반에서의 구체적인 피드백의 기회를 뜻하며, 점진적 변화를 만들고 효과적인 서비스 제공을 위한 주기적 회의 및 리뷰를 말함

### viii 에픽(*Epic*)

: 여러 스프린트에 걸쳐 종료되며, 여러 Story들의 집합. 주로 메인 기능 중심으로 정의함

### ix Story

: 사용자가 원하는 기능을 정의함