1. ver 1
2. Date : 20190714 –

"누가? 왜? 이 행위를 원하는 것이며" - Who & Why

나 자신이 앞으로 개발자로서 살아가기 위해 개발 프로세스를 관리하기 위해 내 자신에게 맞고 필요한 기능들을 추가하기 위함

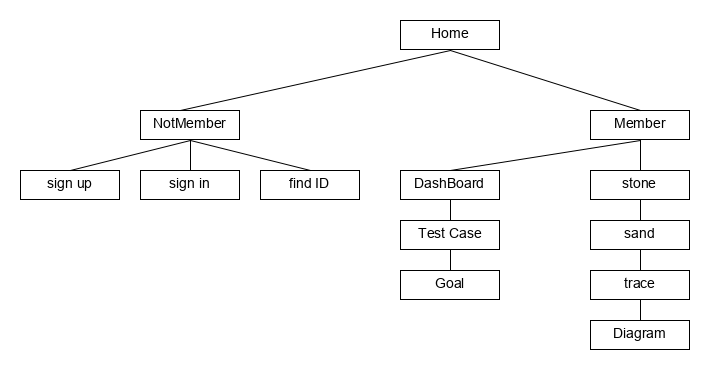
"이 행위의 결과로 얻을 수 있는 비지니스 가치는 무엇인가" – What

개발의 유산@!!

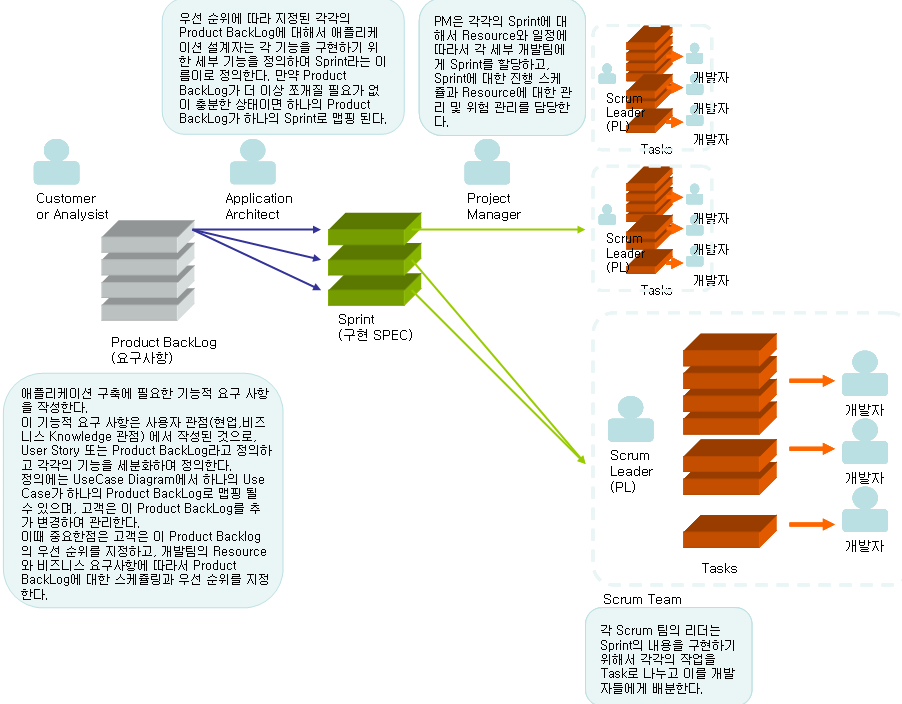
그리고, "이 비지니스 가치를 얻기 위해 하는 행위가 무엇인가가 명확하게 정의되어야 한다" – How

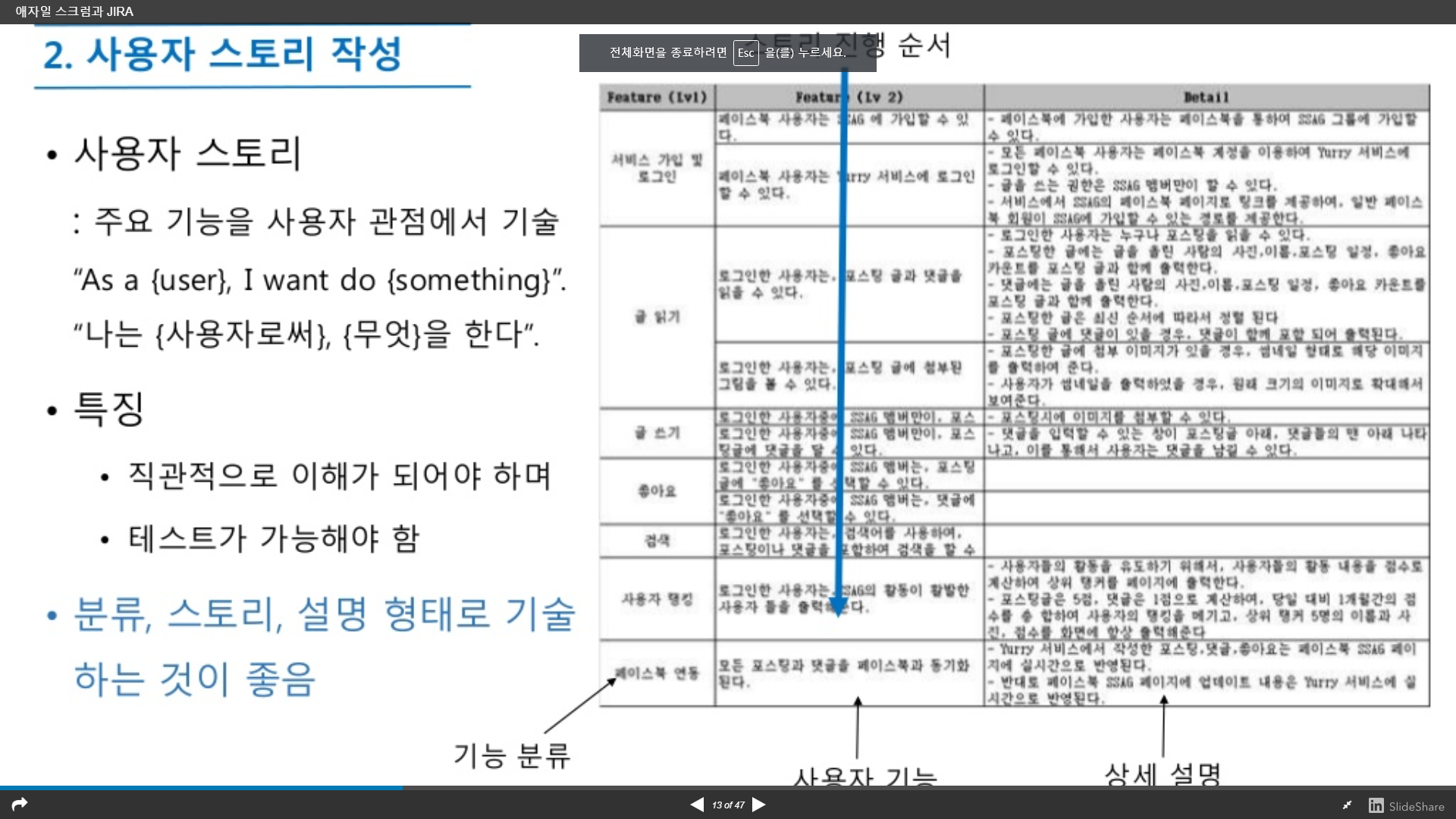
아래와 같이 정의된다.

1. Information architecture



1. Role, 실시간 반영이 중요하므로 socket.io를 이용한다.
   1. 기획+개발+디자인+QA+운영 팀
   2. self organized team 스스로 결정하고 진행



1. Customer
   1. Product Backlog를 입력한다.
      1. Backlog는 추후 설계시, backlog의 입력방법을 다양화할 수 있도록 설계한다.
      2. V1 입력의 요구사항에는 idx, writer, auth, header, content, date, state, 우선순위 가 필요하다.
      3. header를 관리할 수 있다.
      4. 각 backlog는 토의하는 chatting시스템이 존재.
      5. backlog가 반영된(될) 실제 작업 내용을 관계지어 스케줄을 확인할 수 있다.
      6. 
2. application architecture
   1. product Backlog에 대해서 각기능을 구현하기 위한 세부 기능들을 정의하여 Sprint라는 이름으로 정의한다. 만약 Product backlog가 더 이상 쪼개질 필요없이 충분한 상태이면 하나의 ProductBacklog가 하나의 Sprint로 매핑된다.
   2. Sprint는 실제 개발팀에서 분석->설계->구현->테스트->배포까지 수행해야 하는 일련의 프로젝트 단위이다.
3. Project Manage
   1. 각각의 Sprint에 대해서 Resource와 일정에 따라서 각 세부 개발팀에게 Sprint를 할당
   2. Sprint에 대한 진행 스케줄과 Resource에 대한 관리 및 위험 관리를 담당한다.
4. Sprint를 구현
   1. 작업된 내용들의 상세와 결과를 서술한다.
   2. Assignee, Due date
   3. github의 작업 내역과 매핑한다.
   4. 코드 상세 결과를 기술한다.
   5. 예상시간, 시작시간, 걸린시간을 기술
   6. Status는 New/Assigned/In Progress/Postponed/Closed
   7. 단위Task는 실제 작업을 해야하는 Task인데, 보통 1일단위로 나누는 것이 좋고 최대 3일을 넘지 않아야 제대로된 관리가 가능하다.
   8. task와 subtask 관계를 표시한다. owner, status, priority, dependency idx, date
5. 우선순위
   1. Priority (우선순위)

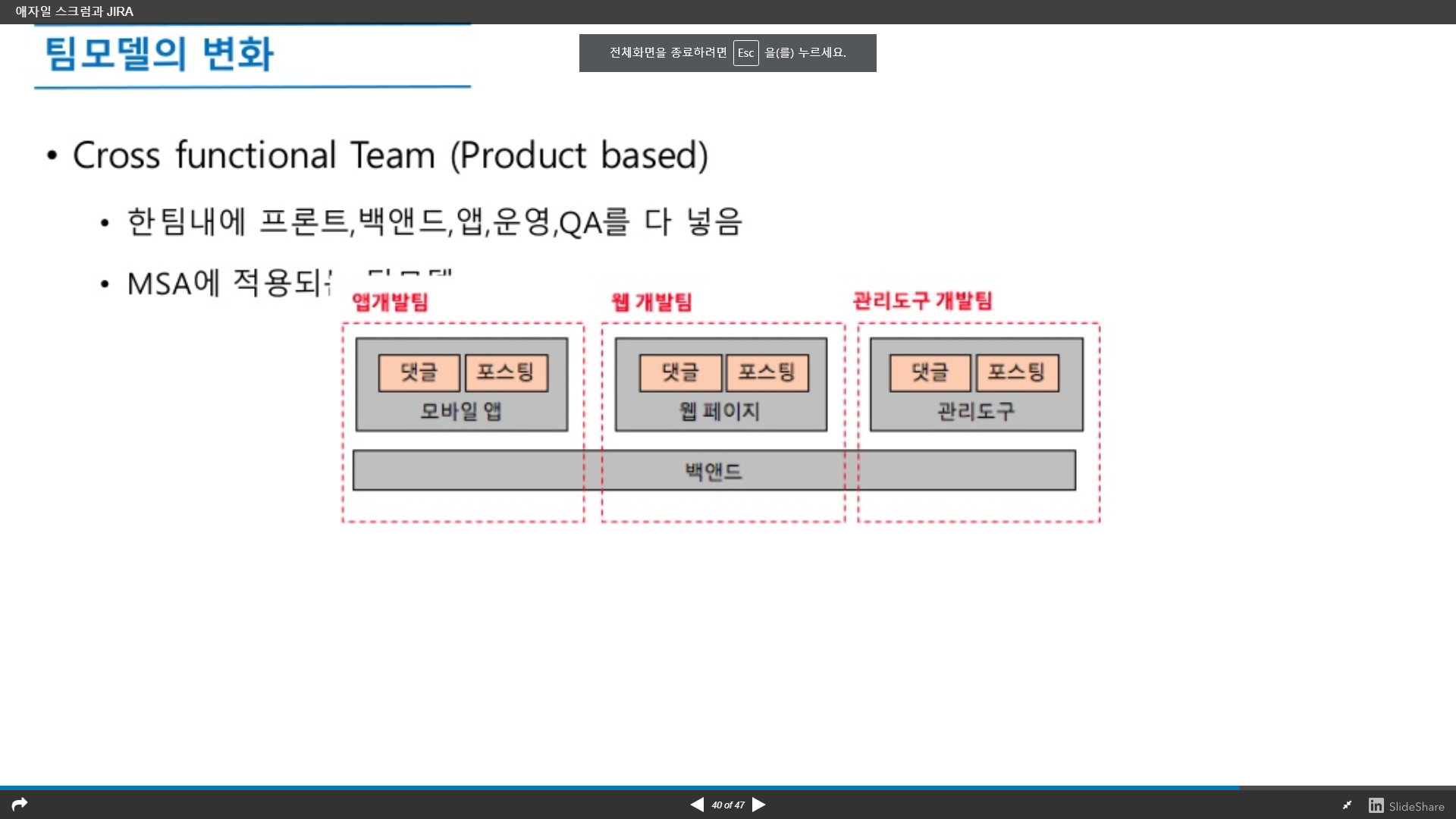
• Blocker의 경우 해결되지 않으면 프로젝트가 진행될 수 없는 경우

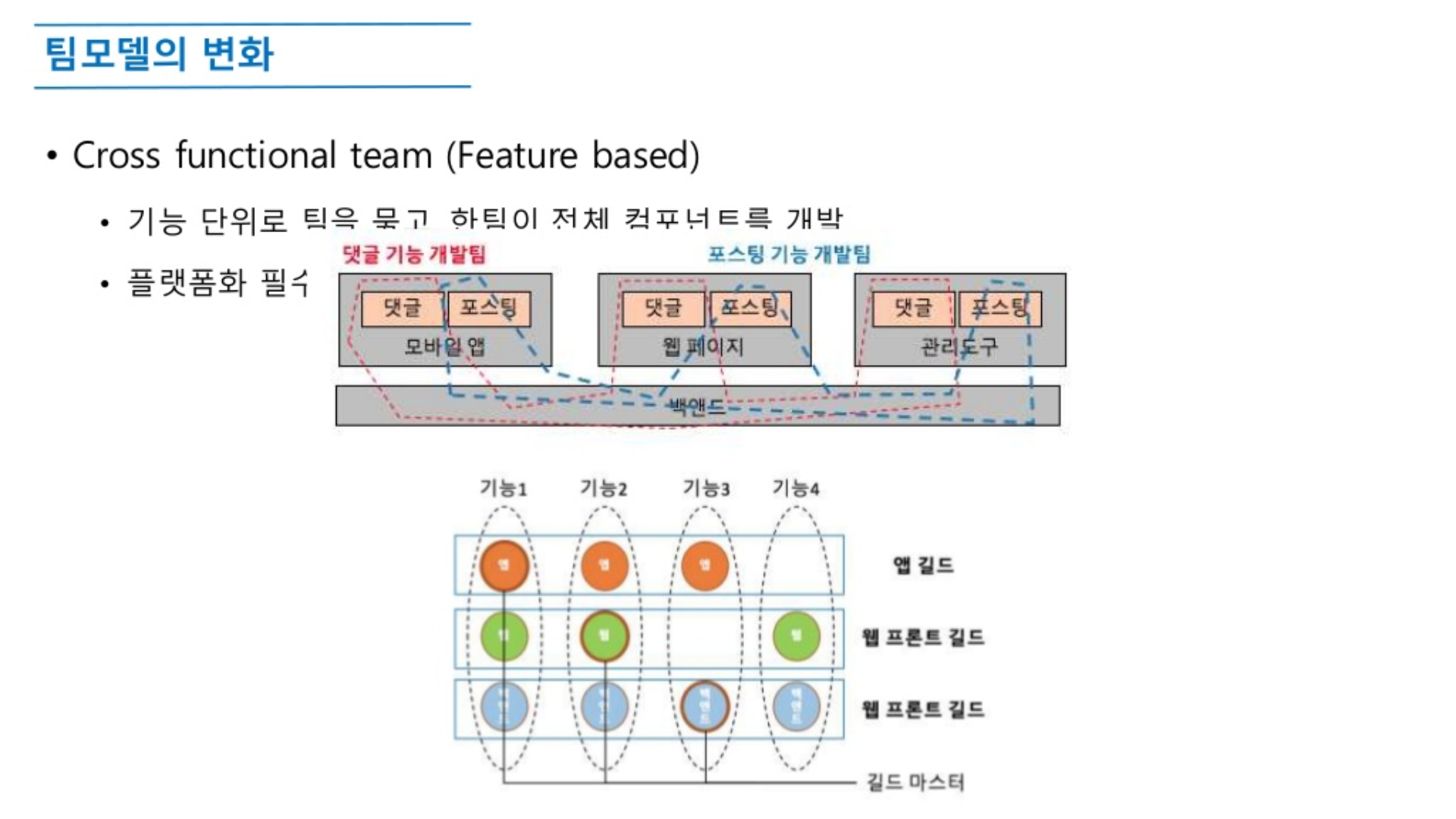
• Critical은 프로젝트 진행은 가능하나 해결하지 않으면 정상적인 서비스 개발이 어려운 경우

• Major 꼭 개발해야 하는 경우

• Minor 개발은 해야 하나 없어도 상관 없는 경우

• Trivial 있으나 없으나 크게 상관 없는 경우

* 



1. content
   1. sign up
      1. id
         * redundant check
      2. pw
         * redundant check
      3. email
         * redundant check
   2. sign in
      1. id, pw
         * right check
   3. find ID
      1. require email
         * check
         * send pw & id to email
   4. DashBoard
      1. test case list
         * upto 10
         * requirement content
         * classification name
      2. goal
   5. stone
      1. Information architecture
      2. milestone
      3. restful api
      4. required state
         * put?
         * list?
      5. sand & Trace
         * break down and analysis required state
         * standby list
           1. order (priority)
         * sand information
           1. ID
           2. name
           3. Type
           4. modifier
           5. modified date
           6. created date
           7. creator
           8. status
           9. resolution
           10. detail
           11. owner
           12. due date
           13. priority
           14. follower
           15. parent
           16. dependent
           17. mentions
   6. 기능적 요구사항

사용자 관점(현헙, 비즈니스 지식 관점)에서 작성된 것. = UserStory, Product Backlog

각각의 기능을 세분화하여 정의

* 1. Diagram
     1. link