The Mythical Man-Month

Chapter 1 타르 구덩이

- 프로그래밍 시스템 제품
  - 프로그램은 소규모로도 만들 수 있는 것이다.
  - 프로그램에 시간과 비용을 들여야 프로그램 제품, 프로그램 시스템이 되는 것이다.
  - 여기서 시간과 비용을 더 들여야 프로그램 시스템 제품이 되어 진짜 쓸모 있는 것이 된다.
- 프로그래밍의 즐거움
  1. 무언가를 만드는 데서 오는 순전한 기쁨 : ex) 레고
  2. 다른 사람에게 쓸모 있는 것을 만드는 기쁨
  3. 복잡하게 움직이는 것을 만들고 작동하는 것을 보는 기쁨
  4. 지속적인 배움에서 오는 기쁨
  5. 유연하고 다루기 쉬운 표현 수단으로 작업하는 데서 오는 기쁨 : 상상력
     a. 그러나 상상력을 현실로 만들 수 있음
- 프로그래밍의 고달픔
  - 완벽함을 요구한다.
  - 내가 아닌 다른 이들이 내 목표를 설정하고 자원과 정보를 제공한다.
    - 주어진 책임에 비해 권한이 부족하다.
  - 다른 이에게 의존해야 한다.
    - 다른사람이 만든 코드를 고쳐야하는 경우
  - 큰 개념 설계는 재미있지만 자잘한 것은 재미 없는 일이다.
  - 버그와 싸우는 일
  - 오래 개발했지만 시기를 놓친 제품이 되어버리는 것

따라서 프로그래밍은 재미있는 타르구덩이이다.

Chapter 2 맨먼스 미신

- 낙관주의

    - 일정대로 잘될 것이라고 낙관적으로 전망하지 마라
        - 버그는 필연적이고 일정은 다른 일정에 꼬리에 꼬리를 물고 있다.
        - 따라서 낙관적인 계획 수립은 실패의 원인이다.
- 맨먼스
    - 소프트웨어 개발에서 인력과 기간은 상호 동등하지 않다.
        - 소프트웨어 개발에서 분업을 하면 커뮤니케이션에 따른 노력이 증가한다.
        - 분업을 할수록 노동자는 늘어나고 커뮤니케이션에 필요한 비용, 노력이 증가하게 되는 것이다.
            - n ⇒ 2 * (0.5n + a)
            - n(n-1)/2 : 상호간 커뮤니케이션일 경우
        - 인력이 추가된다고 일정이 단축되기보단 오히려 연장될 수 있는 것이다.
- 시스템 테스트
    - 디버깅에 절반을 할애하고 계획 수립, 코딩 순으로 시간을 할애하라
        - 테스트에 충분한 시간을 쏟지 않으면 결과가 좋지 않다.
            - 심리적으로도 좋지 않으며 재정적으로도 악영향을 끼친다.
- 비겁한 견적
    - 기간 내에 할 수 있는 견적을 짜라
- 되풀이되는 일정 참사
    - 지연된 프로젝트가 발생 시
        - 낙관적으로 생각하지 말라
        - 맨먼스로 해결할 수 없다. : 새로 투입한 인원 훈련 기간에 시간이 걸린다.
            - 차라리 원래 인원으로 작업 진행하는 것이 낫다.
        - 일정 자체를 가능하도록 수정해야한다.
        - 불필요한 임무를 제거해야한다.
    - 인력은 최소로 일정은 최대로라는 원칙으로 계획을 잡아야 한다.

Chapter 3 외과 수술 팀

- 문제

    - 뛰어난 프로그래머와 평범한 프로그래머의 능력 차이는 10배 정도이다.
    - 그러나 아무리 소수 정예라 할지라도 대규모 프로젝트에서 많은 인원의 속도를 따라잡을 수 없다면 그들이 만든 프로젝트는 구닥다리가 되어버린다.
- 밀스의 제안, 운영방식
    - 대형 프로젝트에서 외과 수술 팀같은 커뮤니케이션 방식을 제안했다.
    - 프로그래밍과 행정, 기록 등의 업무를 나누어 업무의 전문성을 높이고 커뮤니케이션을 단순화 할 수 있다.
    - 프로그래밍을 개인적인 업무가 아닌 팀 단위, 즉 사회적 행위로 변화시켰다.

Chapter 4 귀족 정치, 민주주의, 시스템 설계

- 개념적 일관성
    - 개념적 일관성은 시스템 설계에서 가장 중요하다.
    - 여러 명이 프로그래밍 작업을 나눈 경우 개념 불일치가 발견된다.
- 개념적 일관성의 획득
    - 시스템은 단순하면서도 명확하게 설계해야한다.
    - 단순성과 명확성은 개념적 일관성에서 비롯되는 것이다.
    - 시스템의 모든 부분에서 동일한 철학이 반영 되어야한다.
    - 시스템 사용의 용이성을 위해 설계의 통일성과 개념적 일관성이 필요한 것이다.
- 귀족 정치와 민주주의
    - 훌륭한 기능이나 아이디어보다 개념적 일관성이 더 중요하다.
    - 아키텍트는 소수여야 하고 구현자는 많아도 된다.
    - 모두 창의적이지만, 구현 방안 설계가 아키텍처 설계보다 창의적일 수 없다.
    - 아키텍처 끝나기 전에 구현자에게 본격적인 일거리를 주지 말고
        - 제약사항, 데이터 흐름, 아키텍트팀과의 커뮤니케이션 등을 준비하게 해야한다.