The Tar Pit, The Mythical Man-Month

Tar Pit

수많은 짐승과 공룡이 빠져 죽은 타르 구덩이를 뜻한다.

대형 시스템 프로그래밍이 바로 이것과 같아, 수많은 개발자가 고통받고 있다.

The Programming Systems Product

> Why then have not all industrial programming teams been replaced by dedicated garage duos?

- 프로그램: 완성, 실행 가능
- programming product:
  run/test/repair/extendable
  - tests shoud be
    prepared, run, recorded.
  - document should exist.
- programming system:
  collection of programs
  - precisely defined interfaces
  - prescribed resource budget
- **Programming System Product:**
  costful, but much **useful** object

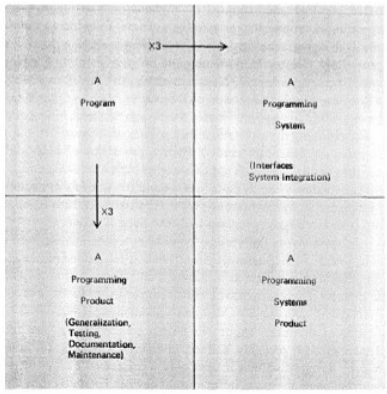


Fig. 1.1 Evolution of the programming systems product

Joys of the Craft

1. **Sheer Joy of making** things
   만듬이라는 행위의 순수 재미
2. Pleasure of making things that are **useful** to other People
3. fascination with crafting complex, puzzle-like objects
4. joy of always Learning
5. delight of working in such a tractable medium.
   "Programmer builds his castles in the air, from air, creating by exertion of the imagination"

Woes of the Craft

1. It must perform PERFECTLY
2. dependencies upon other programmers.
   ex. maldesigned, poorly implemented, no test cases, no doc, ...
3. finding little bug is just Work. no exception.
4. debugging gets longer and longer...
5. product which one has labored so long appears to be Obsolete...

Challange is ...

finding **Real solutions to Real problems on Actual resources**,
not before completion.

The Mythical Man-Month

왜 소프트웨어 개발에는 시간이 부족할까?

1. Estimating technique is poor
   ex. "that all will go well"
2. confuse effort and progress
   : men and month are NOT interchangable!!
3. lack of courteous stubbornness
   : 고집이 부족하다
4. progress is poorly monitored.
5. adding **manpower** makes matters **worse**.
   : 사람을 추가한다고 빨라지는게 아님!!

Optimism

programmers are optimists.
ex. "this time it will surely rum"

Tractable medium

프로그래밍은 아주 다루기 쉽다.

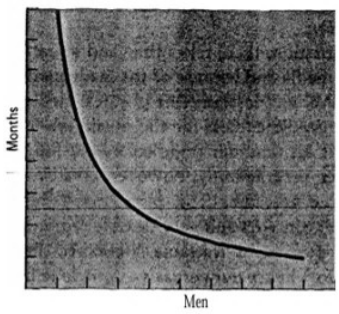
상상하고, 자유롭게 표현하면 된다.
즉 종이/페인트 등으로 제한되는 다른 창작 활동과 아주 큰 차이가 있다.

- 따라서 만들기 쉬울 것 같다는 착각을 불러일으킨다.

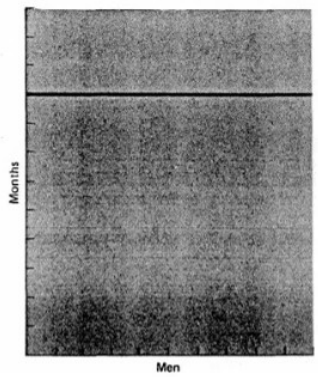
Man-Month

- 적은 사람이 오래 걸리는 것을
  많은 사람이 금방 할 수 있을까?

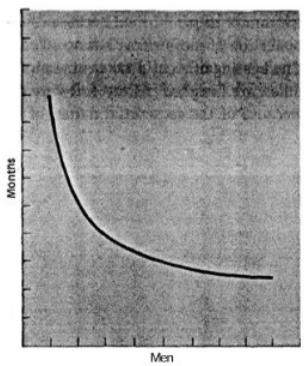
1. PERFECTLY partitionable work
   교환 가능



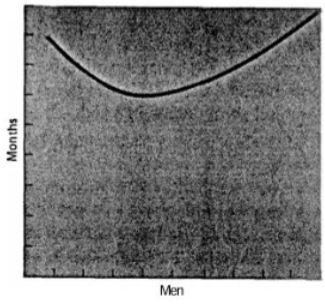
2. UNpartitionable work
   몇명이 달려들어도 소용이 없다



3. partionable, communication needed
   소통하는 overhead가 발생



4. Very Complex
   - 사람 수에 따라 소통 오버헤드가 ${}_nC_2$ 로 증가



Testing

- 1/3 Planning → Larger!
- 1/6 Coding → Small!
- 1/4 Component test, early system test
- 1/4 System test

> 테스트에 많은 시간을 할애하는 이유?

막판에 테스트를 하는데, 기한이 다가와도 아무도 문제를 모르는게 문제다.

또한 금전적/심리적 압박도 한몫한다. 테스트가 실패할때마다 2차, 3차적인 손실이 발생하기 때문이다.

Regenerative Schedule Disaster

ex. 12 man-months task, 3man * 4 months, one milestone per month. (ABCD)

suppose first milestone (A) is reached at two months. two months left.

1. Assume only first part was misestimated.
   → B,C,D should be done in 9 man-months
   → 2 months * 5 men; add 2 men
   - 그러나 단순히 추가한다고 되는게 아니다.
     두명은 한달동안 적응기간이 필요하고, 가르치는데 한명이 필요하다.
     → 한달동안 원래 남은 2명이서만 일하므로 2mm밖에 일을 못함
     → 나머지 한달동안 7mm짜리를 5명이 해야 함 → 기한 못맞춤
   - 두명이 아닌 4명을 추가한다면?
     → 나머지 한달동안 7mm짜리를 7명이 해서 "이론상" 딱 맞춤
     → 그러나 사람이 많아진다고 적게 걸리는게 아니다...

2. Assume whole task was misestimated
   → Since A took 6 man-months, B,C,D need 18 man-months.
   → 2 months * 9 men; add 6 men
3. Do not add people; Reschedule with enough time
4. Do not add people; Trim the task.