뛰어난 프로그래밍 능력이란?

1월 27일 패널

정영범

- 학교 / 산업체 개발 차이점
- 2004년 분석기 개발 시작
- 너무 느렸다
- 워크리스트 알고리즘 고쳤더니 10분만에 끝났다
- 알고보니 할일도 안하는 알고리즘이었다
- 혼났다 이후 최적화후 빠르게 했다
- 산업체에서 이런 일이 있었으면 난리가 났다
- 산업체에서는 프로세스의 도움을 많이 받는다
- 이런 도구를 잘 쓰는 능력이 필요하다
  - 이슈관리자, SVN 사용
  - Writing skill

도경구

- Pascal 분석기 구현
- 박사과정 SML 이용
- 오랫동안 학생 강의한 경험에서 보면
- 프로그래밍 기술은 타고나는 것 같다
- 다른 사람 코드를 보고 배우기도 하는데 그런 패턴을 잘 적용시키는 사람, 그러지 못하는 사람
- 프로그래밍 스타일 유도
  - 예: 빈칸 채우기, 빈칸을 늘여간다
  - 이러니 대부분의 학생이 의도한 스타일로 가긴 한다
- 1. 프로그래밍 기술은 훈련을 통해서 가능하다
- 2. 다른 사람의 코드를 보면서 기술을 빨리 배울 수 있다

김현하

- 1단계. 무슨 프로그래밍을 해야하는지 요구조건을 이해하는게 이해
- 2단계. 프로그래밍을 잘한다
- Tech. writing과 비슷하다
- 멋진 코드를 쓰는 능력은 다른 것이다
- 규칙을 정하고 훈련을 함으로써 가능하다
- 요구조건을 잘 찾아가는 훈련이 중요하다

류석영

- 운전의 롤모델은 택시 기사다
- 좋은 선생님을 모셔야 한다
- SML/NJ Appel 코드를 읽고 이해했어?
- 다른 사람 코딩(Guy, Yarn?)을 보면 tech. writing 하는 것처럼 자기 코드를 계속 물어보고 바꿔간다 (방망이 깍는 노인)
- 4명이 마주보고 앉아 보는데 최상의 코더와 함께 일하는 게 행복하다
- 타고나는 것도 훈련하는 것도 있고...
- SUN에서 구른 경험이 있어서 JavaScript 소프트웨어 개발할 수 있었다

오학주

- 툴, 라이브러리, 언어를 빨리 익히고 자유자재로 쓰는 능력도 중요한데
- 내공이 뭔지 모르겠다
- OCAML을 잘하면 좋겠다
- 회의가 드는데 OCAML을 가르쳤는데 잘 모르더라
- 허충길: OCAML로 잘 짠다는 것은?
- 답변: 아름다운 코드를 잘 짠다
- 류석영: 짧고 간결하게 짜는게 좋은 것은 아닌 것 같다

이욱세

- 프로그래밍은 기획에서 마이크로레벨까지 있음
- 주제를 좁혀서 코딩에 한정하면
- 빠른 시간에 짜는 것 중요
- 1. 로직을 기술
  - OCAML은 로직을 짧게 기술할 수 있다
- 2. 추상화 능력
  - 재사용가능한 코드를 잘 짠다
- 3. 디버깅 능력
  - 원인을 찾아야 한다

배성경

- 정보올림피아드 코딩 시작
- 대학오기전 대학온 후 프로그래밍을 잘한다는게 뭔지에 대해서 생각이 달라짐
- 전: 빠른게 돌아가는게 중요, 맞는 것은 당연
- 후: 무엇을 하는게 정하는지 중요
- 0. 맞게 짜는 능력
- 1. 테스트를 잘 짜는 능력이 중요
- 2. Readability 다른 사람 읽기 좋게 짜는 능력
- 3. 방어적 프로그래밍 능력이 중요
- + 디버깅 중요

강지훈

- 옷을 잘 입는 방법: 잘 생긴다. 옷을 입는다.
- 프로그래밍: 똑똑하다. 프로그래밍 한다.
- 똑똑하다: 체계적인 반성하는 능력

- 이광근
  - 처음에 C, 그 다음에 고급 탕수육
- 윤용호
  - 전자과 컴파일러 C로 구현
  - 그 다음에 Ocaml로 하니 감동
- 홍신
  - SE 전공자, 관심을 가져달라
  - 빠진 topic: 중요한 능력, 보통 코더와 팀으로 일을 같이 할 것이기 때문에 프로그래밍의 리더쉽이 있어야 한다, 코드를 보여줘서 따라올 수 있게 하는 능력

- 김한준
  - 프로그래밍 = writing 관점에서 보면
  - 운영체제 과목 보면 실력이 는다, Pintos 프로그램을 보면서 실력이 많이 는다
  - 코드를 읽어보고 고쳐봐야 한다
- 배성경
  - 같은 생각
  - 다른 사람 코드를 읽을 수 있는 Python 배우는 프로그램
- 류기열
  - PL 과목이 아니라 다른 모든 전산 과목에서 프로그래밍을 제대로 가르쳐야

- 허기홍
  - 프로그래밍 = Technical writing
- 허충길
  - 빠르면서 정확한 것을 찾아야 한다
  - 모순이다
  - 계속 고민하면서
  - dependency 줄이면서 abstract 하게 짜야한다
  - Performance 나오게 짜야한다

뛰어난 프로그래밍 능력이란?

1월 27일 패널 요약

- 산업체 개발 프로세스에서 쓰는 이슈관리자, 버전관리자와 같은 도구 사용 능력
  - Jira, SVN, Git, ...
  - 정영범
- 프로그래밍 실력은 타고 난다?
  다른 사람의 코드를 보면서 프로그래밍 기술을 빨리 배울 수 있다
  - 도경구
- 무슨 프로그래밍을 해야하는지 요구조건을 이해하는 능력
  - 김현하

- 좋은 선생님이 필요하다
  코드를 계속 물어보고 바꿔가야 한다
  – 류석영
- OCAML로 아름다운 코드를 잘 짠다
  – 오학주

⇒ 짧고 간결하게 코드 짜는게 항상 좋은 것은 아닌 것 같다
– 류석영

- 1. 로직을 기술하는 능력
  2. 추상화 능력
  - 재사용가능한 코드를 잘 짠다
  3. 디버깅 능력
  - 이욱세
- 0. 맞게 짜는 능력
  1. 테스트를 잘 짜는 능력
  2. 다른 사람 읽기 좋게 짜는 능력
  3. 방어적 프로그래밍 능력
  - 배성경

- 똑똑함: 체계적으로 반성하는 능력
  - 강지훈
- 프로그래밍의 리더쉽: 코드를 보여줘서 따라올 수 있게 하는 능력
  - 홍신
- 코드를 읽어보고 고쳐봐야 한다
  - 김한준
- dependency 줄이면서 abstract 하게 작성 Performance 나오게 짜야한다
  - 허충길

뛰어난 프로그래밍 능력이란?

1월 28일 패널

김철기

- Top-down: (자기 머리속에 있는 내용을) 요구조건을 명세하는 능력
  - 명세를 잘하면 자기가 뭘해야 할지를 안다
  - 명세를 잘 못하면 cowork을 할 때 문제가 발생한다
- Bottom-up: 컴퓨터 입장에서 생각해 볼 수 있어야 한다
  - “너가 컴퓨터가 되어 봐”
  - 인간이 가진 가정때문에 코너케이스가 생김
  - 디펜스브 프로그래밍
- 어떻게 키울것이냐? Peer review

전병곤

- 경험: 프로그래밍 많이 해봐야 한다
  - 회사에서 스스로 실력이 늘어가는 것을 봤다
- 서울대 과목
  - tool 사용
  - requirement specification
  - agile process통해서 milestone 점검
- 이후 꾸준히 경험을 쌓는게 중요
- 연구실 예:
  - code review를 통해서 많이 배움
- 노력을 할 때 중요한 recipe:
  - 잘 정의된 programming principle 습득해서 적용
  - 여러 principle을
- Open source 참석해서 잘하는 사람들과 interaction하는 게 정리

- 이광근
  - crowd reviewing
  - 학생들이 싫어 했음
  - 그러나 몇 학생들이 많이 배웠음
- 류석영
  - 회사에서 code review 강제
- 전병곤
  - open source 프로젝트에서 code review함

박지혁

- 반성하는 태도, '왜'라는 생각을 가지고 반성
  - 예: recursive call을 while로 바꾸어라는 의견만 받으면 실력이 늘지 않음
- 초급이라면 라이브러리를 잘 쓸 줄 알아야 중급으로 넘어갈 수 있다
- Pair programming이 중요하다
  - 동아리에서 pair programming을 하면서 선배로부터 세세한 것까지 들으면서 배움

- 류기열
  - 어떤 부분이 프로그래밍에서 가장 어려운가?
- 박지혁
  - 같은 부분에서 항상 어려움이 발생하는 것은 아니다. 프로젝트마다 다름.
- 전병곤
  - Pair programming: 잘 하는 사람과 pair 되어야 한다
  - 학생들 코드:
    - exception handling 잘 하지 않는다
    - concurrency handling 잘 하지 않는다

- 최광훈
  - “Haskell? 라이브러리에 다 있어”
  - 좋은 API에 대해서 정리를 잘 정리
  - API 뒤에도 프로그래밍 모델이 있다
  - 따라서 이런 능력을 키워야 한다
- 김철기
  - 인터넷에서 잘 짠 코드를 많이 가져온다

백진언

- 프로그래밍 경험이 별로 없음
- ACM 대회 코드 작성시 느끼는 것
  - 많이 틀린다
  - 유도리
  - 수학적 마인드가 있어야 할 것 같다
    - 수학 증명을 axiom에서 시작하듯이
- 전체 component를 보는 능력이 필요하다

- 허충길
  - 짤 때 꼼꼼하게 짜는 것, 이게 힘들었음
  - 그래서 툴의 도움이 있었으면 좋겠다
- 전병곤
  - 꼼꼼하게 짜는 것은 좋다
  - 수학자 관점에 대해서 생각: 현실적으로 production system 개발할 때 spec 쓰는 것은 쉽지 않다, SW spec은 계속 evolve된다

조성근

- 중요한 것: 변수와 함수 이름 짓는 것
  - Coq programming에서 dejavu를 많이 봄
  - 이름을 잘못 지어서 그런 것
- 함수가 하는 일을 가장 간결하게 표현하는 이름을 찾아야 한다
- 이름을 잘 못찾으면 함수 할 일을 잘 정의한게 아닐 수 있다

- 김한준
  - 주석을 통해서 함수하는 일을 익히는 경우가 많이 때문에 필요하다
- 전병곤
  - 코드 보고 이해안되면 참조할 수 있도록 주석을 쓴다
- 윤용호
  - OCAML programming에서 함수 이름짓기 어려운 적이 많았는데?
- 조성근
  - 여전히 잘 지어야 한다

강동옥

- 프로그래밍은 searching이 잘 되어야 한다
- 코드 읽기는 쉽지 않았다
- 잘 모르는 사람이 바로 잘 따라 갈수 있도록 짜야한다

- 강지훈
  - tool을 잘 쓰면 된다
- 최광훈
  - Flow를 따라가기 어려운 것을 말한게 아닐까?

- 류기열
  - 학생들이 제일 어려워하는게 모델링
  - 문제를 보면 어려워하는게 접근
  - Programming paradigm에 맞는 모델링을 하는게 중요