Apprenticeship Patterns
견습 과정 패턴

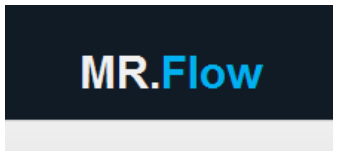
SPARCS 세미나, Feb, 24th 2011

김민우
Julingks_at_sparcs.kaist.ac.kr

NEXR
TOWARD OPEN PLATFORM

앞에 있는 사람은 누구인가?

- 카이스트 05학번
- 07년 SPARCS 회장
- OTL 개발자 (otl.kaist.ac.kr)
- 넥스알 (NexR)
  - iLook, Hadoop Source, MR.Flow, HadoopAppliance, iCube Cloud, EMR



이 세미나의 목표는 무엇인가?

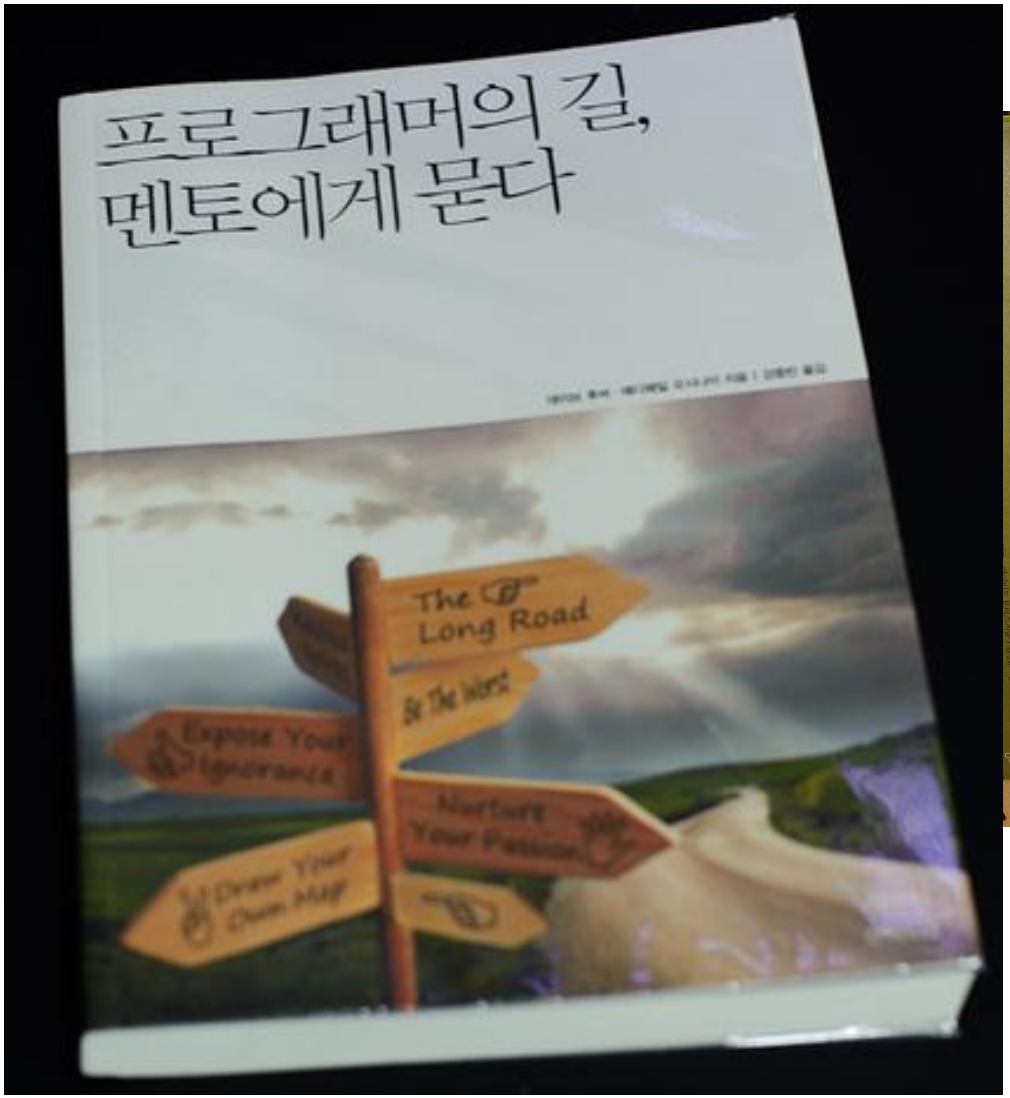
- 기술을 습득하고 수련하는 것은 피할 수 없는 운명
- 그것 자체가 수련해야 할 또 하나의 기술
- 스스로를 키우는 법을 배워야 한다.

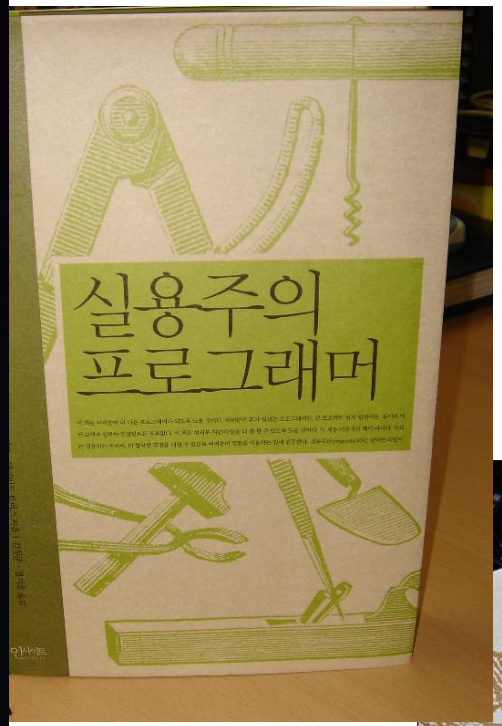
- 경험이 적은 소프트웨어 개발자들이 흔히 마주치는 **딜레마**에 대한 **해결책** 공유

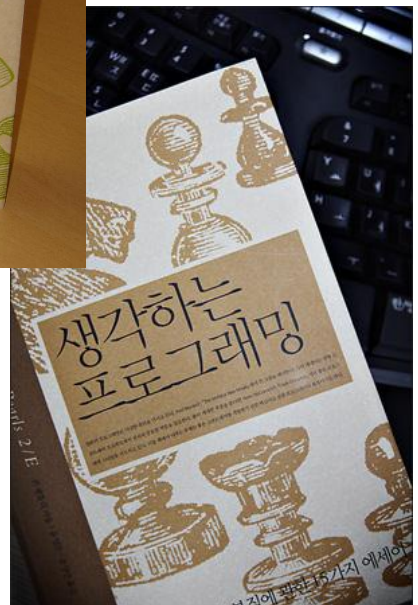
어떤 사람이 들어야 하는가?

- 훌륭한 소프트웨어 개발자로 성장하기를 열망하는 사람
- 대학생 혹은 대학을 막 졸업한 사람
- 수년 정도의 경험이 있는 개발자
- 숙련공이 되고 싶은 견습생 프로그래머
- 바로 여러분들

혹시, 읽으셨나요?


프로그래머의 길,
멘토에게 묻다
The Long Road
Expose Your Ignorance
Nurture Your Passion
Draw Your Own Map


실용주의
프로그래머


생각하는
프로그래밍

패턴이란?

- 특정 **맥락**에서 어떤 **문제**에 **되풀이**하여 적용할 수 있는 **방안**에다 **이름**을 붙이고 **설명**을 덧댄 것

- 앞으로 소개할 패턴들은 모두 **상황**, **문제**, **해결책**, **실천방안** 으로 이루어져 있다

패턴이 만들어진 과정

- 웹사이트에 패턴들을 올려 놓은 뒤 공개적인 댓글 형식으로 피드백을 받음
- 소프트웨어 개발 분야의 영향 있는 사상가들과 인터뷰를 진행해서 초기 패턴에 대한 의경 청취
- 경험이 많지 않은 실무자들과 인터뷰하고 그 사람들의 최근 경험에 비추어 패턴들을 검증

소프트웨어 장인정신이란 무엇인가?

- 구글로 검색하면 76,100개의 결과
  - 유용한 정의를 제고하는 경우는 별로 없음
- 피트 맥브린의 “Software Craftsmanship”
  - 소프트웨어 장인정신 vs 소프트웨어 공학 양자 택일을 강요
- 중세 유럽의 장인
  - 지금 우리의 현실에 맞지 않음

소프트웨어 장인정신 가치 기준들

- 당신은 더 나아질 수 있으며 당신이 제대로 할 준비가 되어 있다면 모든 것은 개선될 수 있다.
- 당신을 둘러싼 세계로부터 얻는 피드백을 바탕으로 항상 적응하고 변화해 갈 필요성
- 독단적이기보다는 실용적이려는 욕구
- 우리가 가진 지식을 쌓아두기만 하여 희소하게 만들기보다는 서로 나누는 편이 더 낫다는 신념
- 결과적으로 자신이 틀렸다고 증명될지라도 기꺼이 실험해 보고자 하는 자세

소프트웨어 장인정신 가치 기준들

- 자신의 운명에 대해 누가 답을 주기를 기다리기보다는 스스로 운명을 지배하고 책임지고자 하는 태도
- 그룹 보다는 개인에 초점 맞추기
- 우리와 다름을 끌어안고 가겠다는 서약
- 우리는 프로세스 중심적이기보다는 역량 중심적이다
- 어떤 기술을 배우려면 그 기술을 사용해서 자기 목표를 이루려고 하는 사람과 같은 방안에 있는 것이 제일이라는 점

견습생이 된다는 것

- “기본적으로 견습과정이란, 내가 지금 하고 있는 일을 항상 좀 더 좋고 세련되고 빠르게 해결하는 방법이 있을 거라고 생각하는 태도가 아닐까요. 견습과정은 당신이 발전해 가면서 더 나은 방법을 찾아가는, 그리고 더 좋고 세련되고 빠른 방법을 배우도록 만드는 사람, 회사 혹은 상황을 찾는 상태이자 과정이라고 봅니다.” Marten Gustafson
- 견습과정은 소프트웨어 장인으로서 걷게 되는 여정의 첫 시작
- 자신과 성장의 필요성에 집중하는 것이 곧 견습생

숙련공이 된다는 것

- 견습생과 마찬가지로 숙련공과 마스터도 자기 분야에서 배우고 성장하기를 위한 내적 집중은 계속 유지
- 추가적으로 전문가들을 잇는 연결, 팀 내부와 밖과의 소통에 집중
- 자신의 진전을 보여줄 더 큰 규모의 애플리케이션 포트폴리오를 작성하는 데 집중
- 포트폴리오를 다양화하고 심화시키려고 여러 프로젝트를 옮겨 다닌다
- 공동체 내의 자기 위상을 높이고자 하며, 마스터가 될 준비를 갖추기 위해 애쓴다

마스터가 된다는 것

- 견습생이나 숙련공이 맡던 역할은 물론이고, 소프트웨어 개발 분야 전체를 발전시켜 나가기 위한 일에 초점을 맞춤
- "절정의 기술과 테크닉의 습득"은 단지 시작일 뿐
- 마스터는 그런 기술을 다른 이들의 역량을 몇 배 향상시킬 수 있는 확대경으로 변화시켜야 한다.
- 마스터는 우수한 기술을 습득하고 사용하고 공유하는 것을 소프트웨어 장인에게 가장 중요한 일로 여긴다.

소프트웨어 장인정신 선언

- 2009년 3월, software_craftmanship 메일링 리스트 내의 오랜 논의 끝에 아래와 같은 선언문의 도안이 도출되었다.

- 우리는 동작하는 것을 넘어서 잘 짜인 소프트웨어에,
- 변화에 대응할 뿐 아니라 지속적으로 가치를 더하는 일에,
- 개인들 그리고 그 사이의 상호작용에 더해서 전문가들의 공동체에
- 고객과의 공동 작업 뿐 아니라 생산적인 파트너십에 가치를 둔다.
- 즉, 우리는 왼편의 항목을 추구함에 있어서 오른편의 내용이 필수불가결함을 알게 되었다

- http://agilemanifesto.org

하지만,

- 저자들의 소프트웨어 장인정신에 대한 비전
- 기꺼이 받아들이든, 더 낫게 만들든, 배척하든, 아니면 전혀 다른 길을 택하는 그것은 선택의 문제
- 하지만 여러분들이 바라는 성공의 길로 가는 데 충분히 도움이 될 만한 내용

이제 어디로 가는가

- 이제 패턴을 하나씩 배워가기 시작할 것
- 앞으로 다가올 몇 년 내에 어떤 패턴이 갑자기 여러분과 밀접한 연관성을 갖게 될 수도 있다.
- 그러다가 어느 날 갑자기 더 이상 적절하지 않다는 느낌이 들 수도 있다
- 패턴을 선택하고 조합해서 자신의 처한 유일한 상황에 적용할 주체는 여러분

첫 번째 언어

- 상황
  - 당신은 완전히 새내기이며 한 두 가지 프로그래밍 언어를 대략만 아는 정도다
- 문제
  - 어떤 특정한 프로그래밍 언어를 써서 팀의 다른 동료들과 동일한 품질 수준으로 결과물을 낼 수 있느냐에 내 일자리가 달려 있는 것 같다. 또는, 첫 일자리를 얻을 수 있을지 여부가 특정한 프로그래밍 언어에 얼마나 능숙하느냐에 달려 있는 상황이다

첫 번째 언어

- 해결책
  - 언어를 하나 선택하고, 그 언어에 능숙해져라
  - 이 언어가 앞으로 몇 년 동안 당신이 문제를 해결할 때 쓸 주력언어이다
  - 선택이 쉽지만은 않으며 여러 가지를 주의 깊게 고려해야 한다
- 실천 방안
  - 자신이 선택한 언어의 명세를 찾아서 읽어보라
  - 표준 라이브러리가 오픈소스 형태라면, 소스를 이용하여 통독하라.
  - 같이 일하는 사람들에게 첫 번째 언어를 어떻게 선택했는지 물어라

흰 띠를 매라

- 상황
  - 당신은 첫 번째 언어를 심도 있게 이해하게 되었고 어느 정도의 수준에 오른 것 같아 다소 느긋해졌다.
  - 동료들은 당신의 능력을 인정하고 있으며, 당신의 전문 분야 쪽 문제가 생기면 도와달라고 요청하는 정도가 되었다.
  - 당신은 자기 역량에 대해 자부심을 가지고 있다.

흰 띠를 매라

- 문제
  - 새로운 것을 배우려고 애를 써 보지만, 이전에 비해서 새 기술을 익히는 일이 왠지 더 힘들어진 것 같다.
  - 최선을 다해 노력해 보아도 자기 학습의 속도는 점점 더뎌지는 듯하다.
  - 당신은 자기 개발이 교착 상태에 빠진 것이 아닌가 두려워 진다.

흰 띠를 매라

- 해결책
  - 새로운 상황에 들어설 때는, 학습을 통해 얻은 자신감은 그대로 두면서 이전에 얻은 지식은 한편으로 밀어두어라.
  - '알지 못함'의 태도를 취해라.
  - 새로운 기술 분야를 배울 때 이런 접근법을 취하면 학습과정이 엄청나게 가속화 된다.
  - 높은 수준의 전문역량을 성취하여 마침내 자부심을 느끼는 한 사람의 프로그래머로서, 알지 못하는 영역으로 발을 들여놓고 스스로 어리석게 보여야 한다는 것은 고통스러운 일이 될 수 있다

흰띠를 매라

  - 두 번째 프로그래밍 언어를 배울 때 이렇게 하기 어렵다. 기술을 향상시키기 생산성을 희생하는 일이 처음일 것이기 때문이다.
  - 하지만 초보자의 마음가짐으로 접근해야 한다. 이것은 새로운 접근법에 통달한 다음 도약하기 위해서 당분간은 생산성이 다소 저하되는 것을 감수해야 한다는 의미다.

- 실천 방안
  - 특정 패러다임(imperative, object oriented, functional)으로 작성했던 프로그램을 하나 골라서 다른 패러다임에 속한 언어로 다시 구현해 보라.

열정을 드러내라

- 상황
  - 당신은 소프트웨어 개발이라는 기예에 대해 만족할 줄 모르는 열정과 호기심을 지녔다.

- 문제
  - 당신이 동료들에 비해서 얼마나 더 큰 열정을 지녔는지 의식하면서, 스스로 열정을 숨기고 지내게 되었다.

열정을 드러내라

- 해결책
  - 소프트웨어 개발자로서, 여러분들은 불가피하게 팀의 일부로 일해야 한다
  - 프로젝트를 완료해서 납품하는데 관심이 집중되어 잇다.
  - 그래서, 팀은 기술에 열정적이거나 의욕적이지 않다.
  - 하지만 견습생들의 열의와 기여에 대해 열린 팀이라면, 자유로운 상상력이나 열정 같은, 경험이 더 많은 개발자들이 기대하는 독특한 기질을 팀에 가져 올 수 있다.
  - 궁극 적으로, 열정을 드러내는 것이야말로 견습생이 맡아야 하는 몇 안 되는 책무이다

열정을 드러내라

- 실천 방안
  - 어떤 아이디어가 있었지만 실제로 제안하지는 않았던 가장 최근의 기억을 떠올려 보라
  - 제안할 대상으로 생각한 사람을 찾아가서 그 아이디어를 설명하라.
  - 그 사람이 미흡한 점을 지적한다면, 그런 점을 개선할 수 있게 도와 달라고 설득해 보라

구체적인 기술

- 상황
  - 당신은 현재보다 더 나은 학습기회를 얻고자 재능 있는 장인들이 모인 팀에 들어가서 할 수 있는 역할을 찾고 있다
- 문제
  - 유감스럽게도 그 팀은 업무에 직접적인 도움이 되지 않을 사람을 고용하는 위험은 감수하고 싶어 하지 않는다.
  - 더구나 그 팀에서는 단순 작업을 자동화 했다든지 해서, 당신이 간접적으로 기여하는 것마저도 여의치 않을 수 있다

구체적인 기술

- 해결책
  - 구체적인 기술을 습득해서 유지해라.
  - 특정한 도구와 기술 분야에 대해 뚜렷하고 입증할 만한 역량을 지녔다면, 일정 수준으로 성장할 때까지 팀에 간접적으로나마 기여할 수 있으리라는 신뢰를 얻기가 더 쉬울 것이다.

  - 구체적인 기술의 예
    - 대중적인 언어로 빌드 파일 작성하기
    - 하이버네이트, 스트럿츠, 스프링 같이 잘 알려진 오픈소스 프레임 웍에 대한 지식
    - 기초적인 웹 디자인, 자바스크립트, 언어의 표준 라이브러리

구체적인 기술

- 실천 방안
  - 당신이 우러러보는 역량을 가진 사람들의 이력서를 모아 보라.
  - 각 사람들의 다섯 가지 정도 대표적인 역량을 뽑아 보고, 그 중에서 어떤 역량이 지금 들어가고 싶어 하는 팀과 유사한 환경에서 바로 쓸모가 있을지 판단해 보라
  - 그런 기술들을 습득했음을 보일 수 있는 토이 프로젝트에 대한 계획을 세우라
  - 그리고 그 계획을 실행 하라.

NEXR
TOWARD OPEN PLATFORM

무지를 드러내라

- 상황
  - 당신은 소프트웨어 개발자로 채용한 사람들은, 자신이 하는 일이 무엇인지 잘 알고 있을 거라고 믿고 있다
- 문제
  - 관리자나 팀의 사람들은 당신이 잘 해 낼 거라는 확신을 갖기 원하지만, 실제로 당신은 몇몇 필수적인 기술에 그다지 익숙하지 않다.
  - 이런 일은 컨설턴트뿐 아니라 누구에게나 일어날 수 있다.
  - 당신이 팀에 합류하게 된 이유는 아마도 해당 비즈니스 영역이나 팀이 사용하는 기술 분야를 깊이 이해하고 있었기 때문일 것이다.
  - 아니면 그 일을 할 사람이 당신밖에 없었을 지도 모른다.

무지를 드러내라

- 해결책
  - 당신을 믿고 있는 사람들에게 학습과정도 소프트웨어 납품의 일부분임을 보여주어라
  - 산업화된 사회에서 유능하게 보이려는 욕구가 사람들의 마음 깊은 곳에 뿌리내리고 있다
  - 소프트웨어가 일상생활에 더 깊이 파고들수록 개발자라면 유능할거라는 기대도 점점 커져 간다
  - 하지만 여러분은 아직 미숙하기 때문에 모르는 영역이 상당히 많다
  - 당신을 둘러싼 많은 사람들(관리자,고객,동료)은 모두 소프트웨어를 납품해야 한다는 엄청난 압박에 시달리고 있다
  - 어떤 기능을 끝내려면 얼마나 걸리냐고 물어보는 사람들의 눈에서, 확신을 얻고자 하는 간절함을 읽을 수 있다.
  - 그런 사람들을 진정시키고 언제까지 해볼게요, 라며 안심시키는 일은 엄청난 부담으로 다가온다.

무지를 드러내라

- 소프트웨어 장인은 고객이나 동료와 맺은 튼튼한 관계를 통해서 자기 평판을 쌓아 간다
- 사람들이 듣고 싶어 하는 이야기를 해 주는 것은 튼튼한 관계를 쌓는 데 좋은 방법이 아니다
- 진실을 말하라
- 무엇을 원하는지 이제 당신이 이해하기 시작했고, 그것을 해낼 방법을 배워 가는 중이라고 알려주어라
- 그 사람들을 안심시켜야 할 때, 아 는 척 하기보다는 당신이 얼마나 잘 배울 수 있는지를 가지고 안심시켜라
- 평판은 어떤 지식을 고 있느냐가 아니라 학습하는 능력이 얼마나 좋은지를 기반으로 쌓여갈 것이다.

무지를 드러내라

- 실천 방안
  - 업무에 관해 정말로 이해되지 않는 것 다섯 가지를 적어 보라.
  - 그 목록을 다른 사람들이 볼 수 있는 곳에다 붙여 두어라.
  - 그리고 당신의 업무가 바뀔 때마다 그 목록을 갱신하는 습관을 들여라

무지에 맞서라

- 상황
  - 당신이 보유한 기술 목록에 일상 업무와 관련이 있는 기술이 빠져 있음을 알게 되었다.
- 문제
  - 숙달해야 할 도구나 기법들이 있기는 하지만 어떻게 시작해야 좋을지 모르겠다. 그 중 몇 가지는 이미 모두가 다 알고 있는 듯하고, 다른 사람은 당신도 당연히 알거라 기대하는 것 같다.

무지에 맞서라

- 해결책
  - 도구나 기법을 하나 고른 다음에 그것과 관련된 지식의 빈틈을 능동적으로 메워라
  - 이렇게 할 때는 자신에게 가장 효과적인 방법을 택해라

무지에 맞서라

- 무지를 드러내라 패턴의 실천 방안에서 언급된 항목들 각각에 대해서 학습을 하도록 노력하고, 학습이 진행된 다음에는 목록에서 하나씩 지워나가라
- 이렇게 해서 얻은 새 지식은 그 전까지 알아채지 못했던 빈틈을 드러낼 수도 있다.
- 그 빈틈을 자신의 목록에 잊지 말고 추가하여라

깊은 쪽

- 상황
  - 당신은 조금씩 안전하게 걸음을 옮겨 가는 것이 불만족스럽다. 당신은 안정된 상태가 아니라 판에 박힌 관습에 빠져 있을지 모른다는 두려움이 들기 시작한다.
  - 안정된 상태라면 더 높은 단계로 오르기 위해서 부지런히 연습하며 역량을 강화하고자 할 것이다.
  - 하지만 판에 박힌 듯한 상태에서는 무난한 수준의 능력일지라도 결국은 평범함으로 퇴보하게 된다.

깊은 쪽

- 문제
  - 당신은 기술과 자신감, 성공적인 업무 포트폴리오를 키워가야 한다.
  - 또한 더 큰 일을 통해 스스로 도전할 필요가 있다고 느낀다. 그것은 더 큰 규모의 프로젝트나 더 큰 팀, 더 복잡한 과제, 새로운 사업분야, 또는 새로운 장소와 관련될 수도 있다.

깊은 쪽

- 해결책
  - 깊은 쪽으로 뛰어들어라. 다 준비될 때까지 기다리다가는 아무 일도 못할 수가 있다
  - 당신에게 두드러지는 역할이나 어려운 문제가 주어진다면, 그 기회를 놓치지 말고 두 손으로 꽉 잡아라.
  - 두렵게 생각되는 일을 맡고, 능력을 넘어서는 듯한 일을 실제로 함으로써만 당신은 성잘 할 수 있다.
  - 물론 여기에는 위험이 따른다.
  - 하지만 실패한다고 해도 경력을 망치지 않고 위험을 감수해 볼 수 있는 기회가 많다.
  - 위험이란 기회의 다른 모습이다.

깊은 쪽

- 실천 방안
  - 당신이 참여했던 프로젝트 중에서 코드의 라인 수나 개발자의 수로 봐서 가장 규모가 크고 성공적이었던 프로젝트는 무엇인가?
  - 당신이 단독으로 작업한 것 중 가장 규모가 튼 코드는 무엇인가?
  - 프로젝트를 평가할 수 있는 척도를 찾아 이때껏 참여 했던 프로젝트를 평가해 보라.
  - 다음 프로젝트가 시작되면, 모든 프로젝트를 망라한 차트를 그려 놓고 새 프로젝트가 어디쯤에 위치하는지 점찍어 볼 수 있다
  - 이 차트에서 여러분의 경력이 어떤 방향으로 가고 있는지 알 수 있을 것이고, 이것에 기초해서 선택을 하기 시작할 수도 있을 것이다.

부숴도 괜찮은 장난감

- 상황
  - 경험이란 성공할 때만큼은 아니겠지만 실패로부터도 쌓인다
- 문제
  - 당신은 실패가 용납되지 않는 환경에서 일하고 있다.
  - 하지마 실패는 종종 무언가를 배울 수 있는 가장 좋은 방법이 된다.
  - 오직 과감한 일을 하고, 실패하고, 그 실패로 부터 학습하고, 또 다시 시도하는 것, 우리는 그렇게 해야 만 어려운 문제에 맞닥뜨렸을 때 성공해 내는 사람으로 성장할 수 있다.

부서도 괜찮은 장난감

- 해결책
  - 업무 때와 비슷한 도구를 써서, 업무 때 구축하는 시스템 범위에는 들지 않는 토이 시스템을 설계하고 구현하여 실패해 볼 수 있는 여지를 만들어라.
  - 경험이라 성공뿐 아니라 실패로부터 얻어진다.
  - 여러분에게는 실패해 볼 수 있는 다소 개인적인 공간이 필요하다
  - 공 세 개 수준의 저글러가 공연 중에 다섯 개짜리를 시도하지는 않듯이, 소프트웨어 개발자도 마음 놓고 실수를 할 수 있는 안전한 장소가 필요하다

부서도 괜찮은 장난감

- 실천방안
  - 좋아하는 도구들을 동원해서 높은 품질을 유지하면서도 세상에서 가장 단순한 위키를 만들어 보라.
  - 초기 버전은 텍스트 파일을 보거나 편집하는 정도의 간단한 사용자 인터페이스만 있으면 된다
  - 시간이 지나면서 기능도 더 추가하고 기존의 위키들과 차별화할 흥미로운 방법도 찾을 수 있을 것이다.
  - 기존 구현 방식에 얽매이지 말고, 당신의 직업적인 관심을 이끌도록 해라.
  - 검색 엔진에 관심이 있다면 위키에 랭킹 알고리즘이나 태깅에 관련된 실험적인 기능을 넣을 수 있다.

소스를 활용하라

- 상황
  - 오픈소스에 세계에 처음 입문한 사람들은 종종 자기가 올린 질문에 “Use the source, Luke” 라는 답변이 돌아오는 것을 경험한다.
  - 이 말은 소프트웨어에 대한 기본적인 진실을 말해준다. 결국은 코드가 최종 결정자라는 것이다.
  - 프로그래머의 의도란, 코드가 그 의도를 제대로 반영하지 못한다면 공허한 것 되어 버린다.
  - 시스템을 진정으로 이해한다는 것은 오로지 코드를 읽어 보아야만 가능하다.

소스를 활용하라

- 문제
  - 공부하고 모방할 좋은 사례가 없다면, 연습, 연습 또 연습 패턴은 스스로 자각하지 못하는 나쁜 습관을 더욱 굳히는 결과만 가져올 것이다.
  - 다른 사람의 신발을 신고 걸어보지 않는다면, 당신은 모든 신발 속에는 돌멩이가 들어 있는 거라 믿게 될 수도 있다.
  - 그렇다면, 주변에 좋은 코드와 나쁜 코드를 구별할 만한 사람이 없는 환경에서, 당신이 짜 놓은 코드가 제대로 짠 것인지 어떻게 알 수 있을까?

소스를 활용하라

- 해결책
  - 다른 사람들의 코드를 찾아서 읽어라
  - 당신이 매일 사용하는 도구나 애플리케이션부터 시작해 보라
  - 견습생을 주저하게 만드는 것은 도구들을 만든 사람은 왠지 당신과 다를 것 같고 더 특출하며 훌륭할 거라 믿는 일이다.
  - 그 사람들의 코드를 읽고 그들처럼 프로그래밍 하는 법을 배우고. 당신을 둘러싼 인프라를 만들어 낸 사고 과정이 어떤 것이었는지 이해하게 된다.

소스를 활용하라

- 실천방안
  - 알고리즘이 복잡한 오픈소스 프로젝트를 하나 골라보라.
  - 예를 들어 subversion, Git, mercurial 같은 소스 관리 시스템
  - 소스를 둘러보면서, 생소한 알고리즘이나 자료구조, 설계 사상 같은 것들을 기록해 두라
  - 그 프로젝트의 구조를 기술하면서 새로 알게 된 아이디어들을 적은 블로그 포스트를 써라

배운 것을 기록하라

- 상황
  - 당신은 같은 교훈을 계속 되풀이해서 배운다. 이 교훈들은 도무지 몸에 붙지 않는 것 같다. 당신은 또 다시 SQL 계층 구조를 모델링하거나 팀에게 패턴에 대해서 소개하고 있는 자신을 흔히 발견한다. 과거에 상당히 비슷한 일들을 했던 기억이 나기는 하지만, 세부적이 ㄴ사항은 어슴푸레하다.
- 문제
  - 과거로부터 배우지 못하는 이들은 같은 일을 되풀이하도록 되어 있다.

배운 것을 기록하라

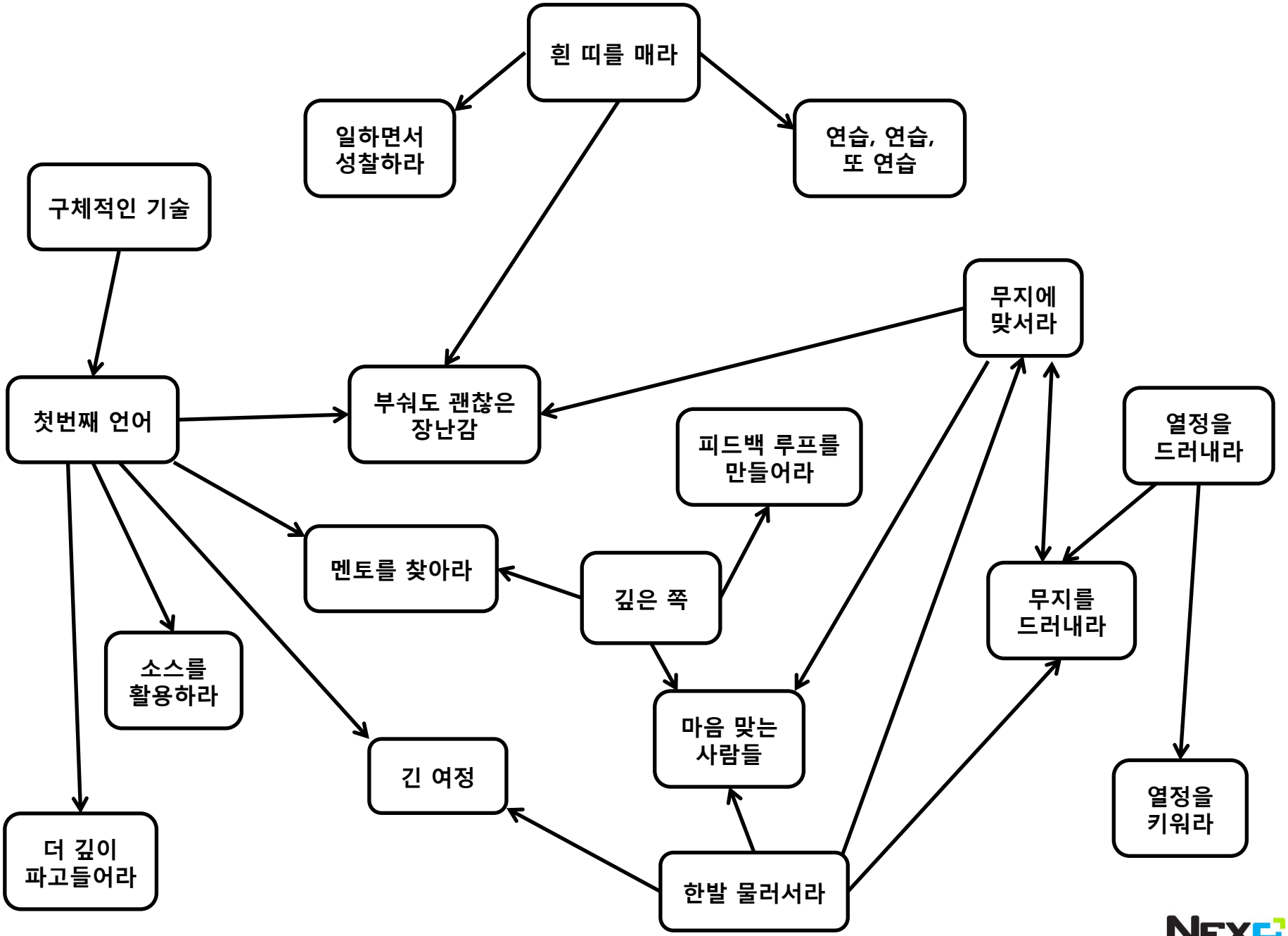
- 해결책
  - 개발 로그를, 위키, 블로그 등으로 남겨라
  - 배운 것을 기록만 하고 그냥 잊어버리는 덫에 빠지지 않게 노력하라
  - 과거에 썼던 글을 정기적으로 다시 읽어라
  - 자신의 일지를 리뷰함으로써 당신은 미래를 만들어 내기 위해 과거와 현재를 재배치 할 수 있다.

배운 것을 공유하라

- 상황
  - 당신은 얼마 동안 견습생이었다. 당신은 몇 가지 업무적인 지식을 알고 있으며 사람들은 당신을 지식의 소스로 여기기 시작한다.
- 문제
  - 지금까지 당신은 장인으로서 자신의 발전에만 집중해 왔다.
  - 하지만 숙련공이 되려면 효과적으로 의사소통하며 다른 사람들이 속도를 내도록 이끄는 능력을 갖출 필요가 있다.

배운 것을 공유하라

- 해결책
  - 배운 것을 정기적으로 공유하는 습관을 겹습과정 초기에 들여 놓아라.
  - 블로그를 운영하는 형태가 될 수도 있고, 마음에 맞는 사람들끼리 점심 도시락 모임을 갖는 것이 될 수 있다.
  - 컴퍼런스에서 발표를 하거나 학습 중인 여러 가지 기술 분야와 기법에 대해 튜토리얼을 쓸 수도 있다.
- 실천방안
  - 블로그에 글을 써라
  - 워크샵의 발표자가 되라
  - 세미나를 해라



이번에 다루지 못한 패턴들

- 예술보다 기예
- 지속적인 동기부여
- 열정을 키워라
- 한발 물러서라
- 자신만의 지도를 그려라
- 직위를 지표로 이용하라
- 전장에 머물러라
- 또 다른 길
- 가장 뒤떨어진 이가 되라
- 팔꿈치를 맞대고
- 바닥을 쓸어라
- 멘토를 찾아라
- 마음에 맞는 사람들
- 긴 여정
- 능력의 폭을 넓혀라
- 연습, 연습, 또 연습
- 일하면서 성찰하라
- 피드백 루프를 만들어라
- 실패하는 법을 배워라
- 독서목록
- 꾸준히 읽어라
- 고전을 공부하라
- 익숙한 도구들

마치며

- 소프트웨어 공학에서 많은 연구성과들이 나오고 최선을 다하고 있지만, 우리 분야에서 아직도 프로젝트 성패를 결정하는 가장 중요한 요소는 개인의 역량
- 역량은 전산학의 지식의 정도, 개발 과정의 효율성, 풍부한 경험을 말하는 것이 아님
- 제대로 동작하는 소프트웨어를 출시하는 데 필요한 모든 것의 총합
- 우리가 보유한 역량 수준과 당면 과제를 풀기 위해 필요한 역량 수준은 흔히 일치하지 않는다.

마치며

- 견습과정은 당신의 경력에서 다른 무엇보다도 자신의 성장에 초점을 맞추게 되는 시기
- 이 시기가 끝나면, 무엇이 중요한지 다시 정리할 필요가 있다
- 공부 할 것은 여전이 많지만 여러분은 더 이상 견습생이 아니며, 당신의 우선순위는 이제 자신이 아닌 다른 이들을 향해야 한다
- 고객, 동료, 그리고 당신이 속한 커뮤니티로.

참고

- 프로그래머의 길, 멘토에게 묻다
  - 데이브 후버, 애디웨일 오시나이 지음, 강주입 옮김, 인사이트