소프트웨어 공학개론

Introduction to Software Engineering

2022-1학기 (Spring)
선문대학교 AI소프트웨어학과

3월 10일 (목)

강의 목차

Part II. 활용

- 소프트웨어와 펌웨어
- 오픈소스 소프트웨어
  - 대표적인 대규모 소프트웨어 개발의 사실상 표준 (De facto standard) 사례

Part III. 실습

- 과제 없음

소프트웨어와 펌웨어

개요

- 소프트웨어
  - 응용 프로그램과 데이터와 같이, 컴퓨터의 하드웨어 상에서 구동되거나 처리되는 무형물을 하나로 포괄하는 말을 뜻함. 예) 컴퓨터 = 하드웨어 + 소프트웨어 + 펌웨어

- 펌웨어로서의 BIOS (Basic Input/Output System)
  - 메모리, 디스크, 모니터와 같은 주변기기와 같은 하드웨어 제어를 하며, 하드웨어간 통신(예: 정보 전송)을 관리하는 프로그램
    - 운영체제에게 컴퓨터 제어권을 반환하거나 요청한 표준 입출력 및 디스크 요청을 처리함
  - 운영체제에서 BIOS Function Call (Hardware Interrupt 번호)
  - ROM-BIOS 라고도 말함
    - EEPROM (Electrically Erasable Programmable Read-Only Memory)
    - NAND Flash 에 저장 (1990년대)
    - NOR Flash (고비용)
      - Flash Translation Layer

소프트웨어와 펌웨어

개요

- CMOS (Complementary Metal-Oxide Semiconductor)
  - Patent: Frank Wanlass (1967)
    - 상보성 금속 산화막 반도체 (相補性 金屬 酸化膜 半導體)
    - 사용하는 이유는 전력 소비가 매우 적어 낮은 전력 손실, 빠른 속도, 높은 노이즈 마진(전기 전압 불안정에 대한 안정성) 제공
  - 소용량 RAM 역할
    - CMOS는 하드웨어 데이터, BIOS 설정을 저장하기 위한 메모리
    - 컴퓨터 전원을 인가 후 나타나는 부팅 화면(boot screen)에서 CMOS 셋업을 할 수 있음
    - 컴퓨터가 켜지지 않더라도 모든 데이터를 적절하고 동기적인 구성으로 저장할 수 있음
      - 리튬배터리(수은전지) CR2032
      - 날짜 및 시간 설정, 부팅 설정, 하드웨어 설정, 설정 초기화, 부팅 우선순위 설정
      - 제조사가 설정한 속도(clock rate)를 초과하도록 설정하는 오버클록킹(overclocking) 설정

오픈소스 소프트웨어

목차

오픈소스란 무엇인가

개요

- 오픈소스
  - 원래 오픈소스 소프트웨어(Open Source Software, OSS)를 뜻하는 용어

- 오픈소스 소프트웨어
  - 부속물
    - 소스 코드, 문서 (매뉴얼, 아키텍처), 이미지, 저작권 고지사항, 실행가능한 바이너리 등
  - 공개적으로 액세스되며, 누구나 자유롭게 확인, 수정, 배포 **"가능한"** 코드
    - 가능하다는 의미는 공개하는 주체에서 "배포(공표)"권한을 결정하므로, 무조건적인 사항이 아님
      - 배포(공표) 권한을 제한하는 경우는 주로 "상업적 이용(Commercial Use)"에 해당하는 경우가 많음
    - 따라서, 오픈소스 소프트웨어 마다 가지는 "저작권 고지사항"을 주기적으로 확인할 필요가 있음

오픈소스란 무엇인가

개요

- 오픈소스 소프트웨어
  - 동료 평가(peer review) 와 커뮤니티 기반 프로덕션
    - 분산된 동시에 협업 방식으로 개발
  - 단일 작성자 또는 기업이 아닌 커뮤니티가 개발
    - 독점적 소프트웨어보다 저비용, 유연성, 지속성 우위
  - 단순한 소프트웨어 프로덕션을 넘어서는 작업 방식이자 하나의 흐름
  - 오픈소스 소프트웨어의 가치와 분산된 프로덕션 모델을 활용
    - 커뮤니티와 업계가 당면한 문제를 해결할 새로운 방법을 찾는데 도움

인터넷과 역사를 함께하는 오픈소스

1950년대와 1960년대 초기

- 인터넷 기술과 통신 네트워크 프로토콜을 개발하던 연구원들
  - 개방적인 협업 환경에서 연구 진행
- 이후 현대 인터넷의 기반이 된 ARPANET(Advanced Research Projects Agency Network)
  - 동료 평가와 열린 피드백 프로세스를 권장
  - 사용자 그룹은 서로 소스 코드를 공유 및 구축
  - 포럼은 오픈 커뮤니케이션 및 협업을 위한 논의 촉진 및 표준 개발에 도움

1990년 초반

- 인터넷이 탄생할 무렵, 아래 4가지 가치가 기반으로 자리 잡음
  - 협업
  - 동료 평가
  - 커뮤니케이션
  - 개방성

오픈소스 개발 모델의 작동 방식

오픈소스 개발 모델

- 오픈소스 커뮤니티 프로젝트에서 오픈소스 프로젝트를 개발하기 위한 프로세스

작동 방식

- 이러한 소프트웨어는 오픈소스 라이센스로 출시되므로 누구든지 소스 코드를 보고 수정 가능

- 대부분의 오픈소스 프로젝트가 호스팅되는 GitHub에서 리포지토리에 액세스하거나 커뮤니티 프로젝트에 참여 가능

- 널리 사용되는 오픈소스 프로젝트의 예
  - Linux®, Ansible, Kubernetes(쿠버네티스, 쿠버네츠)

오픈소스 개발 모델의 작동 방식

Red Hat의 사례

- 오픈소스 소프트웨어 개발 모델을 사용해 엔터프라이즈 오픈소스 제품 및 솔루션 개발
- 개발자들은 IT 스택 전반의 수백 개 오픈소스 프로젝트에 활발히 참여
- 고객 요구사항(일부 또는 전체)를 충족하는 "커뮤니티 기반 오픈소스 소프트웨어"로 만들어 참여 유도
  - 이후 보안 강화, 취약점에 패치 적용, 새로운 엔터프라이즈 기능 추가
  - 그런 다음, 커뮤니티 전체의 이익을 위해 "개선 사항"을 기존 프로젝트 재적용
    - **하위호환성**(Backward Compatibility) 및 **백포팅**(Backporting)
- 소프트웨어를 사용하는 고객
  - 피드백 제공
  - 버그 리포트 제출
  - 요구사항이 변하는 경우 추가 기능 요청
  - 이러한 고객의 의견은 제품 개발에서 가이드의 근거자료로 활용됨

Linux와 오픈소스

Linux

- GNU GPL(General Public License)을 사용하여 출시된 무료 오픈소스 운영 체제(OS)
- 최대의 오픈소스 소프트웨어 프로젝트
- "Linux 운영 체제는 Unix의 원칙과 설계를 기반으로 했던 MINIX 운영 체제를 대체할 무료 오픈소스 버전으로 제작되었습니다."
  - 이후 페이지에서 다루는 "Linux와 Minix 논쟁" 참고
- Linux
  - 소프트웨어 사용에 대한 제한을 방지하는 오픈소스 라이센스 하에 릴리스됨
  - 누구든지 소스 코드를 실행, 연구, 수정, 재배포 가능
  - 동일한 라이센스가 유지되는 한 수정한 코드의 복사본 판매 가능
    - 유의사항
      - GPL과 MIT, Apache 라이선스
      - PyQt 와 PySide 의 라이선스 비교

Linux와 Minix 논쟁

Tanenbaum-Torvalds Debate (논쟁), “Linux is obsolete”, 1992.

- Andrew Stuart Tanenbaum (Age 46)
  - 네델란드 암스테르담 자유대학교 전산학 교수
  - OSDI (Operating Systems: Design and Implementation) 저자
  - Minix (교육용 Unix)
    - Microkernel
    - http://www.minix3.org

- Linus Benedict Torvalds (Age 23)
  - 핀란드 헬싱키대학교 대학원 컴퓨터과학 석사 (1996)
    - Linux 커널 코드 크기
      - 1991년 약1만 라인
      - 1999년 (약 180만 라인)
    - 동대학 명예박사학위 (2000)
  - Linux Founder 및 현재 커널 개발 최고 설계자
    - Monolithic kernel
    - https://kernel.org

Andrew Stuart Tanenbaum
(1944 ~), Age 77

VS.

Linus Benedict Torvalds
(1969 ~), Age 53

Linux와 Minix 논쟁

Tanenbaum-Torvalds Debate (논쟁), “Linux is obsolete”, 1992.

- Appendix A: The Tanenbaum-Torvalds Debate
  - Open Sources: Voices from the Open Source Revolution
    - https://www.oreilly.com/openbook/opensources/book/appa.html

- 당시 시대 상황
  - Intel 80386 이 지배적, 486은 미출시.
  - Microsoft는 DOS와 DOS용 Word를 판매하는 작은 회사
    - CP/M (Intel 8080/8085용 텍스트 기반 운영체제)
  - Lotus 123은 Spreadsheet 와 WordPerfect 로서 워드 프로세싱 시장을 지배하고 있었음
  - DBASE는 데이터베이스 공급업체로 지배적이었으며,
  - Netscape, Yahoo, Excite 등의 회사는 아직 존재하지 않았음

Linux와 Minix 논쟁

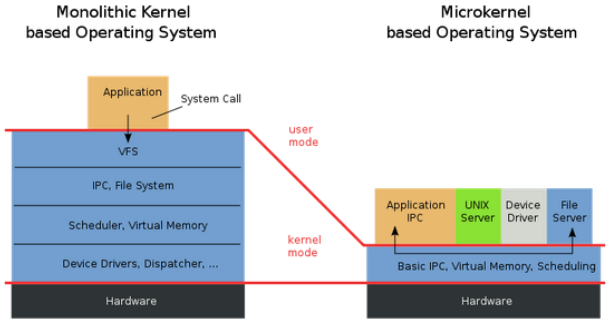
Tanenbaum-Torvalds Debate (논쟁), “Linux is obsolete”, 1992.

- 논쟁 경위
  - 1991년 12월에 Linus Torvalds 가 Linux 0.0.1 (1만라인)을 출시
  - 1992년 1월 19일에 유즈넷 토론그룹 중 하나인 뉴스그룹(alt.os.linux, aka comp.os.linux)에 기고
- 이후, 토론의 출발점
  - Andy Tanenbaum이 “리눅스는 구식이다” 라고 미닉스 뉴스그룹 (comp.os.minix) 기고
- 뉴스그룹은 메일링리스트 형태이며, 논쟁기간은 1992년 1월 29일부터 2월 10일까지 진행
- 논쟁 내용
  - Andy Tanenbaum과 Linus Torvalds와의 운영체제 커널의 설계 방식을 놓고 벌인 지식논쟁
- 등장인물 중
  - Ken Thompson (Unix Founder, C Language Founder)
    - C language (A language → B language → Combined)
  - David Miller (Linux Kernel Hacker)
  - 기타 유명한 인물들이 등장

Linux와 Minix 논쟁

Minix vs. Linux (Trade-off 고려 필요)

- Minix → Microkernel
  - Pros) 컴포넌트(기능) 추가/삭제가 용이한 모듈성 보장, self-healing 특성 있음
  - Cons) 각 컴포넌트간 통신 오버헤드가 있음
- Linux → Monolithic kernel
  - Pros) 커널모드내 각 컴포넌트(기능)간 통신 효율적 → 빠름
  - Cons) Device Driver 추가/삭제하려면, 커널 리빌드 필요, 컴포넌트 장애가 전체시스템에 영향
- 승자는 없음 (No winners)
  - 고성능 가상 네트워킹 솔루션, 6WIND의 200Gbps 성능 달성 (Network Packet Processing)
  - Kernel의 기능을 Userspace로 재배치하여 성능 향상 구조, https://www.6wind.com



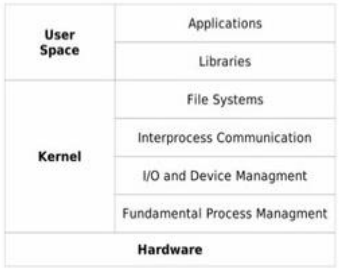


Figure 1: Monolithic kernel based operating system

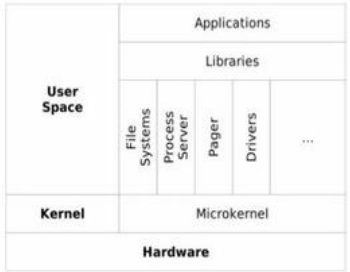


Figure 2: Microkernel based operating system

소프트웨어 철학과 품질

소프트웨어 품질

- 설계판단 및 요구사항 분석 등의 과정을 거쳐 결정하게 되는 Trade-off
  - 일장일단, 거래(교환, 하나를 가져오고, 다른 하나를 내어주어야 하는), 반비례
  - 세상에 공짜 점심 없다(There is no such thing as a free lunch)
- 소프트웨어 개발하는 사람들의 철학과 ~ism(~주의)도 필요하고, 소프트웨어의 품질(특성)에 영향을 줌
  - 6WIND: Speed Matters
  - JetBrains: Right Tool Matters
  - 삼성전자: 일등주의, 교육보국, 초일류, 초격차 경영
  - 대우전자: 탱크주의 (소프트웨어는 아니었지만, 전체적인 기류)

오픈소스 소프트웨어에도 철학이 있지 않을까?

- 독점과 비공개/비밀로부터의 자유
  - "발전"과 영향이 있음
    - 대규모 소프트웨어 대해서, 혼자서 하기보다, 다함께 하며 가치 창출 → 협업 도구 즉, 소프트웨어 필요
    - 많은 참여자들의 관심(감시/모니터링), 이용 및 테스트에 따른 피드백과 소프트웨어 품질 향상

프리, 비공개, 오픈소스 소프트웨어의 차이점

프리 소프트웨어 운동 (유의: 프리웨어와 구분 필요)

- 오랫동안 오픈소스 소프트웨어는 "프리 소프트웨어"라는 초기의 개념을 유지
- 프리 소프트웨어 운동 (참고: Copyleft 운동)
  - 1983년 Richard Stallman이 GNU Project를 통해 확립
  - 소스 코드를 보고, 수정하고, 재배포하는 데 있어, 아래와 같은 생각에서 출발
    - 사용자의 자유가 가장 중요
    - 사용자가 필요한 방식으로 작업할 수 있게 해야 함
- 프리 소프트웨어는 독점 또는 "비공개 소스" 소프트웨어와 대응 관계임

비공개 소스 소프트웨어

- 철저히 보호되며, 소스 코드를 보유한 사람만 해당 코드에 접근 가능한 법적 권한을 가짐
- 비공개 소스 코드는 법적으로 변경 또는 복사 불허
- 사용자는 “제작 의도”대로만 사용해야 함
- 새로운 용도에 맞게 수정하거나 커뮤니티와 공유할 수 없음

프리, 비공개, 오픈소스 소프트웨어의 차이점

프리 소프트웨어의 정체성 혼란

- "프리 소프트웨어"라는 이름은 많은 혼란을 가져왔음 (프리웨어와 구분 필요)
  - 프리 소프트웨어는 마음대로 소유하는 것이 아니라,
    원하는 방식으로 자유롭게 사용할 수 있다는 데 초점을 맞춤
    - 이는 "프리란 자유에 초점을 맞춘 것이지, 무료란 의미가 아니기 때문"

"오픈소스"라는 용어를 만든 Christine Peterson

- '프리 소프트웨어'를 '오픈소스'라는 용어로 대체함으로써 이러한 문제를 해결하려 했음
- *"이전에 사용되던 '프리 소프트웨어'라는 용어는*
  정치적 함의에 문제가 있는 것이 아니라,
  새로운 참여자들이 가격에 더 초점을 맞춘다는 데 문제가 있었습니다.
  소스 코드의 핵심적 이슈에 초점을 맞추고
  개념을 새로 접하는 사람들이 ***혼란을 일으키지 않을 용어가 필요****했습니다."*

프리, 비공개, 오픈소스 소프트웨어의 차이점

Christine Peterson

- '프리 소프트웨어'라는 용어를 '오픈소스'로 대체하자는 아이디어를 (자유소프트웨어 진영에) 제시
  - Richard Stallman의 자유소프트웨어재단
    - 소프트웨어가 공유되어 협력적이고 개방적인 방식으로 수정 가능하게 되었을 때, 더 발전한다는 것을 세상에 알리고자 했음
    - 그래야 소프트웨어를 새롭고 더 나은 방식으로, 벤더에 종속되는 일 없이 더욱 유연하고 저렴하고 지속적으로 활용할 수 있기 때문
    - Eric Raymond (http://www.catb.org/~esr/resume.html)
      - 1997년에 많은 이들에게 영향을 미친 에세이인 "성당과 시장"(The Cathedral and the Bazaar)에서도 주장함

1998년, 이 에세이에 대한 대응(공감)의 일환

- Netscape Communications Corporation
  - Mozilla 프로젝트를 오픈소스화하여 소스 코드를 프리 소프트웨어로 공개
  - 이후 Mozilla Firefox와 Thunderbird의 기반으로 사용됨

프리, 비공개, 오픈소스 소프트웨어의 차이점

오픈소스와 프리 소프트웨어라는 용어의 분리

- Netscape에서 오픈소스에 대한 공개적인 지지 발표
- 이후, 커뮤니티
  - "프리 소프트웨어 운동"의 실질적인 비즈니스적 측면을 강조할 수 있는 방법 강구 부담감 증폭
- 이로 인해, 오픈소스와 프리 소프트웨어라는 용어가 분리됨
  - '오픈소스': 프리 소프트웨어의 방법론과 프로덕션 및 비즈니스 측면을 옹호하는 용어
  - '프리 소프트웨어': 사용자의 자유라는 철학적인 개념을 강조하기 위한 용어

- 1998년 초 Open Source Initiative(OSI)가 설립
  - 오픈소스라는 말이 공식화
  - 업계에서 널리 통용되는 일반적인 정의가 확립
- 1990년대 후반에서 2000년대 초반까지 오픈소스 운동은 꾸준히 경계와 의심의 대상
  - (아마도) 저작권 침해 소지, 상업용 수익 악화, 사회적 무료라는 인식 등 현실성 취약한 측면 우려

- 오늘날은 소프트웨어 프로덕션의 변방에서 업계 표준 (de facto standard)이 되었음

오픈소스의 가치

독점 소프트웨어가 아닌 오픈소스를 선택하는 데는 가장 일반적인 이유 (7가지)

- **동료 평가**
  - 오픈소스 코드는 동료 프로그래머에 의해 적극적으로 검토 및 개선
    - 소스 코드에 누구나 액세스 가능
    - 활발한 오픈소스 커뮤니티
    - 침체 상태에 놓인 비공개 코드보다 훨씬 살아 있는 코드라고 간주됨

- **투명성**
  - 벤더에 의존할 필요 없이 직접, 이를 확인 및 추적 가능 (Traceability, 추적성 보장)
    - 오픈소스를 사용하면 해당 코드에서 정확하게 파악 가능한 내용
      - 어떤 종류의 데이터가 어디로 이동했는지
      - 어떤 변경 사항이 있었는지

오픈소스의 가치

독점 소프트웨어가 아닌 오픈소스를 선택하는 데는 가장 일반적인 이유 (7가지)

- **안정성**
  - 독점 코드
    - 해당 코드의 업데이트, 패치, 작업을 제어하는 단일 작성자 또는 기업에 의존해야 함
  - 오픈소스 코드
    - 활발한 오픈소스 커뮤니티를 통해 지속적으로 업데이트되므로 오래 지속됨
      - Maintainability, Sustainability
    - 오픈 표준과 동료 평가를 통해 테스트 역시 적절한 방식으로 자주 이루어짐

- **유연성**
  - 오픈소스는 수정을 강조 (빈번한 수정이 문화 그 자체임)
    - 오픈소스 코드를 사용해 자신의 비즈니스 또는 커뮤니티에서 겪고 있는 고유한 문제를 해결할 수 있음
    - 해당 코드를 특정한 방식으로만 사용해야 한다는 법이 없으므로, 다양한 접근 방식 검토 가능함
    - 새로운 솔루션을 구현할 때, 커뮤니티의 도움을 받고 동료 프로그래머에게 검토받을 수도 있음

오픈소스의 가치

독점 소프트웨어가 아닌 오픈소스를 선택하는 데는 가장 일반적인 이유 (7가지)

- **비용 절감**
  - 오픈소스 코드 자체는 무료임
  - Red Hat과 같은 기업을 활용하는 경우 A/S 비용 있음
    - 지원 서비스, 보안 강화, 상호 운용성 관리와 같은 부분에 비용 지불 발생 가능

- **벤더 종속성 없음**
  - 오픈소스 코드를 언제 어디에든 가져가 원하는 목적으로 사용할 수 있는 사용자 중심의 자유 만끽

- **오픈 협업**
  - 활발한 오픈소스 커뮤니티 덕분에 하나의 관심 그룹 또는 기업에 의존하지 않아도 됨
  - 다양한 지원, 리소스, 관점을 접할 수 있음

오픈소스 전문 기업의 장단점

Red Hat 등 오픈소스 전문 기업의 솔루션을 선택하는 이유

- Red Hat
  - 세계 최대의 오픈소스 기업
    - 글로벌 오픈소스 비즈니스 구조로 매출 1조원 이상을 달성
  - 활발한 오픈소스 커뮤니티
  - 언제 어디에든 가져갈 수 있는 오픈소스 코드 및 공개 저장소 (Github, Bitbucket 등의 Open Repository)

- 일반적인 오픈소스 전문 기업의 솔루션을 선택하는 이유
  - 장점
    - 유지보수성
    - 자체 전문 개발인력이 절대적으로 요구되지 않음
  - 단점
    - 연단위 구독 라이선스 비용 발생 가능 (개발자에게는 개발자 라이선스를 통해 무료 제공)
    - 장애발생시 문의 및 대응시간 장기화될 수 있음

References

- Red Hat
  - What is open source
    - https://www.redhat.com/en/topics/open-source/what-is-open-source
  - Opensource.com
    - https://opensource.com
- IBM
  - Open projects
    - https://www.ibm.com/opensource/open-projects/
  - What is open source software
    - https://www.ibm.com/topics/open-source

Talktime

Do you have any questions?