제품 소프트웨어 패키징

제품 소프트웨어 패키징

- 패키징
  - ✓ 모듈 별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것
  - ✓ 개발자가 아니라 사용자를 중심으로 진행
  - ✓ 소스 코드는 향후 관리를 고려하여 모듈화하여 패키징
- 소프트웨어 패키징 작업 순서
  - ✓ 기능 식별
  - ✓ 모듈화(Modularity)
  - ✓ 빌드(Build) 진행
  - ✓ 사용자 환경 분석
  - ✓ 패키징 및 적용 시험
  - ✓ 패키징 변경 개선
  - ✓ 배포

릴리즈 노트

- 릴리즈 노트(Release Note)
  - ✓ 개발 과정에서 정리된 릴리즈 정보를 최종 사용자인 고객과 공유하기 위한 문서
  - ✓ 작성 항목
    - ❑ Header(머릿말): 릴리즈 노트 이름, 소프트웨어 이름, 릴리즈 버전, 릴리즈 날짜, 릴리즈 노트 날짜, 릴리즈 노트 버전 등을 표시
    - ❑ 개요
    - ❑ 목적
    - ❑ 문제 요약
    - ❑ 재현 항목
    - ❑ 수정/개선 내용
    - ❑ 사용자 영향도
    - ❑ SW 지원 영향도
    - ❑ 노트
    - ❑ 면책 조항
    - ❑ 연락처

릴리즈 노트

- ❖ 릴리즈 노트(Release Note)
  - ✓ 릴리즈 노트 작성 순서
    - 모듈 식별 : 모듈별 빌드 수행 후 릴리즈 노트에 작성 될 내용 확인
    - 릴리즈 정보 확인: 릴리즈 노트 및 소프트웨어 이름, 릴리즈 버전 및 날짜, 노트 날짜 및 버전 등 확인
    - 릴리즈 노트 개요 작성 : 소프트웨어 및 변경사항 전 체에 대한 간략한 내용 작성
    - 영향도 체크 : 버그나 이슈 관련 내용 또는 해당 릴리즈 버전에서의 기능 변화가 다른 소프트웨어나 기능 을 사용하는데 미칠 수 있는 영향 기술
    - 정식 릴리즈 노트 작성: Header(머릿말), 개요, 영향 도 체크 항목을 포함하여 정식 릴리즈 노트에 작성 될 기본 사항 작성
    - 추가 개선 항목 식별: 추가 버전 릴리즈 노트 작성이 필요한 경우 추가 릴리즈 노트 작성

디지털 저작권 관리

- 디지털 저작권 관리(DRM)
  - ✓ 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도 한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술
  - ✓ 창작자가 가지는 배타적 독점적 권리로 타인의 침해를 받지 않을 고유한 권한
  - ✓ 디지털 저작권 관리의 구성 요소
    - Clearing House: 저작권에 대한 사용 권한, 라이선스 발급, 암호화된 키 관리, 사용량에 따른 결제 관리 등을 수행하는 곳
    - Contents Provider: 콘텐츠를 제공하는 제공자 또는 저작권자
    - Packager: 콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
    - Contents Distributor: 콘텐츠를 유통하는 곳이나 사람
    - Customer: 콘텐츠를 구매해서 사용하는 주체
    - DRM Controller: 배포된 콘텐츠의 이용 권한을 통제하는 프로그램
    - Security Container: 콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

디지털 저작권 관리

- 디지털 저작권 관리(DRM)
  - ✓ 디지털 저작권 관리의 기술 요소
    - 암호화(Encryption): 콘텐츠 및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
    - 키 관리(Key Management): 콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
    - 암호화 파일 생성(Packager): 콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
    - 식별 기술(Identification): 콘텐츠에 대한 식별 체계 표현 기술
    - 저작권 표현(Right Expression): 라이선스의 내용 표현 기술
    - 정책 관리(Policy Management): 라이선스 발급 및 사용 에 대한 정책 표현 및 관리 기술
    - 크랙 방지(Tamper Resistance): 크랙에 의한 콘텐츠 사용 방지 기술
    - 인증(Authentication): 라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

소프트웨어 설치 메뉴얼 작성

- 소프트웨어 설치 매뉴얼 작성
  - ✓ 소프트웨어 설치 매뉴얼은 개발 초기에서부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 설명서와 안내서
  - ✓ 설치 매뉴얼 작성 순서
    - 기능 식별 : 소프트웨어의 개발 목적과 주요 기능을 흐름 순으로 정리하여 기록
    - UI 분류 : 설치 매뉴얼을 작성할 순서대로 UI를 분류 한 후 기록
    - 설치 파일/백업 파일 확인 : 폴더 위치, 설치 파일, 백 업 파일 등의 개별적인 기능을 확인하여 기록
    - Uninstall 절차 확인 : 직접 Uninstall을 수행하면서 그 순서를 단계별로 자세히 기록
    - 이상 Case 확인 : 설치 과정에서 발생할 수 있는 다양한 Case를 만들어 확인하고 해당 Case에 대한 대처법을 자세하게 기록
    - 최종 매뉴얼 적용 : 설치가 완료된 화면과 메시지를 캡쳐하여 추가한 후 완성된 매뉴얼을 검토하고 고객 지원에 대한 내용 기록

소프트웨어 사용자 매뉴얼 작성

- 소프트웨어 사용자 매뉴얼 작성
  - ✓ 소프트웨어 사용자 매뉴얼은 사용자가 소프트웨어를 사용하는 과정에서 필요한 내용을 문서로 기록한 설명서와 안내서
  - ✓ 사용자가 소프트웨어 사용에 필요한 절차, 환경 등의 제반 사항이 모두 포함되도록 작성
  - ✓ 배포 후 발생할 수 있는 오류에 대한 패치나 기능에 대한 업그레이드를 위한 메뉴얼의 버전을 관리
  - ✓ 가능한 개별적으로 동작이 가능한 컴포넌트 단위로 작성
  - ✓ 컴포넌트 명세서와 컴포넌트 구현 설계서를 토대로 작성
  - ✓ 사용자 매뉴얼 작성 순서
    - 기능 식별 : 소프트웨어의 개발 목적과 사용자 활용 기능을 흐름 순으로 정리하여 기록
    - 사용자 화면 분류 : 사용자 화면을 메뉴 별로 분류하여 기록
    - 사용자 환경 파일 확인 : 폴더 위치, 사용자 로그 파 일, 백업 파일 등의 개별적인 기능을 확인하여 기록
    - 초기화 절차 확인 : 프로그램을 사용하기 위한 초기화 절차를 확인하고 그 단계를 순서대로 기록
    - 이상 Case 확인 : 소프트웨어 사용 과정에서 발생할 수 있는 다양한 이상 Case를 만들어 확인하고 해당 Case에 대한 대처법을 자세하게 기록
    - 최종 매뉴얼 적용 : 사용과 관련된 문의 답변(FAQ)을 기록한 후 완성된 매뉴얼을 검토하고 고객 지원에 대한 내용 기록

형상 관리

- 소프트웨어 패키징의 형상 관리(SCM)
  - ✓ 형상 관리(SCM; Software Configuration Management) 는 소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동이다.
  - ✓ 형상 관리 기능
    - 형상 식별: 형상 관리 대상에 이름과 관리 번호를 부여하고 계층(Tree) 구조로 구분하여 수정 및 추적이 용이하도록 하는 작업
    - 버전 제어: 소프트웨어 업그레이드나 유지 보수 과정 에서 생성된 다른 버전의 형상 항목을 관리하고, 이를 위해 특정 절차와 도구(Tool)를 결합시키는 작업
    - 형상 통제(변경 관리): 식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선(Base Line)이 잘 반영될 수 있도록 조정하는 작업
    - 형상 감사: 기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업
    - 형상 기록(상태 보고): 형상의 식별, 통제, 감사 작업 의 결과를 기록·관리하고 보고서를 작성하는 작업

형상 관리

- 소프트웨어 버전 등록 과정
  - ✓ 가져오기(Import) : 개발자가 저장소에 신규로 파일을 추가함
  - ✓ 인출(Check-Out) : 수정 작업을 진행할 개발자가 저장소에 추가된 파일을 자신의 작업 공간으로 인출함
  - ✓ 예치(Commit) : 인출한 파일을 수정한 후 설명을 붙여 저장소에 예치함
  - ✓ 동기화(Update) : 커밋(Commit) 후 새로운 개발자가 자 신의 작업 공간을 동기화(Update)함
  - ✓ 차이(Diff) : 새로운 개발자가 추가된 파일의 수정 기록 (Change Log)을 확인하면서 이전 개발자가 처음 추가 한 파일과 이후 변경된 파일의 차이를 확인함

형상 관리

- 소프트웨어 버전 관리 도구
  - ✓ 공유 폴더 방식: 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식
  - ✓ 클라이언트/서버 방식: 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식
  - ✓ 분산 저장소 방식: 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식
  - ✓ 실례
    - ✓ Subversion(SVN): CVS(Concurrent Version System)를 개선한 것으로, 클라이언트/서버 구조이며 아파치 소프트웨어 재단에서 2000년에 발표하였음
      - add: 새로운 파일이나 디렉토리 등록
      - commit: 클라이언트의 소스 파일을 서버의 소스 파일에 적용
      - update: 서버의 최근 commit 이력을 클라이언트의 소스 파일에 적용
      - checkout: 버전 관리 정보와 소스 파일을 서버에서 클라이언트로 받아옴
      - lock/unlock: 서버의 소스 파일이나 디렉토리를 잠그거나 해제
      - import: 아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장
      - export: 버전 관리에 대한 정보를 제외한 순수한 소스 파일만을 서버에서 받아옴
      - info: 파일에 대한 위치나 마지막 수정 일자 등에 대한 정보를 표시
      - diff: 파일이나 경로에 대하여 이전 리비전과의 차이를 표시
      - merge: 다른 디렉토리에서 작업된 버전 관리 내역을 기본 개발 작업과 병합

형상 관리

- ❖ 소프트웨어 버전 관리 도구
  - ✓ 실례
    - ❑ Git
      - ○ 리누스 토발즈(Linus Torvalds)가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지 보수되고 있음
      - ○ 분산 버전 관리 시스템으로 2개의 저장소(지역 저장소, 원격 저장소)를 이용하고 버전 관리는 지역 저장소에서 진행하므로 버전 관리가 빠르게 처리되고 원격 저장소나 네트워크에 문제가 발생해도 작업이 가능
      - ○ 명령어
        - ▪ add: 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역(Staging Area)에 추가
        - ▪ commit : 작업 내역을 지역 저장소에 저장
        - ▪ branch : 새로운 브랜치 생성
        - ▪ checkout : 지정한 브랜치로 이동
        - ▪ merge : 지정한 브랜치의 변경 내역을 현재 HEAD 포인터가 가리키는 브랜치에 반영함으로써 두 브랜치를 병합
        - ▪ init : 지역 저장소 생성
        - ▪ remote add : 원격 저장소에 연결
        - ▪ push : 로컬 저장소의 변경 내역을 원격 저장소에 반영 •fetch : 원격 저장소의 변경 이력만을 지역 저장소로 가져와 반영
        - ▪ clone : 원격 저장소의 전체 내용을 지역 저장소로 복제
        - ▪ fork : 지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제

빌드 자동화 도구

- 빌드
  - ✓ 빌드는 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정
  - ✓ 빌드 자동화 도구
    - ❑ Jenkins
      - Java 기반의 오픈 소스 형태로 가장 많이 사용되는 빌드 자동화 도구
      - 서블릿 컨테이너에서 실행되는 서버 기반 도구
      - SVN, Git 등 대부분의 형상 관리 도구와 연동이 가능
      - 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트가 가능
    - ❑ Gradle
      - ❑ Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구로 안드로이드 앱 개발 환경에서 사용
      - ❑ 안드로이드 뿐만 아니라 플러그 인을 설치하면 Java, C/C++, Python 등의 언어도 빌드가 가능
      - ❑ Groovy를 사용해서 만든 DSL(Domain Specific Language)을 스크립트 언어로 사용
      - ❑ Gradle은 실행할 처리 명령들을 모아 태스크(Task)로 만든 후 태스크 단위로 실행
    - ❑ Maven
      - ❑ 자바 용 프로젝트 관리 도구
      - ❑ 아파치 Ant의 대안으로 만들어 짐
      - ❑ 아파치 라이선스로 배포되는 오픈 소스 소프트웨어