Session041 소프트웨어 패키징

1 소프트웨어 패키징의 개요

1.1 모듈별 생성된 실행 파일들을 묶어 배포용 설치 파일 만드는 것

- 사용자 중심으로 진행
- 소스 코드는 향후 관리를 고려, 모듈화 하여 패키징
- 다양한 환경에서 사용할 수 있게 일반적이 배포 형태로 패키징
- 사용자의 편의성 및 실행 환경을 우선적으로

2 패키징 시 고려사항

- 운영체제, CPU, 메모리 등 최소 환경 정의
- UI는 시작 자료와 함께 제공, 매뉴얼과 일치시켜 패키징
- 하드웨어와 함께 관리될 수 있도록 Managed Service 형태로 제공
- 안정적인 배포
- 요구사항 반영할 수 있도록 패키징의 변경 및 개선에 대한 관리 항상 고려

3 패키징 작업 순서

3.1 패키징 주기는 개발 기법에 따라 다름, 애자일 경우 2~4주내에서 지정, 각 주기 끝날 때마다 패키징

- 결과물은 테스트 서버에 배포
- 최종 패키징 결과물은 온라인 또는 오프라인 배포

- 기능 식별 : 코드의 기능 확인
- 모듈화 : 기능 단위로 코드 분류
- 빌드진행 : 모듈 단위별로 실행 파일 만듬
- 사용자 환경 분석 : 플랫폼 환경, 운영환경 정의
- 패키징 및 적용 시험 : 배포용 파일 형식으로 패키징, 정의된 환경과 동일환경에서 테스팅, 불편사항 확인
- 패키징 변경 개선 : 불편사항 반영
- 배포 : 수행 시 오류 발생하면 개발자에게 수정 요청

Session042 릴리즈 노트 작성

1 릴리즈 노트(Realease Note)의 개요

1.1 개발 과정에서 정리된 릴리즈 정보를 고객과 공유하기 위한 문서

- 테스트 결과와, 사양에 대한 개발팀 준수 여부 확인 가능
- 전체 기능, 서비스의 내용, 개선 사항 등 공유

- 버전 관리나 릴리즈 정보 체계적 관리
- 초기 배포 시 또는 개선 사항 적용한 추가 배포 시 제공
- 초기 배포 시 소프트웨어 기능이나 사용 환경 확인 가능
- 철저한 테스트를 거친 것, 개발팀에서 사양에 대한 최종 승인을 얻은 후 문서화

2 릴리즈 노트 초기버전 작성 시 고려사항

- 완전한 정보 기반, 개발팀에서 현재 시제로 작성
- 신규 소스, 빌드 등의 이력 정확하게 관리, 변경 또는 개선된 항목에 대한 이력 정보 작성
- 일반적 항목
  - Header(머릿말) : 노트 이름, 소프트웨어 이름, 릴리즈 버전, 릴리즈 날짜, 릴리즈 노트 날짜, 릴리즈 노트 버전
  - 개요 : 소프트웨어 및 변경사항 전체에 대한 간략한 내용
  - 목적 : 새로운 기능, 수정된 기능들의 목록, 릴리즈 노트의 목적에 대한 개요
  - 문제 요약 : 수정된 버그 설명 또는 추가 항목 요약
  - 재현 항목 : 버그 발견 과정
  - 수정/개선 내용
  - 사용자 영향도 : 해당 릴리즈 버전에서의 기능 변화가 미칠 수 있는 영향
  - SW 지원 영향도 : 기능 변화가 다른 응용 프로그램들을 지원하는 프로세스에 미칠 수 있는 영향
  - 노트 : SW/HW 설치 항목, 업그레이드, 문서화에 대한 참고 항목
  - 면책 조항 : 회사 및 소프트웨어 관련하여 참조할 사항
  - 연락처

3 릴리즈 노트 추가 버전 작성 시 고려사항

3.1 베타 버전 출시, 긴급 버그 수정, 업그레이드, 사용자 요청 등 릴리즈 노트 추가 작성

- 오류 발생 긴급 수정 경우에 릴리즈 버전을 출시, 버그 번호를 포함 수정 내용 다아 노트 작성
- 업그레이드 완료, 릴리즈 버전 출시, 노트 작성
- 사용자 요청, 별도의 릴리즈 버전 출시, 노트 작성

4 릴리즈 노트 작성 순서

- 모듈 식별 : 작성될 내용 확인
- 릴리즈 정보 환인
- 릴리즈 노트 개요 작성
- 영향도 체크 : 버그, 이슈 관련 내용, 기능 변화가 다른 SW나 기능 사용하는데 미칠 수 있는 영향
- 정식 릴리즈 노트 작성 : 항목을 포함하여 기본 사항들 작성
- 추가 개선 항목 식별

Session043 디지털 저작권 관리(DRM)

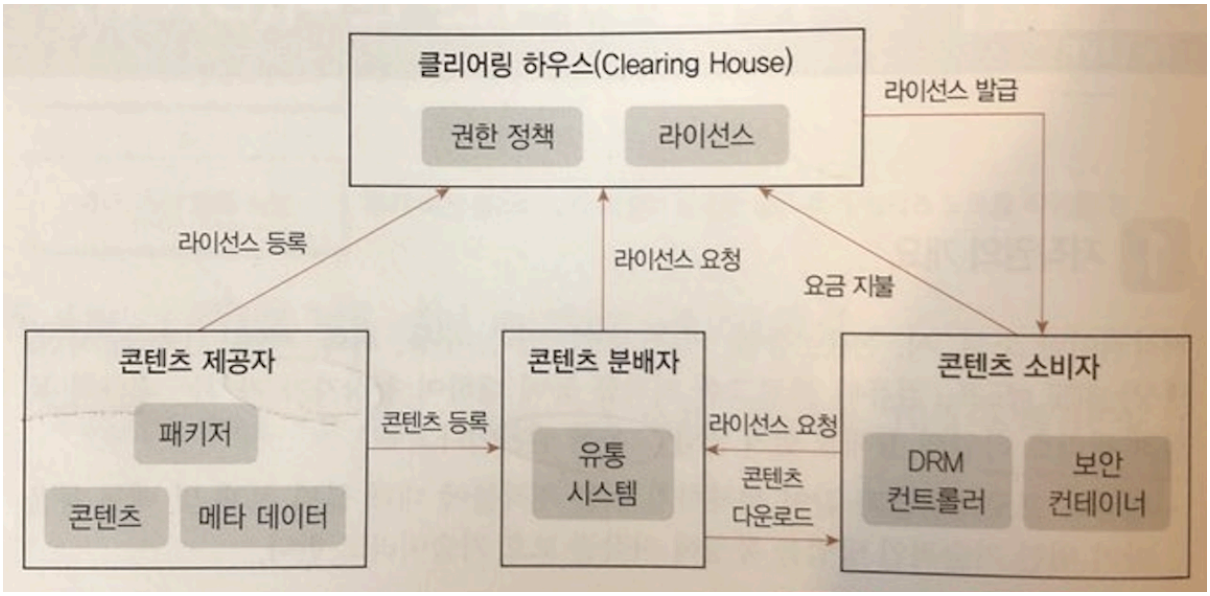
1 저작권의 개요

2 디지털 저작권 관리(DRM; Digital Right Management)의 개요

2.1 저작권자가 의도한 용도로만 사용되도록, 콘텐츠의 생성, 유통, 이용 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술

- 아날로그 -> 디지털 변환 후, DRM 패키징 수행
- 크기가 작은 경우 실시간 패키징, 큰 경우 미리 패키징 수행 후 배포
- 패키징 수행 후 콘테츠에 암호화된 저작권자의 전자서명 포함, 라이선스 정보가 클리어링 하우스에 등록
- 사용자는 사용하기 위해 클리어링 하우스에 등록된 라이선스 정보를 통해 인증과, 권한 소유 여부 확인 받는다
- 종량제 방식일 경우 실제 사용량 측정하여 요금 부과

3 디지털 저작권 관리의 흐름도



- 클리어링 하우스(Clearing House) : 저작권 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리 수행하는 곳
- 콘텐츠 제공자(Contents Provider)
- 패키저(Packager) : 메타 데이터를 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
- 콘텐츠 분배자(Contents Distributor) : 콘텐츠를 유통하는 곳이나 사람
- 콘텐츠 소비자(Customer)
- DRM 컨트롤러(DRM Controller) : 이용 권한을 통제하는 프로그램
- 보안 컨테이너(Security Container) : 원본을 안전하게 유통하기 위한 전자적 보안 장치

4 디지털 저작권 관리의 기술 요소

- 암호화(Encryption) : 암호화하고 전자 서명 할 수 있는 기술
- 키 관리(Key Management) : 암호화한 키에 대한 저장 및 분배 기술
- 암호환 파일 생성(Packager) : 암호화된 콘텐츠 생성 위한 기술
- 식별 기술(identification) : 식별 체계 표현 시루
- 저작권 표현(Right Expression) : 라이선스의 내용 표현 기술
- 정책 관리(Policy Management) : 라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
- 크랙방지(Tamper Resistance) : 크랙 사용 방지 기술
- 인증(Authentication) : 라이선스 발급 및 사용 사용자 인증 기술

Session046 소프트웨어 버전 등록

1 소프트웨어 패키징의 형상 관리(Software Configuration Management)

1.1 소프트웨어의 변경 사항 관리 위해 개발된 일련의 활동

- SW 변경 원인 알아내고 제어, 적절히 변경되고 있는지 확인, 담당자에게 통보
- 전 단계에 적용, 유지보수 포함
- 개발의 전체 비용을 줄이고, 방해 요인 최소화되도록 보증 목적

2 형상 관리의 중요성

2.1 SW의 지속적인 변경사항 체계적으로 추적, 통제

2.2 무절제한 변경 방지

2.3 버그, 수정사항 추적

2.4 SW는 가시성이 결핍, 진행 정도를 확인하기 위한 기준

3 형상 관리 기능

- 형상 식별 : 관리 대상에 이름, 관리 번호 부여, 계층 구조로 구분, 수정 및 추적이 용이하도록
- 버전 제어 : 다른 버전의 형상 항목 관리, 이를 위해 특정 절차와 도구 결합
- 형상 통제(변경 관리) : 변경 요구 검토, 기준선(Base Line)이 잘 반영 되도록 조정
- 형상 감사 : 무결성 평가 위해, 확인, 검증, 검역 과정을 통해 공식적 승인 작업
- 형상 기록(상태 보고) : 식별, 통제, 감사 작업 결과 기록, 관리하고 보고서 작업

4 소프트웨어의 버전 등록 관련 주요 용어

- 저장소(Repository)
- 가져오기(Import) : 저장소에 처음으로 파일 복사
- 체크아웃(Check-out) : 저장소에서 파일 받아옴, 버전 관리 위한 파일도
- 체크인(Check-in) : 수정 완료 후 저장소 파일 새로운 버전으로 갱신

- 커밋(Commit) : 충동 시 알리고 diff 도구를 이용 수정 후 갱신 완료
- 동기화(Update) : 저장소에 있는 최신버전으로 로컬 작업 공간 동기화

5 소프트웨어 버전 등록 과정
- Import
- Check-out
- Commit
- Update
- Diff

Session047 소프트웨어 버전 관리 도구

1 공유 폴더 방식

1.1 로컬 컴퓨터 공유 폴더에 저장되어 관리
- 약속된 공유 폴더에 매일 복사
- 파일을 자기 PC로 복사 후 컴파일 하여 이상 유무 확인
- 오류 확인 시, 해달 개발자에게 수정 의로
- 오류 없으면, 각 개발자들이 동작 여부 다시 확인
- 파일 변경 사항을 DB에 기록 관리
- SCCS, RCS, PVCS, QVCS

2 클라이언트/서버 방식

2.1 자료가 중앙 시스템에 저장되어 관리
- 서버의 자료를 자신의 PC에 복사, 작업 후 변경된 내용을 서버에 반영
- 모든 버전 관리는 서버에서 수행
- 하나의 파일을 서로 다른 개발자가 작업할 경우 경고 메시지 출력
- 서버에 문제 발생시, 복구 전까지 협업 및 버전 관리 작업 중단
- CVS, SVN(Subversion), CVSNT, Clear, Case, CMVC, Perforce

3 분산 저장소 방식

3.1 자료가 원격 저장소, 분산된 개발자 로컬 저장소에 함께 저장되어 관리
- 작업 후 변경 내용을 로컬 저장소에서 우선 반영 후 원격저장소에 반영
- 원격 저장소에 문제가 생겨도 로컬 저장소 자료 이용 작업 가능
- Git, GNU arch, DCVS, Bazzar, Mercurial, TeamWare, Bitkeeper

4 Subversion(서브버전, SVN)

4.1 CVS 개선, 아파치에서 발표
- 클라이언트/서버 구조

- 모든 개발 작업은 trunck 디렉터리에서 수행, 추가 작업은 branches 디렉토리 안에서 작업, 완료 후 trunk 디렉토리와 병합
- 커밋할 때마다 리버전이 1씩 증가
- 서버는 주로 Unix
- 오픈 소스, 무료
- 파일 or 디렉토리 이름 변경, 이동 가능
- 명령어
  - Add : 관리 대상으로 등록, add 대상이 아니면 적용X
  - Commit : 서버의 소스 파일에 적용
  - Update : 최신 commit 이력 로컬 소스 파일에 적용, commit 전에 매번 update 수행, 서버의 변동내역을 클라이언트에 적용
  - Checkout : 서버에서 소스파일 받아 옴, 버전 관리 정보도
  - Lock/unlock : 파일이나 디렉터리를 잠그거나 해제
  - Import : 맨 처음 서버 저장소에 소스 파일을 저장하는 명령
  - Export : 버전 관리 제외 소스 파일만 서버에서 받아옴
  - Info : 위치, 마지막 수정 일자 등 정보 표현
  - Diff : 이전 리비전과의 차이 표시
  - Merge : 버전 관리 내역을 기본 개발 작업과 병행

5 Git(깃)

5.1 리누스 토발즈 개발, 주니오 하마노 유지보수

- 분산 버전 관리 시스템
- 로컬 저장소 실제 개발 진행하는 장소, 버전 관리 수행
- 원격 저장소 협업을 위해 버전 공동 관리하는 곳, 자신의 관리 내역을 반영 또는 다른 개발자의 변경 내용을 가져 올때
- 로컬에서 버전 관리가 되므로 신속하고, 원격 또는 네트워크 문제가 있어도 작업 가능
- 브랜치 이용하면 다양한 형태의 기능 테스팅 가능
- 파일 변화를 스냅샷으로 저장, 이전 스탭샷의 포인터를 가진다
- 명령어
  - Add : 작업 내역을 로컬 저장소에 저장하기 위해 Staging Area에 추가
  - Commit : 작업 내역 로컬 저장소에 저장
  - Branch : 새로운 브랜치 생성
  - Checkout : 지정한 브랜치로 이동, HEAD 포인터
  - Merge : 두 브랜치를 병합
  - Init : 로컬 저장소 생성
  - Remote add : 원격 저장소에 연결
  - Push : 로컬 변경 내역을 원격에 반영
  - Fetch : 원격의 변경 이력만을 가져와 반영

- Clone : 원격 전체 내용을 지역으로 복제
- Fork : 지정 원격 내용을 자신의 원격으로 복제

Session048 빌드 자동화 도구

1 빌드 자동화 도구의 개념

1.1 빌드 : 소스 코드 파일을 컴파일 후, 여러 개의 모듈을 묶어 실행 파일로 만드는 과정

1.2 빌드 포함 테스트 및 배포를 자동화하는 도구

- 애자일 같은 지속적 통합(Continuous Integration) 개발 환경에서 유용
- Ant, Make, Maven, Gradle, Jenkins

2 Jenkins

2.1 자바 기반의 오픈 소스 형태, 가장 많이 사용

- 서블릿 컨테이너에서 실행되는 서버 기반 도구
- 대부분의 형상 관리 도구와 연동 가능
- Web GUI
- 여러 대의 컴퓨터를 이용한 분산 빌드 테스트 가능

3 Gradle

3.1 Groovy를 기반으로 한 오픈 소스 형태, 안드로이드 앱 개발 환경에서 사용

- 플러그인을 설정하면, JAVA, C/C++, Python
- DSL(Domain Specific Language)을 스크립트 언어로 사용
- 실행할 처리 명령들을 모아 Task로 만든 후 태스크 단위 실행
- 태스크 재사용, 다른 시스템 태스크 공유 할 수 있는 빌드 캐시 기능 지원