제품 소프트웨어 패키징

1. 제품 소프트웨어 패키징

1) 애플리케이션 패키징(Application Packaging)

* 개념
  - 개발이 완료된 제품 소프트웨어를 배포하고 설치할 수 있도록 고객에게 전달하기 위한 형태로 제작
  - 설치와 사용에 필요한 제반 내용을 포함하는 매뉴얼을 작성하는 활동

* 특징
  - 사용자 중심으로 진행 :
    >사용자의 실행 환경을 이해하고 범용 환경에서 사용이 가능하도록 일반적인 배포 형태로 분류하여 패키징이 진행
  - 신규 및 변경 개발 소스를 식별>이를 모듈화하여 상용 제품으로 패키징
  - 신규/변경 이력을 확인 ← 버전 관리 및 릴리즈 노트를 통해 지속적 관리

* 고려사항
  • 사용자 실행 환경의 이해
  - 고객 편의성을 위해 사용자 실행 환경을 우선 고려하여 패키징 진행
  - OS(Operating System)부터 시작하여 실행 환경, 시스템 사양 및 고객의 사용 방법까지 상세 분류하여 실행환경을 사전 정의
  - 만약 여러 가지 실행 환경이 나오게 된다면 해당 경우에 맞는 배포본을 분류하여 패키징 작업을 여러 번 수행

  • 사용자 관점에서의 패키징 고려사항 :

사용자 시스템 환경정의	사용자 시스템의 환경인 운영체제, CPU, 메모리 등의 수행을 위한 최소 환경을 정의
UI 제공	- 사용자가 직관적으로 확인할 수 있는 UI 제공 - 매뉴얼과 일치시켜 패키징 작업수행
관리 서비스 형태로 제공	애플리케이션은 하드웨어와 함께 통합 적용할 수 있도록, 패키징을 관리 서비스 형태로 제공
패키징의 변경 및 개선 관리 고려	다양한 사용자의 요구사항을 반영하기 위해 패키징의 변경 및 개선 관리를 고려하여 패키징 배포

* 애플리케이션 패키징 프로세스

기능 식별	- 입출력 데이터 식별 - 전체적 기능 정의 및 데이터 흐름 식별 - 기능 단위 및 출력에 대한 상세 정의
모듈화	- 기능 및 서비스를 모듈 단위로 분류 - 기능의 공유와 재활용 분류 - 모듈 간 결합도와 응집도 식별
빌드 진행	- 신규 개발 소스 및 컴파일 결과물 준비 - 정상 기능 단위 및 서비스 분류 - 빌드 도구 확인 및 정상 수행 - 컴파일 이외 도구의 다양한 기능 확인
사용자 환경 분석	- 최소 사용자 환경 사전 정의 - 모듈 단위의 사용자 환경 테스트 수행
패키지 적용 시험	- 사용자 환경에서의 패키징 적용 시험 - UI 및 시스템 상의 편의성 체크
패키지 변경 개선	- 패키징 적용 시 변경점 도출 - 최소 사용자 환경에서 서비스 가능한 수준의 개선 - 개선 버전의 재배포

※ 순서 : 기능 식별>모듈화>빌드 진행>사용자 환경 분석>패키징 적용 시험>패키징 변경 개선

* 애플리케이션 패키징 릴리즈 노트

• 개념

- 애플리케이션 최종 사용자인 고객과 잘 정리된 배포 정보를 공유하는 문서
- 상세 서비스를 포함하여 수정/변경 또는 개선되는 정보에 대한 사항 제공

• 작성 항목

헤더	문서 이름, 제품 이름, 버전 번호, 릴리즈 날짜, 참고 날짜, 노트 버전 등
개요	제품 및 변경에 대한 간략한 전반적 개요
목적	- 릴리즈 버전의 새로운 기능 목록 - 릴리즈 노트의 목적에 대한 개요 - 버그 수정 및 새로운 기능 기술
이슈 요약	버그의 간단한 설명 또는 릴리즈 추가 항목 요약
재현 항목	버그 발견에 따른 재현 단계 기술
수정/개선 내용	수정/개선의 간단한 설명 기술
사용자 영향도	버전 변경에 따른 최종 사용자 기준의 기능 및 응용 프로그램상의 영향도 기술
소프트웨어 지원 영향도	버전 변경에 따른 소프트웨어의 지원 프로세스 및 영향도 기술
노트	소프트웨어 및 하드웨어 설치 항목, 제품, 문서를 포함한 업그레이드 항목 메모
면책 조항	- 회사 및 표준 제품과 관련된 메시지 - 프리웨어 및 불법 복제 방지 - 중복 등 참조에 대한 고지사항
연락 정보	사용자 지원 및 문의 관련한 연락처 정보

• 작성 프로세스 :

모듈 식별	- 릴리즈 노트 작성을 위한 모듈 및 빌드 정리

	- 입출력 데이터, 전체적 기능 정의, 데이터 흐름 정리 - 기능 단위 및 출력에 대한 상세 정의
릴리즈 정보 확인	- 문서 이름(릴리즈 노트 이름), 제품 이름 정보 확인 - 버전 번호, 릴리즈 날짜 확인 - 참고 날짜, 노트 버전 확인
릴리즈 노트 개요 작성	- 제품 및 변경에 대한 간략한 전반적 개요 작성 - 개발 소스의 빌드에 따른 결과물 기록 - 버전 및 형상 관리에 대한 전반적인 노트 기록
영향도 체크	- 버그의 간단한 설명 또는 릴리즈 추가 항목 기술 - 버그 발견을 위한 재현 테스트 및 재현 환경을 기록 - 소프트웨어 및 사용자 입장에서의 영향도 파악
정식 릴리즈 노트 작성	- 릴리즈 정보, 헤더 및 개요 등 기본사항 기술 - 정식 버전을 기준으로 릴리즈 노트 개요 작성 - 이슈, 버그 등 개선내용 기술
추가 개선 항목 식별	- 추가 개선에 대한 베타 버전을 이용 테스트 수행 - 테스트 중 발생한 긴급 버그 수정 - 추가 기능 향상을 위해 작은 기능 수정 - 사용자 요청에 따른 추가개선

※ 순서 : 모듈 식별>릴리즈 정보 확인>릴리즈 노트 개요 작성>영향도 체크>정식 릴리즈 노트 작성>추가 개선 항목 식별

2) 애플리케이션 배포 도구

* 개념

- 배포를 위한 패키징 시에 디지털 콘텐츠의 지적 재산권을 보호하고 관리하는 기능 제공
- 안전한 유통과 배포를 보장하는 도구/솔루션

* 기술 요소(DRM 기술요소와 동일)

암호화	콘텐츠 및 라이선스를 암호화하고, 전자서명을 할 수 있는 기술 > 공개 키 기반 구조(PKI), 대칭 및 비대칭 암호화, 전자서명
키 관리	콘텐츠를 암호화한 키에 대한 저장 및 배포기술(중앙 집중형, 분산형)
식별 기술	콘텐츠에 대한 식별 체계 표현 기술 > DOI, URI
저작권 표현	라이선스의 내용 표현 기술 > XrML/MPEG-21
암호화 파일 생성	콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
정책 관리	라이선스 발급 및 사용에 대한 정책표현 및 관리 기술 > XML, 콘텐츠 관리 시스템(CMS)
크랙 방지	크랙에 의한 콘텐츠 사용 방지 기술 > 난독화, Secure DB ※ 크랙 : 소프트웨어를 수정하여 소프트웨어 복사 방지나 소프트웨어 조작 보호 등을 비활성화하거나 제거

인증	라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술 > 사용자/장비 인증, SSO

* 애플리케이션 배포 도구 세부 기술

공개키 기반 구조 (PKI)	공개키 암호 방식 기반으로 디지털 인증서를 활용하는 소프트웨어, 하드웨어, 사용자, 정책 및 제도 등을 총칭하는 암호기술
대칭 및 비대칭 암호화	- 대칭 암호화는 암호화와 해독을 위해 동일한 키를 사용하는 암호화 방식 - 비대칭 암호화는 데이터를 암호화할 때와 이를 해독할 때 서로 다른 키를 사용하는 방식
전자서명	서명자를 확인하고 서명자가 해당 전자문서에 서명했다는 사실을 나타내기 위해 특정 전자문서에 첨부되거나 논리적으로 결합된 전자적 형태의 정보
DOI (Digital Object Identifier)	- 디지털 저작물에 특정한 번호를 부여하는 일종의 바코드 시스템 - 디지털 저작물의 저작권 보호 및 정확한 위치 추적이 가능한 시스템
URI (Uniform Resource Identifier)	인터넷에 있는 자원을 나타내는 유일한 주소
XrML (eXtensible Right Markup Language)	디지털 콘텐츠/ 웹 서비스 권리 조건을 표현한 XML 기반의 마크업 언어
MPEG-21	멀티미디어 관련 요소 기술들이 통일된 형태로 상호 운용성을 보정하는 멀티미디어 표준 규격
XML (eXtensible Markup Language)	XML은 W3C에서 개발된 다른 특수한 목적을 갖는 마크업 언어를 만드는데 사용하도록 권장하는 다목적 마크업 언어 ※ 마크업 언어 : 문서의 구조를 표현하는 역할을 하는 태그 방법의 체계
CMS (Contents Management System)	콘텐츠 관리시스템은 다양한 미디어 포맷에 따라 각종 콘텐츠를 작성, 수집, 관리 ,배급하는 콘텐츠 생산에서 활용, 폐기까지 전 공급 과정을 관리하는 기술
코드 난독화	역공학을 통한 공격을 막기 위해 프로그램의 소스 코드를 알아보기 힘든 형태로 바꾸는 기술 ※ 역공학 : 기존 개발된 시스템의 기술적 원리를 시스템의 코드나 데이터 등의 구조 분석을 통해 도출해 내는 작업
Secure DB	커널 암호화 방식으로 데이터베이스 파일을 직접 암호화하고, 접근 제어와 감사 기록 기능이 추가된 데이터베이스 보안 강화 기술
SSO (Single Sign On)	한 번의 시스템 인증을 통하여 여러 정보시스템에 재인증 절차 없이 접근할 수 있는 통합 로그인 기술

* 애플리케이션 배포 도구를 활용한 배포 프로세스

빌드 내용 식별	- 릴리즈 노트 작성을 위한 모듈 및 빌드 정리 - 입출력 데이터, 전체적 기능 정의, 데이터 흐름 정리 - 기능 단위 및 출력에 대한 상세 정의
패키징 도구 식별	- 패키징 도구의 사전 선택 - 암호/보안 기능 확인
DRM 흐름을 확인하여	- 콘텐츠 분배자, 배포자, 소비자 간 DRM 흐름 확인

패키징 수행	- 패키징 수행 시 키 관리, 보안 개념 확인하며 수행
패키징 도구 설치	- 환경에 맞게 패키징 도구 설치 작업 진행 - 패키징 도구 설치 완료 후 정상 작동 확인
배포 작업	- 패키징 도구 설치 이후 제품 소프트웨어의 배포 작업을 진행 - 배포 후 최종 패키징 완료 확인
정상 배포 확인	- 암호화, 보안 기능 적용 확인 - 제품 소프트웨어 배포본 기준으로 암호화/보안 기능 체크리스트 확인

※ 빌드 내용 식별>패키징 도구 식별>DRM 흐름을 확인하여 패키지 수행>패키징 도구 설치>배포 작업>정상 배포 확인

* 애플리케이션 배포 도구를 활용 시, 고려사항

암호화/보안	패키징 시 사용자에게 배포되는 소프트웨어임을 감안하여 반드시 내부 콘텐츠에 대한 암호화 및 보안 고려
이기종 연동	패키징 도구를 활용하여 여러 가지 이기종 콘텐츠 및 단말기 간 DRM 연동 고려
복잡성 및 비효율성 문제	사용자의 입장에서 불편해질 수 있는 문제를 고려하여, 최대한 효율적으로 적용될 수 있도록 함
최적합 암호화 알고리즘 적용	암호화 알고리즘이 여러 가지 종류가 있는데, 제품 소프트웨어의 종류에 맞는 알고리즘을 선택하여 배포 시 범용성에 지장이 없도록 고려

3) 애플리케이션 모니터링 도구

* 개념 :

제품 소프트웨어를 사용자 환경에 설치한 후 기능 및 성능, 운영 현황을 모니터링하여 제품을 최적화하기 위한 도구

* 기능

기능	설명	도구
애플리케이션 변경 관리	- 애플리케이션 간의 종속관계를 모니터링 - 애플리케이션의 변경이 있을 경우, 변경의 영향도 파악에 활용	ChangeMiner
애플리케이션 성능 관리	- 애플리케이션 서버로 유입되는 트랜잭션 수량, 처리시간, 응답시간을 모니터링	Jeniffer, Nmon
애플리케이션 정적 분석	- 소스 코드의 잠재적 문제 발견 기능 - 코딩 규칙 오류 발견	PMD, Cppcheck, Checkstyle, SonarQube
애플리케이션 동적 관리	- 프로그램에 대한 결함 및 취약점 동적 분석 도구 - 메모리 및 오류 문제 발견	Avalanche, Valgrind

* 효과

서비스 가용성	- 모니터링 자동화 및 관련 데이터 생성 - 시스템에 의한 서비스 모니터링
서비스 성능	- 24시간 대상 애플리케이션 측정 - 다양한 서비스 대상 측정 - 시스템에 의한 객관적 측정
장애 인지/리소스 측정	- 서비스 24시간 모니터링 - 가용성 데이터 및 관련 자료 생성

	- 성능 저하 발생 시 자동 열람 가능
근본 원인 분석	- 사용자 입장에서 원인 분석 - 사용자 영역(End to End) 분석으로 누구나 공감 - 문제 분석을 위한 시간의 획기적 단축 - 문제 분석 후 객관적인 원인 분석 자료 제공

4) DRM(Digital Rights Management)

* 개념 :

- 디지털 콘텐츠에 대한 권리정보를 지정
- 암호화 기술을 이용해 허가된 사용자의 허가된 권한 범위 내에서 콘텐츠의 이용이 가능하도록 통제

* 특징 :

거래 투명성	저작권자와 콘텐츠 유통업자 사이의 거래구조 투명성 제공
사용규칙 제공	- 사용가능 횟수, 유효기간, 사용 환경 등을 정의 가능 - 다양한 비즈니스 모델 구성 및 콘텐츠 소비 형태 통제 제공
자유로운 상거래 제공	- 이메일, 디지털 미디어, 네트워크 등을 통한 자유로운 상거래 제공 - 허가 받은 사용자는 별도의 비밀키를 이용해 대상 콘텐츠를 복호화 하고 허가된 권한으로 사용가능

* 구성 및 동작 방식

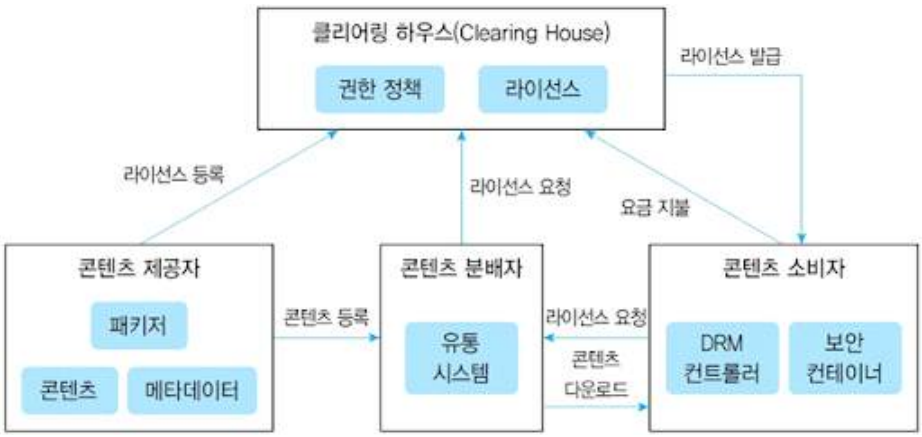

그림. DRM 구성

• DRM 동작 방식

번호	단계	동작방식
1	라이선스 등록	클리어링 하우스에 라이선스 등록을 하면서 동시에 유통시스템에 콘텐츠를 등록
2	라이선스 요청	콘텐츠 소비자가 유통시스템으로 라이선스 요청
3	라이선스 요청	유통시스템에서 클리어링 하우스를 통해 라이선스 요청

4	요금 지불	콘텐츠 소비자가 요금 지불
5	라이선스 발급	클리어링 하우스를 통해 라이선스 발급
6	콘텐츠 다운로드	그 이후에 콘텐츠 소비자가 콘텐츠를 다운로드 받을 수 있음

• DRM 구성 요소(저작권 관리 구성요소)

<table>
<tr><td>콘텐츠 제공자
(Contents Provider)</td><td>콘텐츠를 제공하는 저작권자</td></tr>
<tr><td>콘텐츠 소비자
(Contents Customer)</td><td>콘텐츠를 구매해서 사용하는 주체</td></tr>
<tr><td>콘텐츠 분배자
(Contents Distributor)</td><td>암호화된 콘텐츠를 유통하는 곳이나 사람</td></tr>
<tr><td>클리어링 하우스
(Clearing House)</td><td>- 저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 관리 등을 수행
- 키 관리 및 라이선스 발급 관리
- 콘텐츠 권한 정책, 라이선스 관리 수행
<table><tr><td>콘텐츠
권한 정책</td><td>- 라이선스 발급 여부를 결정하는 정책에 대한 부합 여부 확인
- 적절한 사용 권한을 부여하는 역할 수행</td></tr><tr><td>콘텐츠
라이선스</td><td>- 클리어링 하우스에 의해서 사용자에게 전달되는 콘텐츠의 권리 인증
- 콘텐츠에 대한 사용 조건 및 허가 정보를 포함</td></tr></table></td></tr>
<tr><td>DRM 콘텐츠
(DRM Contents)</td><td>서비스하고자 하는 암호화된 콘텐츠, 콘텐츠와 관련된 메타 데이터, 콘텐츠 사용정보를 패키징하여 구성된 콘텐츠

※ 메타 데이터 : 자료의 속성등을 설명하는 데이터</td></tr>
<tr><td>패키저
(Packager)</td><td>콘텐츠를 메타 데이터와 함께 배포 가능한 단위를 묶는 도구</td></tr>
<tr><td>DRM 컨트롤러
(DRM Controller)</td><td>배포된 디지털 콘텐츠의 이용 권한 통제</td></tr>
<tr><td>보안 컨테이너
(Security Container)</td><td>원본 콘텐츠를 안전하게 유통하기 위한 전자적 보안 장치</td></tr>
</table>

* DRM 기술요소(애플리케이션 배포 도구의 기술요소와 동일)

암호화	콘텐츠 및 라이선스를 암호화하고, 전자서명을 할 수 있는 기술 > 공개 키 기반 구조(PKI), 대칭 및 비대칭 암호화, 전자서명
키 관리	콘텐츠를 암호화한 키에 대한 저장 및 배포기술(중앙 집중형, 분산형)
식별 기술	콘텐츠에 대한 식별 체계 표현 기술 > DOI, URI
저작권 표현	라이선스의 내용 표현 기술 > XrML/MPEG-21
암호화 파일 생성	콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
정책 관리	라이선스 발급 및 사용에 대한 정책표현 및 관리 기술

	> XML, 콘텐츠 관리 시스템(CMS)
크랙 방지	크랙에 의한 콘텐츠 사용 방지 기술 > 난독화, Secure DB ※ 크랙 : 소프트웨어를 수정하여 소프트웨어 복사 방지나 소프트웨어 조작 보호 등을 비활성화하거나 제거
인증	라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술 > 사용자/장비 인증, SSO

2. 제품 소프트웨어 매뉴얼 작성

1) 제품 소프트웨어 매뉴얼 작성

* 개념
  - 제품 소프트웨어 개발 단계부터 적용한 기준이나 패키징 이후 설치 및 사용자 측면의 주요 내용 등을 문서로 기록한 것
  - 사용자 중심의 기능 및 방법을 나타낸 설명서와 안내서
  - 설치 매뉴얼과 사용자 매뉴얼이 있음

* 제품 소프트웨어 설치 매뉴얼
  • 개념
  - 사용자가 제품을 구매한 후 최초 설치 시 참조하는 매뉴얼
  - 설치 과정에서 표시될 수 있는 예외상황에 대한 관련 내용을 별도로 구분하여 설명
  - 설치 시작부터 완료할 때까지의 전 과정을 빠짐없이 순서대로 설명

  • 작성 항목

목차 및 개요	- 매뉴얼 전체 내용을 순서대로 요약 - 설치 매뉴얼의 주요 특징, 구성과 설치 방법, 순서 등에 대해 기술
문서 이력 정보	매뉴얼 변경 이력에 대한 정보를 버전별로 표시
설치 매뉴얼 주석	- 주의 사항 : 사용자가 제품 설치 시 반드시 숙지해야 하는 중요한 정보 주석 표시 - 참고 사항 : 설치 관련하여 영향을 미치는 특별한 사용자 환경 및 상황에 대한 내용 주석 표시
설치 도구의 구성	- exe/dll/ini/chm 등 해당 설치 관련 파일 설명 - 폴더 및 설치 프로그램 실행 파일 설명
설치 위치 지정	설치 폴더와 설치 프로그램 실행 파일 설명

  • 체크 항목

사용자 환경	CPU 및 메모리, 운영체제(OS) 등의 적합 환경 확인
응용 프로그램	설치 전 다른 응용 프로그램의 종료 확인
업그레이드 버전	업그레이드 이전 버전에 대한 존재 유무 확인

백업 폴더 확인	데이터 저장 폴더를 확인하여 설치 시 폴더 동기화

• 구성요소

제품 소프트웨어 개요	- 제품 소프트웨어의 주요 기능 및 UI 설명 - UI 및 화면상의 버튼, 프레임 등을 도식화하여 설명
설치 관련 파일	- 제품 소프트웨어를 설치하기 위한 관련 파일 설명 - 설치 구동을 위한 exe 실행 파일 - ini나 log파일 같은 관련 파일
설치 절차	- 제품 소프트웨어 설치를 위한 상세 절차 설명 - 설치 디렉토리, 위치 설명
설치 아이콘	- 윈도즈 구동용 설치 아이콘 설명
삭제 방법	- 제품 소프트웨어 삭제 시 원하는대로 삭제하는 방법 설명
설치 버전 및 작성자	- 제품 소프트웨어 릴리즈 버전 및 작성자 정보
고객 지원 방법 및 FAQ	- 설치 관련하여 기술적 지원이나 제품 서비스를 받을 수 있는 유선 및 이메일, 홈페이지 주소 - 설치 시, 자주 발생하는 오류 및 처리 방법에 대한 요약 설명
준수 정보 & 제한 보증	- 시리얼 보존, 불법 등록 사용 금지 등의 준수 사항 권고 - 저작권 정보 관련 사항 작성

• 작성 프로세스

순서	프로세스	설명
1	개요 및 기능 식별	제품 소프트웨어 개발 목적 및 제품의 전체적 기능 식별
2	UI 분류	- 설치를 위한 화면 단위 및 메뉴 분류 - UI 정의서에 기초한 메인 항목 분류 - 설치 매뉴얼에 작성될 순서대로 UI 분류
3	설치 파일 / 백업 파일 확인	- 실제로 제품을 설치할 파일 및 백업 파일명 확인 - 제품을 설치할 파일 및 백업 파일명의 폴더 위치 확인 - 실행, 환경, 로그, 백업 등의 다양한 파일들을 확인하고 기능을 숙지
4	삭제 절차 확인	- 제품의 제거를 고려하여 삭제 파일 절차 확인 - 삭제 이후 설치 전 상태로의 원본을 최종 확인
5	이상 유형 확인	- 설치에 대한 이상 유형에 대한 테스트 수행 - 다양한 이상 현상 발생 시 이에 따른 메시지 정리 - 유형별로 발생하는 메시지가 정상적인지 확인
6	최종 매뉴얼 적용	최종 설치 정상 완료 시 결과를 캡쳐 후 최종 매뉴얼에 적용

※순서 : 개요 및 기능 식별>UI 분류>설치 파일/백업 파일 확인>삭제 절차 확인>
이상 유형 확인> 최종 매뉴얼 적용

* 제품 소프트웨어 사용자 매뉴얼

• 개념

- 개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 패키징하고 설치와 사용에 필요한 제반 절차 및 환경 등 전체 내용을 포함하는 문서
- 개발된 컴포넌트 사용 시에 알아야 할 내용을 기술하며 패키지의 기능, 인터페이스, 포함하고 있는 메소드나 오퍼레이션, 메소드의 파라미터 등의 설명이 포함

• 작성 항목

목차 및 개요	- 매뉴얼 전체 내용을 순서대로 요약 - 제품 소프트웨어의 주요 특징 정리 - 사용자 매뉴얼에서의 구성과 실행방법, 메뉴에 대한 설명을 비롯하여 사용법, 각 항목에 따른 점검 기준, 설정 방법 등에 대해 기술
문서 이력 정보	버전, 작성자, 작성일, 검토자, 일시, 검수인 등을 일자별로 기록
사용자 매뉴얼 주석	- 주의 사항 : 사용자가 반드시 숙지해야 하는 중요한 정보의 주석 표시 - 참고 사항 : 특별한 사용자 환경 및 상황에 대한 내용의 주석 표시
기록항목	제품 명칭, 모델명, 문서 번호, 제품 번호 등의 항목
기본사항	개요, 사용방법 및 관리방법, 모델, 버전별 특징, 제품 소프트웨어 기능 및 인터페이스 특징, 구동환경 등
고객 지원 방법 및 FAQ	- 설치 관련하여 기술적인 지원이나 제품 서비스를 받을 수 있는 유선 및 이메일, 홈페이지 주소 - 설치 시, 자주 발생하는 오류 및 처리방법에 대한 요약 설명
준수 정보&제한 보증	- 시리얼 보존, 불법 사용 금지 등의 준수 사항 권고 - 저작권 정보 관련 사항 작성

• 작성 프로세스

순서	프로세스	설명
1	작성 지침 정의	- 사용자 매뉴얼을 작성하기 위한 지침 설정 - 사용자 매뉴얼은 실제 사용자 환경에 필요한 정보를 제공할 수 있는 형태로 작성
2	사용자 매뉴얼 구성요소 정의	제품 소프트웨어의 기능, 구성 객체 목록, 객체별 메소드, 메소드의 파라미터, 실제 사용 예제, 사용자 환경 세팅 방법 등의 사용자 매뉴얼 구성요소를 정의
3	구성요소별 내용 작성	제품 소프트웨어 구성요소별로 내용 작성
4	사용자 매뉴얼 검토	- 작성된 사용자 매뉴얼이 개발된 제품의 기능을 제대로 설명하는지, 제품 사용 시 부족한 정보가 없는지 등을 검사 - 해당 기능별 관련 개발자와 함께 기능 내용이나 인터페이스, 메소드나 메소드의 파라미터 등을 검토 - 점검 사항을 반영하여 사용자 지침서 수정, 보완

※ 작성 지침 정의>사용자 매뉴얼 구성 요소 정의>구성요소별 내용 작성>사용자 매뉴얼 검토

2) 국제 표준 제품 품질 특성

* 개념

- 명확하게 정의된 특성 : 품질을 평가하는 기준 항목
- 품질에 관련된 국제 표준화는 ISO/IEC, ITU-T, IEEE를 중심으로 진행
- 제품 품질 표준과 프로세스 품질 표준으로 세분화

※ 정의

- ISO : 여러 나라의 표준 제정 단체들의 대표들로 이루어진 국제 표준화 기구
- IEC : 국제전자기술위원회 ☞ 전기, 전자 및 관련 기술을 위한 국제 표준
- ITU-T : 국제전기통신연합 전기통신표준화부문의 하나 ☞ 통신 분야의 표준 책정
- IEEE : 전기 전자 기술자 협회 ☞ 전기전자공학 전문가들의 국제조직

* 국제 제품 품질 표준

• 국제 제품 품질 표준

- IT 프로젝트를 진행하거나 완성된 IT 제품에 대해 기능성, 신뢰성 등을 평가하는 기준

ISO/IEC 9126	- ISO/IEC 9126의 품질 모델은 소프트웨어 품질을 측정, 평가하기 위해서 소프트웨어의 품질요소와 특성을 정의 - 품질 특성은 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성으로 분류
ISO/IEC 14598	- 소프트웨어 제품 평가 프로세스 및 평가 모듈 제공 - 패키지 소프트웨어와 SI개발 소프트웨어에 있어서 개발과정 또는 개발이 완료된 제품의 품질에 대한 평가 표준과 프로세스 제공
ISO/IEC 12119	- 소프트웨어 패키지 제품에 대한 품질 요구사항 및 테스트 국제 표준 - 대상 : 제품 설명서, 사용자 문서, 실행 프로그램
ISO/IEC 25000	- SQuaRE로도 불리며, ISO/IEC 9126과 ISO/IEC 14598, ISO/IEC 12119를 통합하고, ISO/IEC 15288을 참고한 소프트웨어 제품 품질에 대한 통합적 국제표준 - 개발 공정 각 단계에서 산출되는 제품이 요구사항을 만족하는지 검증하기 위해 품질 측정 및 평가를 위한 모델

• ISO/IEC 9126의 소프트웨어 품질 특성

품질 특성	설명	부록성
기능성 (Functionality)	소프트웨어가 특정 조건에서 사용될 때 명시된 요구와 내재된 요구를 만족하는 기능을 제공하는 소프트웨어 제품의 능력	적합성, 정확성, 상호운용성, 보안성, 준수성 등
신뢰성 (Reliability)	- 명시된 조건에서 사용될 때, 성능 수준을 유지할 수 있는 소프트웨어 제품의 능력 - 옳고 일관된 결과를 얻기 위해 요구된 기능을 수행할 수 있는 정도 - 주어진 시간 동안 주어진 기능을 오류 없이 수행하는 정도	성숙성, 결함허용성, 회복성, 준수성 등
사용성	명시된 조건에서 사용될 경우, 사용자에 의해 이해되고 학습되고 사	이해성, 학습

(Usability)	용되고 선호될 수 있는 소프트웨어 제품의 능력	성, 운용성, 친밀성, 준수성 등
효율성 (Efficiency)	명시된 조건에서 사용되는 자원의 양에 따라 요구된 성능을 제공하는 소프트웨어 제품의 능력	시간 반응성, 자원 효율성, 준수성 등
유지보수성 (Maintainability)	- 소프트웨어 제품이 변경되는 능력 - 변경에는 환경, 요구사항 및 기능적 명세에 따른 소프트웨어의 수정, 개선, 혹은 개작 등이 포함	분석성, 변경성, 안정성, 시험성, 준수성 등
이식성 (Portability)	하나 이상의 하드웨어 환경에서 운용되기 위해 쉽게 수정될 수 있는 시스템 능력	적응성, 설치성, 공존성, 대체성, 준수성 등

• ISO/IEC 14598의 소프트웨어 품질 특성 :
개발자에 대한 소프트웨어 제품 품질 향상과 구매자의 제품 품질 선정 기준을 제공하는 표준

반복성 (Repeatability)	특정 제품을 동일 평가자가 동일 사양으로 평가하면 동일한 결과가 나와야한다는 특성
재현성 (Reproductibility)	특정 제품을 다른 평가자가 동일 사양을 평가하면 유사한 결과가 나와야한다는 특성
공정성 (Impartiality)	평가가 특정 결과에 편향되지 않아야 한다는 특성
객관성 (Objectivity)	평가 결과는 객관적 자료에 의해서만 평가되어야 한다는 특성

* 국제 프로세스 품질 표준(숫자 위주로 숙지)

품질 표준	설명	세부사항
ISO/IEC 9001	- 설계/개발, 생산, 설치 및 서비스 과정에 대한 품질 보증 모델 - 필요한 품질 시스템 순기활동과 그에 따른 공급자와 구매자 각각의 관리책임을 명시 - 운영 중인 품질 시스템이 표준에 적합할 경우 품질 인증 부여	ISO 9000-3 (ISO 9001표준을 소프트웨어 산업에 맞게 변형한 국제표준)
ISO/IEC 12207	소프트웨어의 획득, 공급, 개발, 운영, 유지보수를 체계적으로 관리하기 위한 소프트웨어 생명주기 단계별 필요 프로세스를 규정한 국제표준	기본/지원/조직 프로세스
ISO/IEC 15504 (SPICE)	- 소프트웨어 프로세스를 평가하고 개선함으로써 품질 및 생산성을 높이고자 하는 국제표준 - 소프트웨어 프로세스 영역을 ISO/IEC 12207에 준거하여 기본/지원/조직 프로세스로 구분하고 있으며 각 프로세스 영역별로 프로세스 카테고리와 기본 프로세스 정의	수행단계 구분 0. 불완전 1. 수행 2. 관리 3. 확립 4. 예측

		5. 최적화
CMMi	- 기존 CMM 모델을 통합하고 ISO15504를 준수하는 소프트웨어 개발 능력/성숙도 평가 및 프로세스 개선 활동의 지속적 품질 개선 모델 - 적용 및 평가 방식은 조직차원의 성숙도를 평가하는 단계별 표현과 프로세스 영역별 능력도를 평가하는 연속적 표현으로 세분화	- 프로세스 영역 - 목표 - 실행 - 공통특징

※ CMMi 단계별 표현 : 조직의 전체적인 성숙도 확인을 위해 5단계의 성숙도 레벨로 정의하고 성숙도 수준을 검증하고 평가

※ CMMi 연속적 표현 : 능력 수준을 이용해 프로세스 영역을 4개의 범주로 그룹화하여 프로세스 영역별로 조직의 성숙도를 평가

* 소프트웨어 품질 평가 통합 모델(ISO/IEC 25000)

• ISO/IEC 25000 개념

- 소프트웨어 품질 특성(ISO/IEC 9126) 및 품질 평가 방법을 통합한 소프트웨어 품질 평가(ISO/IEC 14598) 모델 국제 표준
- SQuaRE(System and Software Quality Requirements and Evaluation)

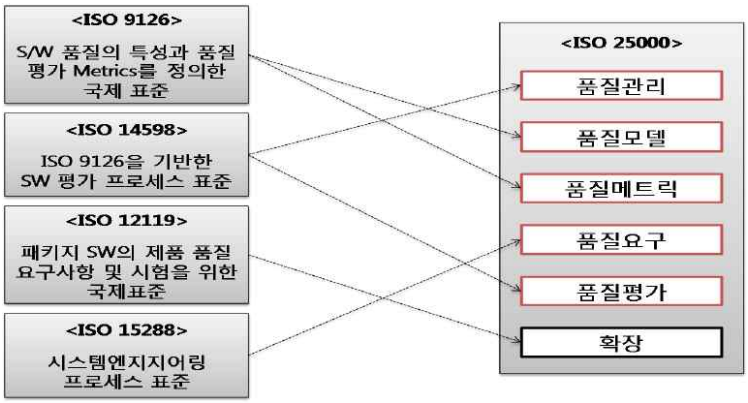

그림. 기존 표준과 ISO/IEC 25000과의 관계

• ISO/IEC 25000 구조

표준번호	구성요소	설명
2500n	품질 관리	- SQuaRE 표준의 가이드라인과 품질평가의 관리에 관한 표준 제시 - ISO 14598-2 기반
2501n	품질 모델	- 제품 품질 평가의 일반 모델 제시 - 구 표준에는 없는 데이터 품질 모델이 제정됨 - ISO 9126-1 기반
2502n	품질 측정	- 품질측정 메트릭 정의 - 소프트웨어의 내부 측정, 외부 측정, 사용품질 측정, 품질 측정 요소 등을 다룸 - ISO 9126-2, 3 ,4 기반

		※ 메트릭 : 소프트웨어의 품질 특성 등을 수치화하여 측정하고 평가하는 기준
2503n	품질 요구	- 품질 요구사항 설정 프로세스 - ISO 15288 기반
2504n	품질 평가	- 품질평가 절차를 정의 - ISO 14598 기반

* 소프트웨어 공학의 기본원칙

• 소프트웨어 개념 :
하드웨어를 동작시켜 사용자가 작업을 편리하게 수행하도록 하는 프로그램 및 자료구조

• 소프트웨어 위기(Software Crisis)

☞ 개념 : 여러 원인에 의해 소프트웨어 개발 속도가 하드웨어 개발 속도를 따라가지 못함에 따라 사용자들의 요구사항을 충족시키지 못하는 현상

☞ 원인과 극복 방안

소프트웨어 위기	소프트웨어 극복 방안
- 소프트웨어의 특징에 대한 이해 부족 - 소프트웨어의 관리 부재 - 소프트웨어 복잡도 및 크기 증가 - 소프트웨어 제품 수명주기 단축	- 공학적 접근 - 표준화 - 자동화 도구 - 품질 보증

• 소프트웨어 공학(Software Engineering)

☞ 개념

- 소프트웨어의 개발, 운용, 유지보수 및 파기에 대한 체계적 접근 방법
- 신뢰성 있는 소프트웨어를 경제적 비용으로 획득하기 위해 공학적 원리를 정립하고 이를 이용하는 방법
- 소프트웨어 위기를 극복하기 위한 방안으로 연구된 학문

☞ 원칙

- 현대적인 프로그래밍 기술을 계속적으로 적용
- 개발된 소프트웨어의 품질이 유지되도록 지속적 검증 수행
- 소프트웨어 개발 관련 사항 및 결과에 대한 명확한 기록 유지

☞ 특성

- 소프트웨어는 유지보수가 용이
- 소프트웨어는 신뢰성이 높아야 함
- 소프트웨어는 충분한 테스팅 필요

☞ 공학 관련 법칙

브룩스의 법칙	- “지체되는 소프트웨어 개발 프로젝트에 인력을 추가하는 것은 개발

(Brook's Law)	을 늦출 뿐이다” - 인력이 추가돼서 개발 생산성이 향상되지 않고, 오히려 방해된다는 의미 내표
파레토 법칙 (Pareto Principle)	- 전체 결과의 80%가 전체 원인의 20%에서 일어나는 현상을 나타낸 법칙 - 소프트웨어 테스트 원리 중 20%의 모듈에서 80%의 결함이 발견된다는 ‘결함 집중’의 원리 내포
롱테일 법칙 (Long Tail)	사소해 보이는 80%의 다수가 20%의 소수 핵심보다도 뛰어난 가치를 창출해낸다는 법칙

3. 제품 소프트웨어 버전 관리

1) 소프트웨어 버전 관리 도구

* 개념

- 형상 관리 지침을 활용하여 제품 소프트웨어의 신규 개발, 변경, 개선과 관련된 수정 사항을 관리하는 도구
- 제품 소프트웨어 버전 관리는 소프트웨어 개발과 관련하여 코드와 라이브러리, 관련 문서 등 시간의 변화에 따른 변경을 관리하는 전체 활동을 의미

* 유형

버전 관리 도구 유형	설명	도구
공유 폴더 방식	- 매일 개발 완료 파일은 약속된 위치의 공유폴더에 복사하는 방식 - 담당자 한 명이 매일 공유 폴더의 파일을 자기 PC로 복사하고 컴파일하여 에러 확인과 정상 동작 여부 확인	RCS
클라이언트/서버 방식	- 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식 - 개발자들의 현재 작업 내용과 이전 작업내용 축적에 용이 - 서로 다른 개발자가 같은 파일을 작업했을 때 경고 메시지 출력	CVS, SVN
분산 저장소 방식	- 로컬 저장소와 원격 저장소로 분리되어 관리되는 방식 - 중앙의 저장소에서 로컬에 복사한 순간 개발자 자신만의 로컬 저장소에 생성 - 개발 완료한 파일을 수정한 다음에 로컬 저장소에 우선적으로 커밋(Commit)한 이후, 다시 원격 저장소에 반영(Push)하는 방식	Git

* 도구별 특징

RCS (Revision Control System)	-CVS와 달리 소스 파일의 수정을 한 사람만으로 제한하여 다수의 사람이 파일의 수정을 동시에 할 수 없도록 파일 잠금 방식으로 버전을 관리하는 도구 - 다른 방향으로 진행된 개발 결과를 합치거나 변경 내용을 추적할 수 있는 소프트웨어 버전 관리 도구

CVS (Concurrent Versions System)	가장 오래된 형상 관리 도구 중의 하나로서 중앙 집중형 서버 저장소를 두고 클라이언트가 접속해서 버전 관리를 실행하는 도구
SVN (Subversion)	CVS와 같은 중앙 집중형 클라이언트-서버 방식이나, CVS의 단점을 보완해 가장 널리 사용되고 있는 도구
Git	- Git은 중앙 집중형 방식이 아닌 분산형 방식으로 각 PC 스스로 완전한 저장소가 구성되며, 필요에 따라 중앙 집중형 방식으로도 운영 - Git의 커밋 동작은 로컬 저장소에서 이루어지고, 푸시라는 동작으로 원격 저장소에 반영

* 사용 시, 유의사항

버전에 대한 쉬운 정보 접근성	- 형상 관리 지침에 의거 버전에 대한 정보를 언제든지 접근할 수 있어야 함 - 프로젝트 단위 접근이든, 파일 단위의 접근이든 간에 개발자가 원하는 때에 원하는 모습을 다시 구성할 수 있어야 함.
불필요한 사용자에 대한 접근 제어	- 제품 소프트웨어 개발자, 배포자 이외에 불필요한 사용자가 소스를 수정할 수 없도록 해야 함 - 중요한 파일 혹은 폴더에 대한 접근은 개발자, 배포자 등 권한이 있는 자만 접근할 수 있도록 함
동일 프로젝트에 대한 동시 사용성	- 동일한 프로젝트에 대해서 여러 개발자가 동시에 개발할 수 있어야 함. - 여러 개발자가 동일한 파일, 폴더에 접근 시 동시에 파일 수정이 일어나더라도 개발자의 수정 내역이 통합될 수 있어야 함
빠른 오류 복구	- 에러 발생 시 최대한 빠른 시간 내에 복구가 가능해야 함 - 과거 버전의 소스를 가지고 신속하게 원복할 수 있어야 함

※ 형상 관리와 버전 관리를 통해 프로젝트 비용을 관리하지 않는다.

2) 빌드 자동화 도구

* 빌드 자동화 프로세스

순서	프로세스	설명
1	컴파일	소스 코드를 바이너리 파일로 컴파일
2	패키징	바이너리 파일을 배포 형태로 패키징
3	단위테스트	단위테스트(커버리지 포함) 수행
4	정적분석	정적분석 수행
5	리포팅	분석 결과 리포팅
6	배포	패키징한 파일을 테스트 서버에 배포
7	최종빌드	최종 빌드

* 빌드 자동화 구성요소

구성요소	설명	도구
CI(Continuous Integration) 서버	빌드 프로세스를 관리하는 서버	Jenkins Hudson
SCM (Source Code Management)	- 소스 코드 형상 관리 시스템 - 소스 코드의 개정과 백업 절차를 자동화하여 오류 수정 과정을 도와줄 수 있는 시스템 - 여러 사람이 같은 프로젝트에 참여할 경우, 각자 수정 부분은 자동으로 동기화하여 팀원 전체가 볼 수 있는 시스템	SVN Git
빌드 도구	- 컴파일, 테스트, 정적분석 등을 통해 동작 가능한 소프트웨어 생성	Ant Maven
테스트 도구	작성된 테스트 코드에 따라 자동으로 테스트를 수행해주는 도구로 빌드 도구의 스크립트에서 실행	Junit Selenium
테스트 커버리지 도구	테스트 코드가 대상 소스 코드에 대해 어느 정도 커버하는지 분석하는 도구	Emma
인스펙션 도구	- 프로그램을 실행하지 않고, 소스 코드 자체로 품질을 판단할 수 있는 정적 분석 도구 - 코딩 표준 준수 검사, 코드 메트릭 측정, 중복 코드 검사, 코드 인스펙션 검사	CheckStyle Cppcheck

※ 코드 인스펙션 : 개발팀에서 작성한 개발소스 코드를 분석하여 잘못 구현된 부분을 수정하는 작업

* 빌드 자동화 도구의 기능

코드 컴파일	테스트를 포함한 소스 코드 컴파일
컴포넌트 패키징	자바의 jar파일이나 윈도의 exe파일 같은 배포할 수 있는 컴포넌트를 묶는 작업
파일 조작	파일과 디렉토리를 만들고, 복사 및 지우는 작업
개발 테스트 실행	자동화된 테스트 진행
버전 관리 도구 통합	버전 관리 시스템 지원
문서 생성	API 문서 생성
배포 기능	테스트 서버 배포 지원
코드 품질 분석	자동화된 검사 도구를 통한 코드 품질 분석

* 빌드 자동화 도구 사례

☞ 젠킨스(Jenkins)

- 자바 기반의 오픈 소스로 가장 많이 활용되는 빌드 자동화 도구
  > 지속적 통합관리(CI : Continuous integration)를 가능케 한다.
- 서블릿 컨테이너 서버 기반으로 구동되는 시스템이며 CVS, SVN, Git 등 다양한 버전 관리 도구를 지원한다.

※ 서블릿 : 서블릿 컨테이너 위에서 작동하는 웹 서비스용 자바 인터페이스의 객체

▶ 특징

쉬운 설치	- Jenkins.war 파일로 제공하여 java -jar jenkins.war 명령어면 설치 가능 - 그 자체로 서블릿 컨테이너에 배포하면 되므로, 이후에 추가적 설치나 데이터베이스가 필요 없음
친숙한 웹 GUI	- 웹 기반 그래픽 사용자 인터페이스(GUI)를 통해 쉽게 전체적인 설정 변경 가능
저장소 부하 감소	- 버전 관리 도구에서 빌드에 사용될 목록만 따로 추출하여 변경 생성할 수 있는 기능 제공 - 전체 빌드로 인한 저장소 부하 감소
최근 빌드 제공	최근 빌드 또는 최근 성공한 빌드 내역에 대한 링크 제공
실시간 피드백	RSS 또는 이메일을 통해 실시간으로 빌드 실패 내역에 대해 통지를 받아서 빌드 결과 확인 가능
분산 빌드	여러 대의 컴퓨터를 통해 분산 빌드나 테스트 가능
3rd Party 플러그인을 통한 확장	Jenkins가 지원하길 바라는 툴이나 프로세스가 있다면 본인이 직접 제작 가능

☞ 그래들(Gradle)

- 그래들은 그루비와 유사한 도메인 언어 채용, 현재 안드로이드 앱을 만드는데 필요한 안드로이드 스튜디오의 공식 빌드 자동화 시스템
- 그래들은 실행할 처리 명령들을 모아 태스크로 만든 후 태스크 단위로 실행
- 그래들은 Java, C/C++, 파이썬 등과 같은 여러 가지 언어를 지원

※ 그루비 : 자바 가상머신 위에서 동작하는 동적 스크립트 언어