

요구사항 확인

2023-04-19 스터디
채희재

소프트웨어 생명 주기 (Software Life Cycle)

소프트웨어 개발 방법론의 바탕이 되는 것으로 소프트웨어를 개발하기 위해 정의하고 운용, 유지보수 등의 과정을 각 단계별로 나눈 것이다.

소프트웨어 개발 단계와 단계별 주요 활동, 활동의 결과에 대한 산출물로 표현한다.

소프트웨어 생명주기의 다른 표현

=> 소프트웨어 생명 주기 모형, 소프트웨어 프로세스 모형 또는 소프트웨어 공학 패러다임

일반적으로 소프트웨어 생명 주기 모형에는

폭포수 모형, 프로토타입 모형, 나선형 모형, 애자일 모형등이 있다.

소프트웨어 공학의 개념(Software Engineering)

소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문 여러가지 방법론과 관리 기법들을 통하여 **소프트웨어의 품질과 생산성 향상을 목적으로** 합니다.

IEEE의 소프트웨어 공학 표준 용어 사전: 소프트웨어 개발, 운용, 유지보수, 폐기 처분에 대한 체계적인 접근 방안

Fairley: 지정된 비용과 기간 내에 소프트웨어를 체계적으로 생산하고 유지보수 하는데 관련된 기술적이고 관리적인 원리

Boehm: 과학적 지식을 소프트웨어 설계와 제작에 응용하는 것이며 이를 개발, 운용, 유지보수하는 데 필요한 문서 작성 과정

소프트웨어 공학의 기본 원칙

1. **현대적인 프로그래밍 기술**을 계속적으로 적용해야 합니다.
2. 개발된 소프트웨어의 품질이 유지되도록 **지속적으로 검증**해야 합니다.
3. 소프트웨어 개발 관련 사항 및 **결과에 대한 명확한 기록을 유지**해야 합니다.

폭포수 모형 (Waterfall Model, 선형 순차적 모형)

- 한 단계가 완전히 끝나야만 다음단계로 넘어가는 개발 방법론 - **선형 순차적 모형**
- 가장 오래되고 가장 폭넓게 사용된 전통적인 생명주기 모형으로 **고전적 생명주기 모형**이라고도 한다.
- 각 단계가 끝난 후에는 다음 단계를 수행하기 위한 **결과물이 명확하게 산출**되어야 한다.

타당성 검토 -> 계획 -> 요구분석 -> 설계 -> 구현(코딩) -> 시험(검사) -> 유지보수

프로토타입 모형(Prototype Model , 원형 모형)

- 사용자의 요구사항을 정확히 파악하기 위해 견본(prototype)을 만들어 최종결과물을 예측하는 모형이다.
- 시제품은 **사용자와 시스템사이의 인터페이스에 중점**을 두어 개발한다.
- 요구된 소프트웨어를 만드는데 구현단계에서 사용될 골격 코드가 된다.
- **소프트웨어의 개발이 완료된 시점에서 오류가 발견되는 폭포수 모형의 단점을 보완하기 위한 모형이다.**

나선형 모형(Spiral Model, 점진적 모형)

- **보ehm(Boehm)이 제안한 것으로, 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형이다.**
- 나선을 따라 돌듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발하는 것으로 **점진적 모형**이라고 한다.
- 개발하면서 발생할 수 있는 위험을 관리, 최소화 하는것을 목표로 한다.
- 점진적으로 개발과정이 반복되므로 누락되거나 추가된 요구사항을 첨가할 수 있고, 정밀하며, 유지보수 과정이 필요없다.

계획수립 -> 위험분석 -> 개발 및 검증 -> 고객평가

애자일 모형(Agile Model)

- 민첩한, 기민한이라는 의미로 **고객의 요구사항 변화에 유연하게 대응**할 수 있도록 일정한 주기를 반복하면서 개발과정을 진행
- **고객과의 소통에 초점을 맞춘 방법론**
- **스프린트(Sprint) , 이터레이션(Iteration)**이라고 불리는 짧은 개발 주기를 반복하며, 반복 주기마다 만들어지는 결과물에 대한 고객의 평가와 요구를 적극 수용한다.
- 각 개발 주기에서는 고객이 요구사항에 **우선순위**를 부여하여 개발 작업을 진행한다.
- 스크럼(Scrum), XP(eXtreme Programming), 칸반(Kanban), Lean, 크리스탈(Crystal), ASD(Adaptive Software Development), 기능 중심 개발(FDD; Feature Driven Development), DSDM(Dynamic System Development Method), DAD(Disciplined Agile Delivery) 등이 있다.

개발 핵심 가치 4가지

1. 프로세스와 도구보다는 개인과 상호작용에 더 가치를 둔다.
2. 방대한 문서보다는 실행되는 SW에 더 가치를 둔다.
3. 계약 협상 보다는 고객과의 협업에 더 가치를 둔다.
4. 계획을 따르기 보다는 변화에 반응하는 것에 더 가치를 둔다.

폭포수 모형과 애자일 비교

구분	폭포수 모형	애자일
새로운 요구사항 반영	어려움	지속적으로 반영
고객과의 의사소통	적음	지속적임
테스트	마지막에 모든 기능을 테스트	반복되는 일정 주기가 끝날 때 마다 테스트

개발중심	계획, 문서	고객
------	--------	----

스크럼 기법(Scrum)

스크럼이란 럭비에서 반칙으로 경기가 중단된 경우 양팀의 선수들이 럭비공을 가운데 두고 상대팀을 밀치기 위해 서로 대치해 있는 대형을 말한다.
이처럼 팀이 중심이 되어 개발의 효율성을 높인다.

제품 책임자(PO:Product Owner)

- 이해관계자들 중 개발될 제품에 대한 이해도가 높고, 요구사항을 책임지고 의사결정할 사람을 선정 주로 개발 의뢰자나 사용자가 담당
- 이해 관계자들의 의견을 종합하여 제품에 대한 요구사항을 작성하는 주체이다.
- 요구사항이 담긴 **백로그(backlog)**를 작성하고 **백로그에 대한 우선순위를 지정**한다.
- 팀원들이 백로그에 **스토리를 추가**할 수 있지만 **우선순위를 지정할 수 없다**.

스크럼 마스터(SM; Scrum Master)

- 스크럼 팀이 스크럼을 잘 수행할 수 있도록 객관적인 시각에서 조언을 해주는 가이드 역할을 수행한다. 팀원들을 통제하는 것이 목표가 아니다.
- 일일 스크럼 회의를 주관하여 진행사항을 점검하고, 개발 과정에서 발생한 장애요소를 공론화하여 처리한다.

개발팀(DT; Development Team)

- 제품 책임자와 스크럼 마스터를 제외한 모든 팀원으로, 개발자 외에도 디자이너, 테스터등 제품 개발을 위해 참여하는 모든 사람이 대상이다.

백로그(Backlog)

제품 개발에 필요한 요구사항을 모두 모아 우선순위를 부여해 놓은 목록을 말한다.

스토리(Story)

백로그에 담겨질 요구사항은 단어형태가 아닌 고객은 상품 주문을 위해 로그인을 수행해야한다와 같이 이야기를 서술하는 형태로 표현한다.

제품 백로그(Product Backlog)

- 제품 개발에 필요한 모든 요구사항을 우선순위에 따라 나열한 목록이다.
- 개발과정에서 새롭게 도출되는 요구사항으로 인해 지속적으로 업데이트 된다.
- 제품 백로그에 작성된 사용자 스토리를 기반으로 전체일정 계획인 릴리즈 계획을 수립한다.(Release Plan)

스프린트 계획 회의(Sprint Planning Meeting)

- 제품 백로그 중 이번 스프린트에서 수행할 작업을 대상으로 단기 일정을 수립하는 것
- 스프린트에서 처리할 요구사항(**User Story**)을 개발자들이 나눠서 작업할 수 있도록 **Task**라는 작업 단위로 분할한 후 개발자별로 수행할 작업 목록인 스프린트 백로그(**Sprint Backlog**)를 작성한다.

스프린트 (Sprint)

- 실제 개발 작업을 진행하는 과정으로, 보통 2주~4주 정도의 기간내에서 진행한다.
- 스프린트 백로그에 작성된 태스크를 대상으로 작업 시간을 추정한 후 개발 담당자에게 할당한다.
- 개발 담당자에게 할당된 태스크는 보통 할 일(TO DO), 진행중(In Process), 완료(Done)의 상태를 갖는다.

일일 스크럼 회의(Daily Scrum Meeting)

- 모든 팀원이 약속된 시간에 약 15분 정도의 짧은 시간동안 진행 상황을 점검한다.
- 회의는 보통 서서 진행되며, 남은 작업 시간은 소멸차트(Burn-down Chart)에 표시한다.



- 스크럼 마스터는 발견된 장애 요소를 해결할 수 있도록 도와준다.

스프린트 검토회의 (Sprint Review)

- 부분 또는 전체 완성 제품이 요구사항에 잘 부합되는지 사용자가 포함된 참석자 앞에서 테스트를 수행한다.
- 스프린트 한 주당 한 시간내에서 진행한다.
- 제품 책임자(Product Owner)는 개선할 사항에 대한 피드백을 정리한 후 다음 스프린트에 반영할 수 있도록 제품 백로그를 업데이트 한다.

스프린트 회고(Sprint Retrospective)

- 스프린트 주기를 되돌아보며 규칙을 잘 준수했는지 확인
- 끝난 시점에 함

XP(eXtreme Programming)

수시로 발생하는 고객의 요구사항에 유연하게 대응 하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향상시키는 방법이다.

XP는 짧고 반복적인 개발주기, 단순한 설계, 고객의 적극적인 참여를 통해 소프트웨어를 빠르게 개발하는 것을 목적으로 한다.

릴리즈의 기간을 짧게 반복하면서 고객의 요구사항 반영에 대한 가시성을 높인다.

릴리즈 테스트 마다 고객을 직접 참여시킴으로써 요구한 기능이 제대로 작동하는지 고객이 직접 확인할 수 있다.

소규모 인원의 개발 프로젝트에 효과적이다.

XP의 5가지 핵심 가치 :

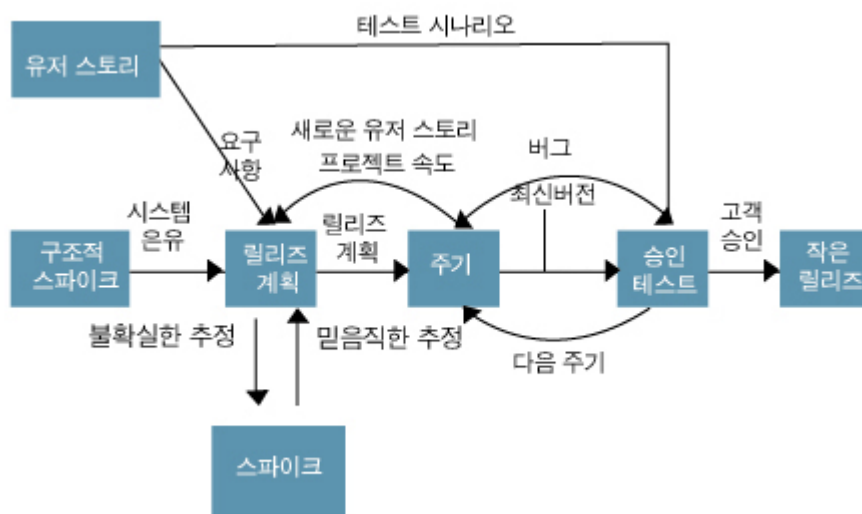
의사소통(Communication),

단순성(Simplicity),

용기(Courage),

존중(Respect),

피드백(Feedback)



사용자 스토리 (user story)

- 고객의 요구사항을 간단한 시나리오로 표현한 것이다.
- 간단한 테스트(Test Case) 사항도 기재함

릴리즈 계획 수립(Release Planning)

- 몇 개의 스토리가 적용되어 부분적으로 기능이 완료된 제품을 제공하는 것을 릴리즈라고 한다.
- 일정을 수립한다.

스�파이크(Spike)

- 요구사항의 신뢰성을 높이고 기술 문제에 대한 위험을 감소시키기 위해 별도로 만드는 프로그램
- 처리할 문제 외의 다른조건은 모두 무시한다.

이터레이션(iteration)

- 하나의 릴리즈를 더 세분화 한 단위를 이터레이션이라고 한다.
- 이 기간 중에 새로운 스토리가 작성될 수 있으며 현재 이터레이션 또는 다음 이터레이션에 포함될 수 있다.

승인 검사(Acceptance Test, 인수테스트)

- 하나의 이터레이션 안에서 계획된 릴리즈 단위의 부분 완료 제품이 구현되면 수행하는 테스트이다.
- 사용자 스토리 작성 시 함께 기재한 테스트 사항에 대해 고객이 직접 수행한다.
- 테스트 과정에서 나온 오류사항은 다음 이터레이션에 포함

소규모 릴리즈(small Release)

- 릴리즈를 소규모로 하게 된다면, 고객의 반응을 기능별로 확인할 수 있어, 고객의 요구사항에 좀 더 유연하게 대응할 수 있다.
- 계획된 릴리즈 기간 동안 진행된 이터레이션이 모두 완료되면 고객에 의한 최종 테스트를 수행한 후 릴리즈, 즉 최종 결과물을 고객에 전달한다.

XP의 주요 실천 방법

Pair Programming (짝 프로그래밍)	다른 사람과 함께 프로그래밍을 수행함으로써 개발에 대한 책임을 공동으로 나눠 갖는 환경을 조성합니다.
Collective Ownership (공동 코드 소유)	개발코드에 대한 권한과 책임을 공동으로 소유합니다.
Test-Driven Development(테스트 주도 개발)	개발자가 실제 코드를 작성하기 전에 테스트 케이스를 먼저 작성하므로 자신이 무엇을 해야할지를 정확히 파악합니다. 자동화된 테스트 프레임워크를 사용합니다.
Whole Team (전체 팀)	개발에 참여하는 모든 구성원 들은 각자 자신의 역할이 있고 그 역할에 대한 책임을 가져야합니다.
Continuous Integration (계속 적인 통합)	모듈 단위로 나눠서 개발된 코드들은 하나의 작업이 마무리될때마다 지속적으로 통합됩니다.
Design Improvement (디자인 개선 또는 Refactoring(리팩토링))	프로그램 기능의 변경 없이, 단순화, 유연성 강화 등을 통해 시스템을 재구성합니다.
Small Releases (소규모 릴리즈)	릴리즈 기간을 짧게 반복함으로써 고객의 요구변화에 신속히 대응할 수 있습니다.

Scrum은 프로젝트 관리 방법론이고, XP는 개발 방법론 입니다.

구분	Scrum	XP
방법론	프로젝트 관리 방법론	개발 방법론
강조	이벤트를 강조	실천방법을 강조

참조 사이트 : <http://blog.skby.net/xp-extreme-programming/>

개발 기술 환경 파악

개발 하고자 하는 소프트웨어와 관련된 운영체제(Operating System), 데이터 베이스 관리 시스템 (Database Management System), 미들웨어(Middle Ware)등을 선정할 때 고려해야 할 사항을 기술하고 오픈 소스 사용시 주의해야할 내용을 제시한다.

미들 웨어(Middleware)

운영체제와 해당 운영체제에 의해 실행되는 응용 프로그램 사이에서 운영체제가 제공하는 서비스에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어이다.

운영체제(OS, Operating System)

컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어이다.

컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해 준다.

컴퓨터 운영체제의 종류에는 Windows, UNIX, Mac OS 등이 모바일 운영체제에는 iOS, Android 등이 있다.

운영체제 관련 요구사항 식별시 고려사항

구분	내용
가용성 : 프로그램이 주어진 시점에서 요구사항에 따라 운영될 수 있는 능력 (메모리 누수: 응용프로그램이 더 이상 사용하지 않는 메모리를 반환하지 않고 계속 점유하고 있는 현상)	- 시스템의 장시간 운영으로 인해 발생할 수 있는 운영체제 고유의 장애 발생 가능성 - 메모리 누수로 인한 성능 저하 및 재가동 -보안상 발견된 허점을 보안하기 위한 지속적인 패치 설치로 인한 재가동 -운영체제의 결함 등으로 인한 패치 설치를 위한 재가동
성능	대규모 동시 사용자 요청에 대한 처리 대규모 및 대용량 파일 작업에 대한 처리 지원 가능한 메모리 크기(32bit, 64bit)
기술 지원	제작업체의 안정적인 기술 지원 여러 사용자들 간의 정보 공유

	오픈 소스 여부(Linux)
주변기기	설치가 가능한 하드웨어 여러 주변 기기 지원 여부
구축 비용	지원 가능한 하드웨어 비용 설치할 응용프로그램의 라이선스 정책 및 비용 유지관리 비용 총 소유 비용(TCO)

데이터베이스 관리 시스템 DBMS

DBMS (DataBase Management System) 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해주는 소프트웨어이다.

DBMS는 기존의 파일 시스템이 갖는 **데이터의 종속성과 중복성의 문제**를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해준다.

DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.

DBMS의 종류에는 Oracle, IBM DB2, MySQL, SQL, MongoDB, Redis 등이 있다.

DBMS 관련 요구사항 식별 시 고려사항

구분	내용
가용성	시스템의 장시간 운영으로 인해 발생할 수 있는 운영체제 고유의 장애 발생 가능성 DBMS의 결함 등으로 인한 패치 설치를 위한 재가동 백업이나 복구의 편의성 DBMS 이중화 및 복제 지원
성능	대규모 데이터 처리 성능 대용량 트랜잭션 처리 성능 튜닝 옵션의 다양한 지원 최소화된 설정과 비용기반 질의 최적화 지원
기술 지원	제작업체의 안정적인 기술 지원 여러 사용자들 간의 정보 공유

	오픈소스 여부
상호 호환성	<p>설치 가능한 운영체제의 종류</p> <p>JDBC, ODBC와의 호환 여부</p> <p>JDBC: java에서 DB에 접근하여 조회, 삽입, 수정할 수 있는 인터페이스</p> <p>ODBC: 응용 프로그램에서 DB에 접근하여 조회 삽입 수정할 수 있는 인터페이스</p>
구축비용	<p>라이선스 정책 및 비용</p> <p>유지관리 비용</p> <p>총 소유 비용(TCO)</p> <p>Total Cost of Ownership:</p> <p>어떤 자산을 획득하려고 할 때 지정된 기간 동안 발생할 수 있는 모든 직간접 비용</p>

웹 애플리케이션 서버(WAS; Web Application Server)

정적인 콘텐츠 처리를 하는 웹 서버와 달리 사용자 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어이다.

데이터 접근, 세션 관리, 트랜잭션 관리 등을 위한 라이브러리를 제공한다.

주로 데이터베이스 서버와 연동해서 사용한다.

웹 애플리케이션 서버의 종류에는 Tomcat, GlassFish, JBoss, Jetty, 등등이 있다.

웹 애플리케이션 서버 관련 요구사항 식별 시 고려사항

구분	내용
가용성	<p>-시스템의 장시간 운영으로 발생할 수 있는 고유의 장애 발생 가능성</p> <p>-WAS의 결함 등으로 인한 패치 설치를 위한 재가동</p> <p>-안정적인 트랜잭션 처리</p> <p>-WAS 이중화 지원</p>
성능	<p>-대규모 트랜잭션 처리 성능</p> <p>-다양한 설정 옵션 지원</p> <p>-가비지 컬렉션(GC; Garbage Collection)의 다양한 옵션</p>
기술지원	//
구축비용	//

오픈소스 사용에 따른 고려사항

오픈 소스 (Open Source)는 누구나 별 다른 제한 없이 사용할 수 있도록 소스코드를 공개한 것으로 오픈 소스 라이선스를 만족하는 소프트웨어이다.

요구사항의 개념 및 특징

- 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영되는데 필요한 제약조건 등을 나타낸다.
- 요구사항은 소프트웨어 개발이나 유지 보수 과정에서 필요한 기준과 근거를 제공한다.
- 이해 관계자들 간의 의사소통을 원활하게 하는 데 도움을 준다.
- 요구사항이 제대로 정의되어야만 이를 토대로 이후 과정의 목표와 계획을 수립할 수 있다.

요구사항 유형

요구사항은 기술하는 내용에 따라 기능 요구사항(Function requirement)과 비기능 요구사항(Non-functional requirement)으로 구분하며, 기술 관점과 대상의 범위에 따라 시스템 요구사항(System requirements)과 사용자 요구사항(User requirement)으로 나뉜다.

유형	내용
기능 요구사항 (Function requirement)	시스템이 무엇을 하는지, 어떤 기능을 하는지에 대한 사항 시스템의 입력이나 출력으로 무엇이 포함되어야 하는지, 시스템이 어떤 데이터를 저장하거나 연산을 수행하는지에 대한 사항 시스템이 반드시 수행해야하는 기능 사용자가 시스템을 통해 제공받기를 원하는 기능
비기능 요구사항 (Non-functional requirement)	시스템 장비 구성 요구사항: 하드웨어, 소프트웨어, 네트워크 등의 시스템 장비에 대한 요구사항 성능요구사항: 처리속도 및 시간, 처리량, 동적정적 적용량, 가용성 등등 인터페이스 요구사항 데이터 요구사항 테스트 요구사항 보안요구사항

	품질요구사항 제약사항 프로젝트 관리 요구사항 프로젝트 지원 요구사항
사용자 요구사항 (System requirements)	사용자 관점에서 본 시스템이 제공해야 할 요구사항
시스템 요구사항 (User requirement)	개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항 사용자 요구사항에 비해 전문적이고 기술적인 용어로 표현된다. 소프트웨어 요구사항이라고도 한다.

요구사항 개발 프로세스

개발 대상에 대한 요구사항을 체계적으로 도출하고 이를 분석한 후 분석 결과를 명세서(Specification Document)에 정리한 다음 마지막으로 이를 확인 및 검증하는 일련의 구조화된 활동이다.

요구사항 개발 프로세스가 진행되기 전에 개발 프로세스가 비즈니스 목적에 부합되는지, 예산은 적정한지 등에 대한 정보를 수집, 평가한 보고서를 토대로 타당성 조사(Feasibility Study)가 선행되어야 한다.

도출
(Elicitation)
분석
(Analysis)
명세
(Specification)
확인
(Validation)

요구사항 도출(Requirement Elicitation, 요구사항 수집)

요구사항 도출은 사람들이 서로 의견을 교환하여 요구사항이 어디에 있는지, 어떻게 수집할 것인지를 식별하고 이해하는 과정이다.

소프트웨어가 해결해야 할 문제를 이해하는 첫 번째 단계이다.

요구사항 도출 단계에서 개발자와 고객 사이의 관계가 만들어지고 이해관계자가 식별된다.

이 단계에서는 다양한 이해관계자 간의 효율적인 의사소통이 중요하다.

요구사항 도출은 개발 주기 동안 지속적으로 반복된다.

도출하는 주요 기법은 청취, 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스 등이 있다.

요구사항 분석(Requirement Analysis)

요구사항 분석은 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정이다.

- 사용자 요구사항의 타당성을 조사하고 비용과 일정에 대한 제약을 설정
- 내용이 중복되거나 하나로 통합되어야 하는 등 서로 상충되는 요구사항이 있으면 이를 중재하는 과정이다.
- 도출된 요구사항을 토대로 소프트웨어의 범위를 파악한다.
- 도출된 요구사항들을 토대로 소프트웨어의 범위를 파악한다.
- 도출된 요구사항들을 토대로 소프트웨어와 주변 환경이 상호 작용하는 방법을 이해한다.
- 요구사항 분석에는 **자료흐름도(DFD)**, **자료 사전(DD)** 등의 도구가 사용된다.

요구사항 명세(Requirement Specification)

분석된 요구사항을 바탕으로 모델을 작성하고 문서화하는 것을 의미한다.

요구사항을 문서화할 때는 기능 요구사항은 빠짐없이 완전하고 명확하게 기술해야 하며, 비기능 요구사항은 필요한것만 명확하게 기술

구체적인 명세를 위해 **소단위 명세서(Mini - Spec)**가 사용될 수 있다.

소프트웨어 요구사항 명세서(SRS; Software Requirement Specification)

요구사항 확인(Requirement Validation)

개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지 검토하는 활동이다.

- 분석가가 요구사항을 정확하게 이해한 후 요구사항 명세서를 작성했는지 확인(**Validation**)하는 것이 중요하다.
- 일반적으로 요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 형상관리를 수행한다.
-

* 형상관리(SCM; Software Configuration Management)

소프트웨어 개발 단계의 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터등을 통칭하여 형상이라고 합니다. 그것들을 관리하는것이 형상관리라고 합니다.

요구사항 분석의 개요

요구사항 분석은 소프트웨어 개발의 실제적인 첫 단계로 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화(명세화)하는 활동을 의미한다.

구조적 분석 기법

자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법





- 도형 중심의 분석용 도구와 분석 절차를 이용하여 사용자의 요구사항을 파악하고 문서화 한다.
- 도형 중심의 도구를 사용하므로 분석가와 사용자간의 대화가 용이하다.
- **하향식 방법**을 사용하여 시스템을 세분화할 수 있고, 분석의 중복을 배제할 수 있다.
- 구조적 분석 기법에는 자료흐름도(DFD), 자료사전(DD), 소단위 명세서(Mini-Spec),개체 관계도(ERD), 상태 전이도(STD), 제어 명세서등이 있다.

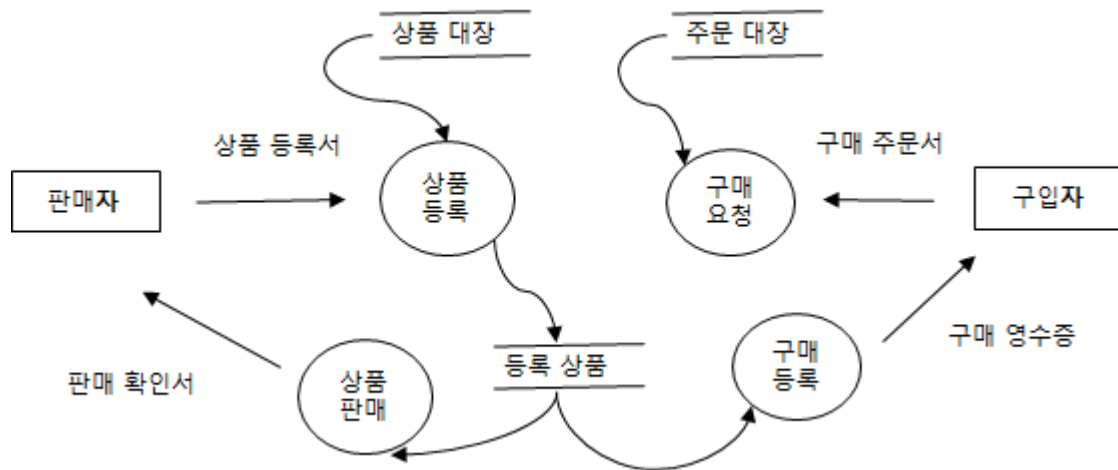
하향식 방법: 소프트웨어의 모든 기능을 모델링 할 수 없으므로 소프트웨어의 기능을 전체적인 수준에서 상세수준까지 위에서 아래로 단계별로 분리하여 모델링하는 것을 의미합니다.

자료흐름도(DFD; Data Flow Diagram)

요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법으로 자료 흐름 그래프, 버블차트라고 한다.

- 자료는 처리(Process)를 거쳐 변환될 때마다 새로운 이름이 부여되며, 처리는 입력 자료가 발생하면 기능을 수행한 후 출력 자료를 산출한다.
- 자료 흐름도에서는 자료의 흐름과 기능을 프로세스(Process), 자료흐름(Flow), 자료 저장소 (Data Store), 단말(Terminator) 네가지 기본 기호로 표시한다.

기 호	의 미	표기법
프로세스 (Process)	자료를 변환시키는 시스템의 한 부분(처리 과정을 나타내며, 처리 기능, 변환, 버블이라고도 함)	
자료 흐름(Flow)	자료의 이동을 나타냄	
자료 저장소 (Data Store)	시스템에서의 자료 저장소(파일, 데이터 베이스)를 나타냄	
단말 (Terminator)	시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고, 출력 데이터를 받음 (정보의 생산자와 소비자)	



자료 사전(DD; Data Dictionary)

자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것이며, 이처럼 데이터를 설명하는 데이터를 데이터의 데이터 또는 **메타 데이터(Meta Data)** 라고 합니다.

기호	의미
=	자료의 정의: ~로 구성되어있다. (is composed of)
+	자료의 연결: 그리고(and)
()	자료의 생략: 생략가능한 자료(Optional)
[]	자료의 선택: 또는(or)
{ }	자료의 반복: Iteration of
* *	자료의 설명

요구사항 분석 CASE와 HIPO

CASE (자동화 도구)

요구사항 분석을 위한 자동화 도구는 요구사항을 자동으로 분석하고, 요구사항 분석 명세서를 기술하도록 개발된 도구를 의미한다.

요구사항 분석을 위한 자동화 도구 사용의 이점

- 표준화와 보고를 통한 문서화 품질 개선
- 데이터 베이스가 모두에게 이용 가능하다는 점에서 분석자들 간의 적절한 조정
- 변경이 주는 영향 추적의 용이성
- 명세에 대한 유지보수 비용의 축소

종류

SADT, SREM, PSL/PSA, TAGS, EPOS

SADT(Structured Analysis and Design Technique)

- SoftTech사에서 개발, 시스템 정의, 소프트웨어 요구사항 분석, 시스템/소프트웨어 설계를 위해 널리 이용
- 구조적 요구분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구

SREM(Software Requirement Engineering Methodology) = RSL/REVS

- TRW사가 우주 국방 시스템 그룹에 의해 실시간 처리 소프트웨어 시스템에서 요구사항 명확히 기술하도록 할 목적으로 개발 RSL/REVS는 사용하는 자동화 도구

RSL(Requirement Statement Language)

- 요소, 속성, 관계, 구조들을 기술하는 요구사항 기술 언어

요소	요구사항 명세를 개발하기 위해 사용되는 개체와 개념
속성	요소를 수정하거나 수식하기 위한 것
관계	개체들 간의 관계
구조	정보 흐름을 묘사하기 위한 것

REVS(Requirement Engineering and Validation System)

- RSL로 기술된 요구사항을 자동으로 분석하여 요구사항 분석 명세서를 출력하는 요구사항 분석기

PSL/PSA

- 미시간 대학에서 개발함
- PSL(Problem Statement Language) 문제 기술 언어
- PSA(Problem Statement Analyzer) PSL로 기술한 요구사항을 자동으로 분석하여 다양한 보고서를 출력

TAGS(Technology for Automated Generation of Systems)

- 시스템 공학 방법 응용에 대한 자동 접근 방법으로, 개발 주기의 전 과정에 이용할 수 있는 통합 자동화 도구이다.
- IOTL : 요구사항 명세 언어

HIPO(Hierarchy Input Process Output)

시스템의 분석 및 설계나 문서화할 때 사용되는 기법, 시스템 실행 과정인 입력 처리, 출력의 기능을 나타낸다.

- 기본 시스템 모델은 입력, 출력, 처리으로 구성되며, 하향식 소프트웨어 개발을 위한 문서 도구이다.
- 체계적인 문서관리, 기호와 도표등을 사용하여 이해가 쉽고, 기능 자료의 의존관계를 동시에 표현 가능하고 변경 유지보수가 용이하다.
- 시스템의 기능을 여러 개의 고유 모듈로 분할하여 이들 간의 인터페이스를 계층구조로 표현한 것을 HIPO chart 라고 한다.

HIPO chart의 종류

가시적 도표(Visual Table of Contents), 총체적 도표(Overview Diagram), 세부적 도표(Detail Diagram)가 있다.

가시적 도표(도식 목차): 시스템의 전체적인 기능과 흐름을 보여주는 계층(Tree)구조

총체적 도표 (총괄도표, 개요 도표): 프로그램을 구성하는 기능을 기술한 것 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표

세부적 도표: 총체적 도표에 표시된 기능을 구성하는 기본요소들을 상세히 기술

UML(Unified Modeling Language)

시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어

- Rumbaugh,Booch,Jacobson 등의 객체 지향 방법론의 장점을 통합하였으며, 객체 기술에 관한 국제 표준기구인 OMG(Object Management Group)에서 표준으로 지정하였다.
- UML을 이용하여 시스템 구조를 표현하는 6개의 구조 다이어그램과 시스템 동작을 표현하는 7개의 행위 다이어그램을 작성할 수 있다.
- UML 구성요소는 사물(Things),관계(Relationships),다이어그램(Diagram)등이 있다.

사물(Things)

사물은 모델을 구성하는 가장 중요한 기본 요소로, 다이어그램 안에서 관계가 형성될 수 있는 대상을 말한다.

사물	내용
구조사물 (Structural Things)	시스템의 개념적,물리적 요소를 표현 클래스(class) , 유스케이스(UseCase), 컴포넌트(Component), 노드(Node) 등
행동 사물 (Behavioral Things)	시간과 공간에 따른 요소들의 행위를 표현 상호작용(Interaction), 상태머신(State Machine) 등
그룹 사물 (Group Things)	요소들을 그룹으로 묶어서 표현 패키지(Package)
주해 사물(Annotation Things)	부가적인 설명이나 제약조건 등을 표현

* 컴포넌트: 문서, 소스코드,파일 라이브러리 등과 같은 모듈화된 자원으로 재사용이 가능합니다.

관계 (Relationships)

사물과 사물 사이의 연관성을 표현하는 것으로, 연관 관계, 집합관계, 포함관계, 일반화 관계, 의존 관계, 실체화 관계 등이 있다.

연관 관계(Association)

2개 이상의 사물이 서로 관련되어 있음을 나타낸다.

연관에 참여하는 객체의 개수를 의미하는 다중도(Multiplicity)를 선위에 표시한다.

집합 관계(Aggregation)

하나의 사물이 다른 사물에 포함되어 있는 관계를 표현

포함관계(Composition)

집합관계의 특수한 형태로, 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 포함

일반화 관계(Generalization)

하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현한다.

의존 관계(Dependency)

연관 관계와 같이 사물 사이에 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계

실체화 관계(Realization)

사물이 할 수 있거나 해야하는 기능(행위, 인터페이스)으로 서로 그룹화 할 수 있는 관계를 표현

관계	UML 표기
Generalization (일반화)	
Realization (실체화)	
Dependency (의존)	
Association (연관)	
Directed Association (직접연관)	
Aggregation (집합, 집합연관)	
	
Composition (합성, 복합연관)	
	

<https://www.nextree.co.kr/p6753/>

다이어그램(Diagram)

사물과 관계를 도형으로 표현한 것이다.

정적인 모델링에서는 구조적 다이어그램을 사용하고, 동적 모델링에서는 주로 행위 다이어그램을 사용한다.

다이어그램 종류 설명

<https://seulhee030.tistory.com/56>

중요한 다이어그램 3가지

유스케이스 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램(순서 다이어그램)

액터: 시스템과 상호작용하는 모든 외부 요소로 주로 사람이나 외부시스템을 의미함