Certified Professional for Requirements Engineering

파운데이션 레벨

실러버스

Stan Bühne
Martin Glinz
Hans van Loenhoud
Stefan Staal

3.2.0 | 2024 년 2 월 26 일

IREB

이용 약관

1. 개인 및 교육기관은 저작권이 인정되고 교육 자료에 명기되는 조건하에 이 실러버스를 교육의 기초 자료로 사용할 수 있다. 이 실러버스를 광고 목적으로 사용하기 위해선 IREB e.V. 의 서면 동의를 받아야 한다.
2. 개인 또는 단체는 본 문서의 출처 및 소유자인 저자와 IREB e.V.의 저작권을 인정하는 경우에 한해 본 실러버스를 기사, 서적 또는 기타 파생 출판물의 기초로 사용할 수 있다.

감사의 말

이 실러버스는 2007 년에 카롤 프뤼하우프, 에머리히 푹스, 마틴 글린츠, 라이너 그라우, 콜린 후드, 프랭크 후덱, 피터 후쉬카, 바바라 파치, 클라우스 폴, 크리스 루프가 처음 만들었습니다. 이안 알렉산더, 조셉 브루더, 사무엘 프리커, 귄터 할만스, 피터 예슈케, 스벤 크라우스, 스테펜 렌츠, 우르테 파우츠, 수잔 로버트슨, 디르크 슈페링, 요하네스 스타우브, 토르스텐 바이어, 조이 비티가 이 실러버스스 제작을 지원했습니다.

Ver. 3 은 스탠 뷘, 마틴 글린츠, 한스 반 로엔후드, 스테판 스탈이 함께 만든 주요 개정판입니다. 카롤 프뤼하우프, 라이너 그라우, 킴 라우엔로스, 크리스 루프, 카밀 살리네시 등이 이 실러버스 개정판 제작을 지원했습니다.

이번 개정 과정에서 자비에 프랜치, 카롤 프루하우프, 라이너 라우, 프랭크 후덱, 토르스텐 바이어가 피드백을 제공했습니다. 여기에 빔 데쿠테레와 한스-요르그 스테페가 추가 피드백을 제공했으며.

크리스토프 에버트, 바바라 패치, 크리스 루프가 리뷰를 수행했습니다.

자비에 프랜치와 프랭크 후덱의 추천에 따라 2020 년 7 월 22 일에 IREB 위원회에서 출시가 승인되었습니다.
V.3.1.0 한국어 버전은 KSTQB 의 권선이, STA 테스팅컨설팅의 송홍진이 번역했습니다.

참여해주신 모든 분들께 감사드립니다.

서문

2017 년 여름에 기존 CPRE (Certified Professional for Requirements Engineering) 파운데이션 레벨 자격증(ver.2.2)의 관련성을 조사하기 위한 설문조사를 실시했다. 설문조사의 목적은 교육기관의 관점과 요구사항 엔지니어로 일하는 공인 CPRE 실무자 [MFeA2019] 로부터 이 인증의 실질적인 시장 관련성에 대한 피드백을 얻기 위한 것이었다. 설문조사 결과, 기존 CPRE 파운데이션 레벨 실러버스 v2.2 가 여전히 시장의 가장 중요한 요구사항을 충족하고 있으며 응시자에게 관련 RE 지식을 전달한다는 점이 반영되었다. 그럼에도 불구하고 몇 가지 기법은 더 이상 실무에서 사용되지 않는 반면, 다른 기법은 더 반복적이고 적응적인 개발을 위해 누락되었다는 피드백을 받았다. 이 피드백은 요구사항 엔지니어링(RE) 분야의 변화에 대한 IREB 의 자체 인식과 일치했다. 따라서 오래된 콘텐츠를 버리고 새로운 요소를 추가해 CPRE 파운데이션 레벨 실러버스를 대대적으로 개정했다. 실러버스 ver.3 은 요구사항을 지정하고 관리하기 위한 계획 중심 접근 방식과 애자일 접근 방식을 모두 다루는 최신 RE 의 상태를 반영한다.

이 실러버스에 따라 CPRE 인증을 받으려는 응시자는 계획 중심적이고 애자일한 접근법으로 시스템 개발에 대한 기본적인 지식을 갖추고 있어야 한다.

문서의 목적

이 실러버스는 국제 요구공학 위원회(IREB)에서 제정한 요구공학 공인 전문가 인증의 파운데이션 레벨을 정의한다. 실러버스는 교육 제공자에게 강의자료 제작의 기초를 제공한다. 학생들은 실러버스를 사용하여 시험에 대비할 수 있다.

실러버스 내용

파운데이션 레벨은 요구공학 주제와 관련된 모든 사람들의 요구사항을 다룬다. 여기에는 요구사항 엔지니어, 비즈니스 분석가, 시스템 분석가, 제품 소유자 또는 제품 관리자, 개발자, 프로젝트 또는 IT 관리자, 도메인 전문가와 같은 역할을 하는 사람들이 포함된다.

본 실러버스 및 관련 핸드북에서는 요구공학의 약어 "RE"를 사용한다.

콘텐츠 범위

CPRE 파운데이션 레벨은 모든 유형의 시스템(예: 모바일 앱, 정보 시스템 또는 사이버-물리 시스템)에 동일하게 적용되는 기본 원칙을 전달한다. 또한 CPRE 파운데이션 레벨은 특정 개발 프로세스를 가정하지 않으며, 특정 애플리케이션 도메인에 맞춰져 있지도 않다. 교육기관은 본 실러버스의 학습 목표를 충분히 다루기만 한다면 특정 유형의 시스템, 프로세스 또는 애플리케이션 도메인에 초점을 맞춘 교육을 제공할 수 있다.

세부 수준

이 실러버스의 세부 수준으로 교육과 시험이 국제적으로 일관성을 유지할 수 있다. 이 목표를 달성하기 위해 실러버스에는 다음이 포함되어 있다:

- 일반적인 학습 목표
- 학습 목표에 대한 설명이 포함된 콘텐츠
- 추가 문헌 참조 (필요한 경우)

학습 목표 / 인지 지식 수준

이 실러버스의 모든 모듈과 학습 목표에는 인지 수준이 할당되어 있다. 다음과 같은 수준이 사용된다:

- **L1: 알기** (설명하기, 열거하기, 특성화하기, 인식하기, 이름 짓기, 기억하기, ...) – 이전에 학습한 내용을 기억하거나 되새긴다.
- **L2: 이해** (설명, 해석, 완성, 요약, 정당화, 분류, 비교 등) – 주어진 자료나 상황에서 의미를 파악/구성한다.
- **L3: 적용** (지정, 작성, 설계, 개발, 구현 등) – 주어진 상황에서 지식과 기술을 적용한다.

상위 수준에는 하위 수준이 포함된다. 학습 목표에 명시적으로 언급되어 있지 않더라도 용어집에서 기본 용어로 지정된 모든 용어는 반드시 알고 있어야 한다 (L1). 용어집은 다음 IREB 홈페이지에서 다운로드 가능하다 https://www.ireb.org/downloads/#cpre-glossary

실러버스의 구조

실러버스는 7 개의 주요 장으로 구성되어 있다. 각 장은 하나의 교육 단위(EU)를 다룬다. 주요 장 제목에는 해당 장의 인지 레벨이 포함되어 있고, 이는 하위 장의 가장 높은 레벨이다. 기간은 교육 과정이 해당 장에 투자해야 하는 시간을 나타낸다. 교육기관은 더 많은 시간을 자유롭게 할당할 수 있지만 EU(교육단위) 간의 비율을 유지해야 한다. 장에서 사용되는 중요한 용어는 장의 시작 부분에 나열되어 있다.

예시

EU 4 요구사항 정교화를 위한 프랙티스 (L3)

소요시간: 4시간 30분

용어: 요구사항 출처, 시스템 경계, 시스템 정황, 요구사항 도출, 요구사항 유효성 검사, 이해관계자, 카노 모델, 충돌, 충돌 해결

이 예에서는 4 장에 L3 수준의 학습 목표가 포함되어 있으며 이 장의 내용을 가르치는 데 4 시간 30 분이 소요됨을 보여준다.

각 장에는 하위 장이 포함되어 있다. 제목에는 장 내용의 인지 수준도 포함되어 있다.

학습 목표(EO)는 실제 텍스트 앞에 열거되어 있다. 번호매김은 해당 하위 장이 속해있는 장을 나타낸다. 예를 들어, 학습 목표 EO 4.2.1 은 하위 장 4.2 에 설명되어 있다.

실러버스의 주제 순서

이 실러버스의 장 순서는 논리적 순서에 따라 주제를 구성한다. 그러나 주제를 정확히 이 순서대로 가르칠 필요는 없다. 교육기관은 교육 맥락에 적합하고 교육 개념에 부합한다고 판단되는 순서(다른 EU 의 주제를 섞는 것 포함)로 자유롭게 자료를 활용해 가르칠 수 있다.

시험

이 실러버스는 CPRE 파운데이션 레벨 자격증 시험의 기초가 된다.

시험의 한 문제는 실러버스의 여러 장에 있는 내용을 다룰 수 있다. 실러버스의 모든 챕터(EU 1 ~ EU 7)에서 시험문제가 출제될 수 있다.

시험 형식은 객관식이다.

시험은 교육 과정 직후에 실시할 수도 있지만, 교육 과정과 독립적으로 실시할 수도 있다 (예: 시험 센터에서 실시). IREB 공인 인증 기관 목록은 웹사이트 https://www.ireb.org 에서 확인할 수 있다.

버전 기록

버전	날짜	콘텐츠
3.1.0	2023 년 5 월 22 일	영문 버전 3.1.0 을 기반으로 한 첫 한국어 버전
3.1.1	2024 년 1 월 1 일	새로운 기업 디자인
3.2.0	2024 년 2 월 26 일	새로운 기업 디자인을 위한 레이아웃 수정

콘텐츠

1 요구공학의 소개 및 개요 (L2)

목표: 요구공학(RE)이 무엇인지 알고 요구공학의 가치를 이해한다.

소요시간: 1시간

용어: 요구사항, 요구사항 명세, 요구공학(RE), 이해관계자, 시스템, 요구사항 엔지니어

학습 목표

EO 1.1.1 기본 용어를 기억한다 (L1)
EO 1.1.2 다양한 종류의 요구사항을 이해한다 (L2)
EO 1.2.1 요구공학의 가치를 설명한다 (L2)
EO 1.2.2 부적절한 요구공학의 증상을 열거한다 (L1)
EO 1.3.1 요구공학을 어디에 적용할 수 있는지, 어디에서 요구사항이 발생하는지를 파악한다 (L1)
EO 1.4.1 요구공학의 주요 업무와 이를 수행하기 위한 요구공학 프로세스가 정형화되어야 함을 이해한다 (L1)
EO 1.5.1 요구사항 엔지니어의 역할과 업무를 정의한다 (L1)
EO 1.6.1 요구사항 엔지니어가 학습해야 하는 사항을 기억한다 (L1)

1.1 요구공학: What (L1)

사람과 조직은 새로운 것을 구축하거나 기존 것을 발전시키고자 하는 욕구와 필요를 가지고 있다. 우리는 이러한 필요를 *요구사항*이라고 부른다.

구축하거나 발전시켜야 할 것들의 예를 들자면:

- 고객에게 제공되는 *제품*
- 고객에게 제공되는 *서비스*
- 사람이나 조직이 특정 목표를 달성하는 데 도움이 되는 장치, 절차 또는 도구 등의 기타 *결과물*
- 제품, 서비스 또는 기타 결과물의 *구성* 또는 *구성 요소*

이 모든 것을 *시스템*으로 간주할 수 있다. 이 실러버스에서는 *이해관계자*의 요구사항을 포함하는 모든 종류의 것들을 설명하는 데 *시스템*이라는 용어를 쓴다. *이해관계자*는 시스템 요구사항에 영향을 미치거나 해당 시스템의 영향을 받는 개인 또는 조직이다.

요구공학의 목표는 구현 및 배포된 시스템이 이해관계자의 요구와 필요를 충족할 수 있도록 시스템에 대한 요구사항을 지정하고 관리하는 것이다.

요구공학에서는 3 가지 종류로 요구사항을 구분한다 [Glin2020]:

- *기능 요구사항*은 시스템의 기능이 제공해야 하는 결과 또는 동작과 관련이 있다. 여기에는 데이터에 대한 요구사항 또는 시스템과 환경의 상호작용이 포함된다.
- *품질 요구사항*은 성능, 가용성, 보안 또는 신뢰성과 같이 기능 요구사항에서 다루지 않는 품질 문제와 관련이 있다.
- *제약 조건*은 주어진 기능 요구사항 및 품질 요구사항을 충족하는 데 필요한 것 이상으로 솔루션 공간을 제한하는 요구사항이다.

1.2 요구공학: Why (L2)

적절한 요구공학은 시스템을 개발하고 발전시키는 과정을 통해 다음과 같은 *가치*를 더한다:

- 잘못된 시스템 개발의 리스크 감소
- 문제에 대한 이해도 향상
- 개발 노력 및 비용 추정의 기준
- 시스템 테스팅을 위한 사전 조건

부적절한 요구공학의 일반적인 증상은 요구사항이 누락되거나, 불분명하거나, 잘못되는 경우이다. 이는 특히 다음과 같은 이유 때문에 발생한다:

- 급하게 시스템 구축에 바로 돌입
- 관련 당사자 간의 커뮤니케이션 문제
- 요구사항이 자명하다는 가정
- 불충분한 요구공학 교육과 기술

1.3 요구공학: Where (L2)

요구공학은 모든 종류의 시스템 요구사항에 적용할 수 있다. 그러나 오늘날 요구공학의 주요 적용 사례는 주로 소프트웨어가 중요한 역할을 하는 시스템 대상이다. 이러한 시스템은 일반적으로 소프트웨어 컴포넌트, 물리적 요소 및 조직 요소로 구성된다.

요구사항은 다음과 같이 발생할 수 있다:

- *시스템 요구사항* – 시스템이 수행해야 하는 작업
- *이해관계자 요구사항* – 이해관계자의 관점에서 원하는 것
- *사용자 요구사항* – 사용자 관점에서 사용자가 원하는 것
- *도메인 요구사항* – 필수 도메인 속성

- *비즈니스 요구사항* - 조직의 비즈니스 목표, 목적 및 필요성

1.4 요구공학: How (L1)

요구공학의 주요 작업은 요구사항 도출(4.2), 문서화(3), 유효성 검증(4.4), 관리(6)이다. 도구 지원(7)이 이러한 작업을 수행하는 데 도움이 될 수 있다. 요구사항 분석과 요구사항 충돌 해결(4.3)은 도출의 일부로 간주된다. 요구공학 활동을 제대로 수행하기 위해서는 다양한 정황에 맞는 적합한 요구공학 프로세스를 구현해야 한다 (5).

1.5 요구사항 엔지니어의 역할과 업무 (L1)

요구사항 엔지니어는 일반적으로 직책이 아니라, 다음과 같이 사람들이 수행하는 *역할*을 의미한다:

- 업무의 일환으로 요구사항을 도출, 문서화, 검증하거나 관리한다.
- 요구공학에 대한 심층적인 지식이 있다.
- 문제와 잠재적 솔루션 사이의 간극을 메울 수 있다.

실무에서는 비즈니스 분석가, 애플리케이션 전문가, 제품 소유자, 시스템 엔지니어, 심지어 개발자가 요구사항 엔지니어의 역할을 수행한다.

1.6 요구공학에 대해 학습할 내용 (L1)

이 실러버스에서는 요구사항 엔지니어가 배워야 하는 기본 기술을 다룬다. 요구공학의 기본 원칙(2), 다양한 형식으로 요구사항을 문서화하는 방법(3), 다양한 사례를 통해 요구사항을 구체화하는 방법(4), 적합한 요구공학 프로세스를 정의하고 작업하는 방법(5), 기존 요구사항을 관리하는 방법(6), 도구 지원을 사용하는 방법(7)을 다룬다.

2 요구공학의 기본 원칙 (L2)

목표: 요구공학의 원리를 알고 이해한다.

소요시간: 1시간 30분

용어: 정황, 요구사항, 요구공학(RE), 이해관계자, 공통 이해, 유효성 검사

학습 목표

EO 2.1.1 요구공학의 원칙을 열거한다 (L1)
EO 2.2.1 원칙과 관련된 용어를 기억한다 (L1)
EO 2.2.2 원칙과 원칙이 중요한 이유를 설명한다 (L2)

2.1 원칙 개요 (L1)

요구공학는 요구공학의 모든 작업, 활동 및 프랙티스에 적용되는 일련의 기본 원칙에 따라 관리된다. 다음 9 가지 원칙은 이 실러버스의 다음 장에 나오는 프랙티스의 기초를 형성한다.

1. 가치 지향: 요구사항은 그 자체가 목적이 아니라 목적을 위한 수단이다
2. 이해관계자: 요구공학는 이해관계자의 욕구와 필요를 충족시키는 것이다
3. 공통 이해: 공통 기반 없이는 성공적인 시스템 개발이 불가능하다
4. 정황: 시스템은 따로 분리해서 이해할 수 없다
5. 문제 – 요구사항 – 솔루션: 필연적으로 얽혀있는 3 가지
6. 유효성 검증: 검증되지 않은 요구사항은 쓸모가 없다
7. 진화: 요구사항 변경은 우연이 아니라 일반적인 것이다
8. 혁신: 더 많은 동일함만으로는 충분하지 않다
9. 체계적이고 규율적인 작업: 요구공학에서는 필수이다

2.2 원칙 설명 (L2)

> 원칙 1 - 가치 지향: 요구사항은 그 자체가 목적이 아니라, 목적을 위한 수단이다.

요구사항의 가치는 요구사항의 이점에서 요구사항을 도출, 문서화, 검증 및 관리하는 데 드는 비용을 뺀 값과 같다. 요구사항의 이점은 해당 요구사항이 기여하는 정도이다:

- 이해관계자의 욕구와 필요를 충족하는 시스템을 구축한다.
- 시스템 개발 시 실패 위험과 비용이 많이 드는 재작업을 줄일 수 있다.

원칙 2 - 이해관계자: 요구공학은 이해관계자의 욕구와 필요를 충족시키는 것이다

요구공학는 이해관계자의 욕구와 필요를 이해하는 것이므로 이해관계자를 잘 대하는 것은 요구공학의 핵심 과제이다. 사용자, 클라이언트, 고객, 운영자, 규제 기관 등 모든 이해관계자는 구축할 시스템의 맥락에서 각자의 역할을 담당한다. 이해관계자는 동시에 두 가지 이상의 역할을 수행할 수도 있다. 개인이 너무 많거나 개인을 알 수 없는 이해관계자 역할의 경우, 이해관계자를 대표하는 *페르소나*라는 것을 대신 정의할 수 있다. 최종 사용자 또는 고객의 요구사항만 고려하는 것만으로는 충분하지 않다. 이렇게 하면 다른 이해관계자의 중요한 요구사항을 놓칠 수 있다. 사용 중인 시스템에 대한 피드백을 제공하는 사용자도 이해관계자로 간주해야 한다.

이해관계자마다 요구사항과 관점이 다를 수 있으며, 이로 인해 요구사항이 상충될 수 있다. 이러한 갈등을 파악하고 해결하는 것이 요구공학의 임무이다.

성공적인 요구공학를 위해서는 관련 이해관계자 역할에 적합한 사람을 참여시키는 것이 중요하다. 이해관계자를 식별하고, 우선순위를 정하고, 협력하기 위한 관행은 4 에 설명되어 있다.

원칙 3 - 공통 이해: 공통 기반 없이는 성공적인 시스템 개발이 불가능하다

요구공학는 이해관계자, 요구사항 엔지니어, 개발자 등 관련 당사자 간의 공유된 이해를 도출, 촉진 및 확보한다. 공통 이해에는 두(2) 가지 형태가 있다:

- *명시적인 공통 이해*(문서화되고 합의된 요구사항에 의해 달성됨)
- *암묵적인 공통 이해*(요구사항, 비전, 정황 등에 대한 공통 지식을 기반으로 함)

도메인 지식, 과거의 성공적인 협업, 공유된 문화와 가치 및 상호 신뢰는 공통 이해를 가능하게 하는 요소인 반면, 지리적 거리, 아웃소싱 또는 이직률이 높은 대규모 팀은 장애물이 될 수 있다.

공통 이해 달성을 위한 입증된 방법으로는 용어집 제작(3.5), 프로토타입 작성(3.7) 또는 기존 시스템을 참고해 사용하는 방법이 있다. 공통 이해를 평가하는 방법에는 예상 결과의 예시,

프로토타입 탐색 또는 요구사항 구현 비용 추정 등이 있다. 오해의 영향을 줄이기 위한 가장 적절한 방법은 피드백 루프가 짧은 프로세스를 사용하는 것이다(5).

원칙 4 - 정황: 시스템은 서로 고립된 상태로 이해할 수 없다

시스템은 *정황*에 내장되어 있다. 이러한 정황을 이해하지 못하면 시스템을 제대로 지정할 수 없다. 요구공학에서 시스템의 정황은 시스템 및 요구사항을 이해하는 데 관련된 시스템 환경의 일부이다. *시스템 경계*는 시스템과 주변 정황 사이의 경계이다. 처음에는 시스템 경계가 명확하지 않은 경우가 많으며, 시간이 지남에 따라 변경될 수 있다.

시스템 경계를 명확히 하고 시스템과 상호작용하는 정황 요소 간의 외부 인터페이스를 정의하는 것이 진정한 요구공학 작업이다. 동시에 시스템의 *범위*, 즉 시스템 개발 시 구현 및 설계할 수 있는 사물의 범위도 결정해야 한다. 또한 시스템 환경의 요구공학 관련 부분을 나머지 세계와 구분하는 소위 *정황 경계*도 고려해야 한다.

시스템 지정 시 시스템 경계 내의 요구사항만 고려하는 것만으로는 충분하지 않다. 요구공학에서는 다음 사항도 고려해야 한다:

- 시스템 요구사항에 영향을 줄 수 있는 정황의 변경 사항.
- 시스템과 관련된 실제 요구사항 (및 이를 시스템 요구사항에 매핑하는 방법).
- 시스템을 작동시키고 실제 요구사항을 충족하기 위해 유지해야 하는 정황에 대한 가정.

원칙 5 - 문제 - 요구사항 - 솔루션: 필연적으로 연관되어 있는 3 가지 문제

이해관계자가 현재의 상황에 만족하지 않을 때 *문제*가 발생한다. *요구사항*은 이런 문제를 제거하거나 단순화하기 위해 이해관계자가 필요로 하는 것을 찾아낸다. 이러한 요구사항을 충족하는 사회 기술 시스템이 *솔루션*을 구성한다.

문제, 요구사항, 솔루션이 반드시 이 순서대로 발생하는 것은 아니다. 솔루션 아이디어를 요구사항으로 구체화해서 실제 솔루션으로 구현해야 하는 사용자 요구를 만들어 낼 수 있다. 일반적으로 혁신할 때 이런 경우가 많다.

- 문제, 요구사항, 솔루션은 서로 밀접하게 연관되어 있어 따로 떼어놓고 다룰 수 없다.

- 그럼에도 불구하고, 요구사항 엔지니어는 생각하고, 소통하고, 문서화를 진행할 때 문제, 요구사항, 솔루션을 가능한 한 서로 분리하는 것을 목표로 한다. 이렇게 문제를 분리하면 요구공학 작업을 더 쉽게 처리할 수 있다.

원칙 6 - (유효성) 검증: 검증되지 않은 요구사항은 쓸모가 없다

결과적으로 배포된 시스템이 이해관계자의 요구와 필요를 충족하는지 검증해야 한다. 이해관계자의 불만족 리스크를 처음부터 제어하려면 요구공학 단계에서 요구사항 검증을 미리 시작해야 한다.
다음 사항을 확인해야 한다:

- 이해관계자 간에 요구사항에 대한 합의가 이루어졌는가?
- 이해관계자의 욕구와 필요가 요구사항에 적절히 반영되어 있는가?
- 가정한 정황(위 원칙 4 참조)은 합리적인가?

요구사항의 유효성을 검사하는 방법은 4.4 에서 확인할 수 있다.

원칙 7 - 진화: 요구사항 변경은 우연이 아니라 일반적인 현상이다

시스템과 그 요구사항은 *진화*할 수 있다. 즉, 끊임없이 변화한다는 뜻이다. 예를 들어, 시스템에 대한 요구사항 또는 일련의 요구사항 변경 요청은 예를 들어 다음과 같은 이유로 발생할 수 있다:

- 변경된 비즈니스 프로세스
- 새로운 제품 또는 서비스를 출시하는 경쟁사
- 우선순위 또는 의견을 변경하는 고객
- 기술의 변화
- 법률 또는 규정의 변경
- 새로운 기능이나 변경된 기능을 요청하는 시스템 사용자의 피드백

또한 요구사항 검증 시 이해관계자의 피드백, 이전에 도출된 요구사항에서의 결함 발견이나 이해관계자의 니즈 변경으로 인해 요구사항이 변경될 수 있다.

결과적으로 요구사항 엔지니어는 상반되는 두 가지 목표를 추구해야 한다:

- 요구사항 변경 허가
- 요구사항의 안정적 유지

이를 달성하는 방법에 대한 자세한 내용은 6.7 에서 확인할 수 있다.

원칙 8 - 혁신: 더 많은 동일한 것만으로는 충분하지 않다

이해관계자가 원하는 것만을 정확하게 제공하면 이해관계자의 기대보다 만족스러운 시스템 구축 기회는 놓치게 된다. 좋은 요구공학는 이해관계자를 만족시키는 것에서 나아가 그들이 행복하고, 신나고, 안전하다고 느낄 수 있도록 노력해야 한다. 이것이 바로 혁신의 궁극적인 목표이다.

요구공학는 혁신적인 시스템을 구축한다:

- 소규모로는 흥미로운 새로운 기능과 사용 편의성을 위해 노력한다.
- 대규모로는 획기적인 새로운 아이디어를 위해 노력한다.

4.2 에서는 요구공학의 혁신을 촉진하기 위한 몇 가지 기술에 대해 논의한다.

원칙 9 - 체계적이고 규율적인 작업: 요구공학에서 필수적 요소이다

사용 중인 실제 개발 프로세스에 관계없이 요구사항을 체계적으로 도출, 문서화, 검증, 관리하기 위한 적절한 프로세스와 프랙티스을 채택할 필요가 있다. 임시방편으로 시스템을 개발하는 경우에도 체계적이고 규율화된 방식으로 요구공학에 접근하면 결과 시스템의 품질이 향상된다.

요구공학에는 주어진 모든 상황, 또는 적어도 대부분의 상황에서 잘 작동하는 단일 프로세스나 프랙티스은 존재하지 않는다. 체계적이고 규율적인 작업은 요구사항 엔지니어가 다음을 수행함을 의미한다:

- 주어진 문제, 정황 및 환경에 맞게 프로세스와 관행을 조정한다.
- 항상 동일한 프로세스와 프랙티스을 사용하지 않는다.
- 과거 성공적인 요구공학 작업의 프로세스와 프랙티스을 회고없이 재사용하지 않는다.

모든 요구공학 노력에 대해 특정 상황에 가장 적합한 프로세스, 프랙티스 및 작업 산출물을 선택해야 한다. 자세한 내용은 3, 4, 5, 6 에 자세히 설명되어 있다.

3 작업 산출물과 문서화 (L3)

목표: 요구공학에서의 작업 산출물의 기본 역할을 이해하고 산출물을 만들 수 있다

소요시간: 7시간

용어: 작업 산출물, 자연어 기반 작업 산출물, 템플릿 기반 작업 산출물, 모델 기반 작업 산출물, 용어, 품질 기준, 요구사항 명세

학습 목표

EO 3.1.1 요구공학 작업 산출물의 특성을 파악하고 자주 사용되는 작업 산출물의 유형을 열거한다 (L1)
EO 3.1.2 각 작업 산출물의 사용 가능 용도와 수명을 파악한다 (L1)
EO 3.1.3 적절한 추상화 수준과 세부 수준 선택법을 포함하는, 요구사항에 대한 다양한 추상화 수준을 설명한다 (L2)
EO 3.1.4 제품 산출물에서 고려해야 할 측면과 이러한 측면 간의 상호 관계를 파악한다 (L1)
EO 3.1.5 일반적인 문서화 가이드라인의 이름을 지정한다 (L1)
EO 3.1.6 사용할 작업 산출물을 계획하는 것이 왜 중요한 지 설명한다 (L1)
EO 3.2.1 자연어 기반 작업 산출물과 그 장단점을 파악한다 (L1)
EO 3.2.2 우수한 자연어 기반 요구사항 작성 규칙을 설명한다 (L2)
EO 3.3.1 템플릿 기반 작업 산출물의 범주와 장단점을 파악한다 (L1)
EO 3.3.2 구문 템플릿을 사용해 개별 요구사항 및 사용자 스토리를 작성한다 (L3)
EO 3.3.3 양식 템플릿을 사용해 유스 케이스를 지정한다 (L3)
EO 3.4.1 요구공학에서 모델의 역할, 목적 및 사용법을 이해한다 (L2)
EO 3.4.2 요구공학에서 모델링의 장점과 한계를 이해한다 (L2)
EO 3.4.3 모델, 모델링 언어, 활동 모델, 활동 다이어그램, 클래스 모델, 클래스 다이어그램, 정황 모델, 정황 다이어그램, 도메인 모델, 목표 모델, 상호작용 모델, 프로세스 모델, 시퀀스 다이어그램, 상태 차트, 상태 머신, 상태 머신 다이어그램, 사용 사례, 유스케이스 다이어그램 용어를 정의한다 (L1)
EO 3.4.4 주어진 상황에서 요구사항 지정에 적합한 모델 유형을 선택하는 방법을 이해한다 (L2)
EO 3.4.5 (해당하는 경우 UML로 작성한) 정황 모델, 유스 케이스 유스 케이스 다이어그램, 도메인 모델, 클래스 모델, 활동 모델, 프로세스 모델 및 상태 차트와 같은 간단한 모델을 이해하고 해석한다 (L2)
EO 3.4.6 UML 클래스 다이어그램을 사용해 시스템 데이터 또는 도메인 대상에 대한 간단한 모델을 정한다 (L3)
EO 3.4.7 UML 활동 다이어그램으로 간단한 시스템 기능이나 비즈니스 프로세스를 정한다
EO 3.5.1 용어집의 목적과 작성 방법을 설명한다 (L2)
EO 3.6.1 자주 사용되는 요구사항 명세서서를 파악한다 (L1)
EO 3.6.2 어떤 문서 구조가 어떤 목적에 부합하는지, 그 구조화 기준은 무엇인지 설명한다 (L2)
EO 3.7.1 다양한 종류의 프로토타입과 그 용도를 파악한다 (L1)
EO 3.8.1 단일 요구사항에 대한 품질 기준을 파악한다 (L1)
EO 3.8.2 작업 산출물에 대한 품질 기준을 파악한다 (L1)

3.1 요구공학의 작업 산출물 (L2)

작업 산출물은 작업 프로세스에서 만들어진 중간 결과물이나 최종 결과물의 기록이다. 예를 들어, 요구공학에는 단기간만 사용되고 사라지는 그래픽 스케치부터 계속 진화하는 사용자 스토리 모음, 수백 페이지에 달하는 정식 계약 요구사항 명세 문서에 이르기까지 다양한 작업 산출물이 있다.

3.1.1 작업 산출물의 특성 (L1)

작업 산출물은 목적, 표현 방식, 크기, 수명에 따라 각기 다른 특성이 있다. 다음 작업 산출물은 주어진 목적을 위해 실무에서 자주 발생하는 작업물이다. 작업 산출물에는 다른 작업 산출물이 포함될 수 있다.

- 단일 요구사항에 대한 작업 산출물에는 개별 요구사항과 사용자 스토리가 포함된다.
- 일관된 요구사항 집합을 위한 작업 산출물에는 유스케이스, 특정 유형의 그래픽 모델(3.4), 작업 설명, 외부 인터페이스 설명 및 에픽이 있다.
- 포괄적인 문서나 문서 구조를 구성하는 작업 산출물에는 시스템 요구사항 명세, 제품과 스프린트 백로그, 스토리 맵 등이 있다.
- 다른 산출물로는 용어집, 텍스트 노트, 그래픽 스케치, 프로토타입 등이 있다.

산출물은 다양한 형태로 *표현* 된다:

- 자연어 기반 (3.2)
- 템플릿 기반 (3.3)
- 모델 기반 (3.4)
- 드로잉이나 프로토타입과 같은 기타 표현물 (3.7)

대부분의 작업 산출물은 파일, 데이터베이스, 또는 요구공학 도구에 전자적으로 *저장된다*. 비공식적이고 임시적인 작업 산출물은 종이나 포스트잇 메모 등 다른 매체에 저장할 수도 있다 (예: 칸반 보드의 포스트잇 메모).

작업 산출물의 수명과 연관지어어 보는 경우에는 다음 세 가지로 구분한다:

- *임시 작업 산출물:* 커뮤니케이션을 지원하고 이해를 돕기 위해 제작된다.
- *진화하는 작업 산출물:* 보통 시간이 지남에 따라 여러차례 반복해 생기며, 일부 메타데이터(6.5)가 필요하고, 변경 제어가 적용될 수 있다.
- *내구성 작업 산출물: 기준이 설정되었거나 출시되었으며,* 전체 메타데이터 세트가 필요하고(6.5), 반드시 변경 제어를 따라야 한다 (6.3, 6.4).

임시 작업 산출물을 보관하고 메타데이터를 추가해 진화하는 작업 산출물로 변환할 수 있다. 마찬가지로, 일시적이거나 진화하는 작업 산출물도 기준이 정해지거나 출시되어 내구성 작업 산출물이 될 수 있다.

3.1.2 추상화 수준 (L2)

요구사항은 일반적으로 새로운 비즈니스 프로세스에 대한 높은 수준의 요구사항부터 이벤트 관련 예외적인 특정 소프트웨어 컴포넌트의 반응과 같이 아주 세부적인 수준의 요구사항에 이르기까지 *다양한 추상화 수준*으로 존재한다.

적절한 추상화 수준의 선택은 지정할 대상과 명세의 목적에 따라 달라진다. 그러나 추상화 수준이 다른 요구사항을 혼용하지 않아야 한다. 중소 규모의 작업 산출물에서는 요구사항이 어느 정도 동일한 수준의 추상화 수준을 가져야 한다.

시스템 요구사항 명세 같은 대규모 작업 산출물에서는 서로 다른 추상화 수준에 대한 요구사항 명세서를 적절하게 구조화해서 별도로 보관해야 한다 (3.6). 추상화 수준이 높은 요구사항은 보다 낮은 구체적인 수준의 여러 요구사항으로 세분화될 수 있다.

비즈니스 요구사항 명세, 이해관계자 요구사항 명세, 또는 비전 문서와 같이 높은 수준의 비즈니스나 이해관계자 요구사항이 내구성 작업 산출물로 표현되는 경우에는 이들이 시스템 요구사항 명세에 우선한다. 다른 환경에서는 비즈니스 요구사항, 이해관계자 요구사항, 시스템 요구사항이 같이 진화할 수 있다.

3.1.3 세부 수준 (L2)

요구사항의 *세부* 수준은 특히 다음과 같은 몇 가지 요인에 따라 달라진다:

- 문제와 개발 정황
- 문제에 대한 공통 이해 정도
- 설계와 프로그래머에게 주어진 자유의 정도
- 설계와 구현 과정에서 주어지는 신속한 이해관계자 피드백
- 세부 사양의 비용 대비 가치
- 준수해야 하는 표준 및 규제 제약 사항

지정된 요구사항의 세부 수준이 높을수록 예상치 못했거나 정하지 않은 사항이 발생할 위험이 낮아진다. 그러나 세부 수준이 높아질수록 명세에 대한 비용도 증가한다.

3.1.4 작업 산출물에서 고려해야 할 측면 (L1)

작업 산출물에 요구사항을 명세화할 때는 다음과 같은 여러 측면을 고려해야 한다.

1. 요구사항은 종류(1.1)에 따라 다음과 같이 분류된다:
   a) 기능 요구사항
   b) 품질 요구사항
   c) 제약 조건
2. 기능 요구사항은 시스템 기능의 다양한 측면에 중점을 둔다:
   a) 구조 및 데이터
   b) 기능 및 흐름
   c) 상태 및 동작
3. 마지막으로, 요구사항은 정황을 통해서만 이해할 수 있다 (원칙 4 에서 2):
   a) 외부 액터를 포함하는 *시스템 정황*
   b) 시스템 *경계* 및 외부 인터페이스

앞서 언급된 측면들 간에는 많은 상호 관계와 의존성이 있다. 예를 들어, 사용자(정황)의 요청에 대해 주어진 시간(품질) 안에 사용자(정황)에게 결과를 제공하기 위해 데이터(구조와 데이터)가 필요한 또 다른 작업(기능과 흐름)을 시작하는 상태 전환(상태와 동작)을 일으킬 수 있다.

일부 작업 산출물은 특정 측면에 초점을 맞추고 다른 측면은 추상적으로 표현한다. 특히 요구사항 모델(3.4)의 경우 더욱 그렇다. 시스템 요구사항 명세와 같은 다른 작업 산출물은 이러한 모든 측면을 다룬다. 서로 다른 측면이 별도의 작업 산출물이나 동일한 작업 산출물의 별도 장에 문서화되어 있는 경우, 이러한 작업 산출물이나 장은 서로 일관성을 유지해야 한다.

3.1.5 일반 문서 가이드라인 (L1)

사용된 기술과 관계없이 작업 산출물을 만들 때는 다음 가이드라인이 적용된다:

- *의도한 목적*에 맞는 작업 산출물 유형을 선택한다.
- 동일한 콘텐츠를 다시 반복하는 대신 콘텐츠를 참조해 *중복을 피한다*.
- 특히 서로 다른 측면을 다루는 경우, 작업 산출물 간에 *불일치가 없는지 확인한다*.
- 용어집에 정의된 대로 *일관성 있게 용어를 사용한다*.
- 작업 산출물을 적절하게 *구성한다*.

3.1.6 사용할 작업 산출물 계획 (L1)

각 프로젝트 설정과 각 도메인은 서로 다르므로 각 노력에 대해 작업 산출물 결과의 세트를 정의해야 한다. 따라서 다음 사항에 동의해야 한다:

- 어떤 작업 산출물에 어떤 목적으로 요구사항을 기록해야 하는가?
- 어떤 추상화 수준을 고려해야 하는가?
- 각 추상화 수준에서 요구사항을 어느 수준까지 상세하게 문서화해야 하는가?
- 이러한 작업 산출물에서 요구사항을 어떻게 표현해야 하는가?

사용할 작업 산출물은 프로젝트의 초기 단계에서 정의해야 한다. 여기에는 다음과 같은 몇 가지 장점이 있다:

- 노력과 리소스를 계획하는 데 도움이 된다.
- 적절한 표기법을 사용했는지 확인할 수 있다.
- 모든 결과가 올바른 작업 산출물에 기록되도록 보장한다.
- 정보를 크게 재구성하지 않고 '최종 편집'이 필요하지 않도록 한다.
- 중복을 피할 수 있어 업무가 줄어들고 유지 관리가 쉬워진다.

3.2 자연어 기반 작업 산출물 (L2)

체계적인 요구공학가 시작된 이래 자연어 요구사항은 실제로 요구사항을 작성하는 핵심 수단으로 사용되었다.

자연어 기반 작업 산출물에는 다음과 같은 주요 장점이 있다:

- 제약이 없는 자연어는 표현력이 뛰어나고 유연하다.
- 모든 측면에서 상상할 수 있는 거의 모든 요구사항을 자연어로 표현할 수 있다.
- 자연어는 일상 생활에서 사용되며 학교에서 가르치기 때문에 자연어로 된 텍스트를 읽고 이해하는 데 특별한 훈련이 필요하지 않다.

이러한 장점과는 달리 자연어로 작성된 텍스트는 종종 다른 방식으로 해석될 수 있다는 단점도 있으며, 이는 요구사항을 작성할 때 문제가 될 수 있다. 또한 이러한 텍스트에서 모호함, 누락 및 불일치를 알아내는 것은 어렵고 비용이 많이 든다.

다음과 같은 방법으로 자연어 기반 요구사항의 작성을 지원할 수 있다:

- 문장을 짧고 체계적으로 작성한다.
- 통일되게 용어를 정의하고 일관되게 사용한다 (3.5).
- 모호하거나 애매모호한 용어와 문구를 피한다.
- 아래에 열거된 기술 문서 작성의 함정을 알아둔다.

자연어로 기술 문서를 작성할 때 피하거나 주의해서 사용해야 하는 몇 가지 잘 알려진 함정이 있다 [GoRu2003].

피해야 할 것들:

- 불완전한 설명
- 불특정 명사
- 불완전한 조건
- 불완전한 비교

주의해서 사용해야 할 것들:

- 수동태
- 보편 양화사 ('모두' 또는 '절대' 등)
- 명사화 (예: '증명'과 같이 동사에서 파생된 명사)

3.3 템플릿 기반 작업 산출물 (L3)

템플릿 기반 작업 산출물은 요구사항에 대해 미리 정의된 구조를 제공함으로써 자연어 기반 작업 산출물의 단점을 극복하는 데 사용된다.

- *구문 템플릿*은 요구사항, 특히 개별 요구사항이나 사용자 스토리를 표현하는 구문에 대해 미리 정의된 구문 구조를 제공한다.
- *양식 템플릿*은 유스케이스 또는 측정 가능한 품질 요구사항 작성과 같이 작성할 양식에 미리 정의된 필드 세트를 제공한다.
- *문서 템플릿*은 요구사항 문서에 대해 미리 정의된 구조를 제공한다.

다양한 템플릿이 문헌에 설명되어 있다. [ISO29148], [MWHN2009], [Rupp2014] 에서 개별 요구사항에 맞는 구문 템플릿을 제공한다. [Cohn2004] 에서는 사용자 스토리에 널리 사용되는 구문 템플릿을 정의하고 [Cock2001] 에서는 유스케이스에 대한 양식 템플릿에 대해 설명한다. [Laue2002] 에서는 작업 설명을 위한 템플릿을 제안했다. [ISO29148] 와 [RoRo2012] 에서는 전체

명세에 대한 문서 템플릿을 제공한다. 또한 고객이 프로젝트에서 고객별 템플릿을 사용하도록 지정할 수도 있다.

템플릿 기반 작업 산출물의 *장점*:

- 명확하고 재사용 가능한 구조 제공
- 가장 관련성 높은 정보를 얻도록 지원
- 요구사항 및 요구사항 명세를 일관성 있게 표시
- 요구사항 및 요구사항 명세의 전반적인 품질의 개선

템플릿 기반 작업 산출물의 *단점*과 함정:

- 사람들은 종종 내용보다는 템플릿의 형식적인 완성도에 집중한다.
- 템플릿에 포함되지 않은 측면은 생략될 가능성이 높다.

3.4 모델 기반 작업 산출물 (L3)

자연어로 표현된 요구사항은 특히 일련의 요구사항에 대한 개요를 파악하고 요구사항 간의 관계를 이해하는 데 한계를 가지고 있다 ([Davi1993]). 요구사항 모델링은 이러한 한계를 해결한다.

3.4.1 요구공학에서 모델의 역할 (L2)

*모델*은 현실의 원래있는 부분 또는 현실에 만들어질 부분을 추상적으로 표현한 것이다. 현실이라는 개념에는 다른 모델을 포함해 생각할 수 있는 모든 요소, 현상 또는 개념 집합이 들어있다. 모델과 관련해 현실의 모델링된 부분을 *원본*이라고 한다. 소프트웨어 공학 도메인 이외의 예로는 건물과 기타 건설요소를 계획, 구축, 관리하는 데 필요한 요소를 모델링하는 건물 정보 모델(BIM)([ISO19650])을 예로들 수 있다.

요구공학에서 모델은 요구사항 간의 관계와 상호 연결을 이해하고 일련의 요구사항에 대한 개요를 제공하는 데 도움이 된다. 이는 주로 행동과 같은 일부 측면에 집중하고 다른 모든 측면은 추상화함으로써 달성 가능하다. 모델에 대한 그래픽 표기법을 사용해 개요를 파악하는 것도 도움이 된다. 그러나 모델은 표와 같은 비그래픽 방식으로도 표현할 수 있다.

요구사항 모델은 자연어로 표현되는 요구사항과 비교해 다음과 같은 장점이 있다:

- 요구사항 간의 관계와 상호 연결은 자연어로 지정할 때보다 그래픽 모델을 사용하면 더 쉽게 이해할 수 있다.
- 한 가지 측면에 집중하면 모델링된 요구사항을 이해하는 데 필요한 인지적 부하가 줄어든다.

- 요구사항 모델링 언어의 제한된 구문은 모호함과 누락 가능성을 줄여준다.

모델에는 또한 다음과 같은 한계가 있다:

- 서로 다른 측면에 초점을 맞춘 모델을 일관성 있게 유지하는 것은 어려운 일이다.
- 인과관계를 이해하려면 서로 다른 모델의 정보를 통합해야 한다.
- 모델은 주로 기능적 요구사항에 초점을 맞추기 때문에 대부분의 품질 요구사항과 제약 조건은 합리적인 노력으로 모델에 표현되지 않는다.
- 그래픽 모델링 언어의 제한된 구문은 모든 관련 정보 항목을 모델에 표현할 수 없다는 것을 의미한다.

따라서 요구사항 모델과 자연어로 된 요구사항을 자주 결합한다.

요구공학에서는 모델을 다음 용도로 사용할 수 있다:

- 텍스트로 표현된 요구사항을 대체하기 위한 수단으로 (주로 기능적인) 요구사항의 일부 또는 전체를 *지정*한다.
- 복잡한 현실을 잘 정의되고 상호 보완적인 측면으로 *분해*하고, 각 측면을 특정 모델로 표현한다.
- 특히 요구사항 간의 관계에 대한 이해도를 높이기 위해 텍스트로 표현된 요구사항을 *의역*한다.
- 누락, 모호함 및 불일치를 발견하기 위해 텍스트로 표현된 요구사항을 *검증*한다.

*모델링 언어*는 모델을 표현하는 데 사용한다. UML [OMG2017] 또는 BPMN [OMG2013] 과 같은 여러 모델링 언어가 표준화되었다. 요구사항이 비표준 모델링 언어로 지정된 경우 사용된 모델링 언어의 구문과 의미를 설명하는 범례가 필요하다.

요구공학에서 사용할 수 있는 모델에는 여러 가지 유형이 있다. 요구사항 엔지니어는 주어진 상황에서 요구사항을 지정하는 데 가장 적합한 모델 유형을 이해해야 한다.

초기 단계에서 요구사항 엔지니어는 정황(3.4.2) 또는 의도한 시스템의 목표를 모델링하는 것으로 시작하는 경우가 많다.

3.4.2 정황 모델링 (L2)

정황 측면에 초점을 맞춘 모델은 시스템의 환경 내 구조적 결합과 시스템 정황에서 시스템과 액터 간의 상호작용을 지정한다.

*정황 모델*은 시스템과 시스템 정황에서 시스템과 상호작용하는 액터를 지정한다. 정황 모델은 또한 시스템과 정황 간의 인터페이스를 스케치한다 (예: 교환되는 정보 관련).

*정황 다이어그램*은 정황 모델을 표현하기 위한 그래픽 모델링 언어로 사용된다. 정황 다이어그램에 대한 표준화된 표기법은 없다. 구조적 분석의 정황 다이어그램([DeMa1978]) 또는 맞춤형 박스 및 라인 다이어그램([Glin2019])을 사용해 정황 모델을 표현할 수 있다.

모델링 언어 UML [OMG2017] 에서 *사용 사례 다이어그램*은 시스템의 유스케이스와 이러한 유스케이스를 통해 시스템과 상호작용하는 시스템 정황의 액터 측면에서 시스템과 그 정황를 모델링하는 수단을 제공한다.

*유스케이스*는 시스템 정황의 액터와 액터의 관점에서 시스템 간의 동적 상호작용을 모델링한다. 유스케이스는 대부분 자연어로 된 양식 템플릿(3.3)을 사용하거나 UML 활동 다이어그램(3.4.4)을 사용해 작성된다.

3.4.3 구조 및 데이터 모델링 (L3)

구조 및 데이터 측면에 중점을 둔 모델은 시스템 또는 도메인의 정적 구조적 속성에 대한 요구사항을 지정한다.

정적 도메인 모델은 관심있는 도메인에서 (비즈니스) 대상과 그 관계를 지정한다. UML 클래스 다이어그램으로 표현할 수 있다 [OMG2017].

*클래스 모델*은 주로 시스템의 클래스와 그 속성 및 관계를 지정한다. 클래스는 시스템이 작업을 수행하기 위해 알아야 하는 현실 세계의 유형 및 무형 주체들을 나타낸다. UML 클래스 다이어그램은 일반적으로 클래스 모델을 위한 모델링 언어로 사용된다.

3.4.4 기능 및 흐름 모델링(L3)

기능 및 흐름 측면에 초점을 맞춘 모델은 주어진 입력에서 필요한 결과를 생성하는 데 필요한 일련의 작업 또는 (비즈니스) 프로세스를 실행하는 데 필요한 작업에 대한 요구사항을 지정하며, 여기에는 작업 간의 제어 및 데이터 흐름과 각 작업에 대한 책임자가 포함된다.

*액티비티 모델*은 시스템 기능을 지정하는 데 사용된다. 모델링 언어 UML [OMG2017] 에서 *액티비티 다이어그램*은 액티비티 모델을 표현하는 데 사용된다. 이들은 액션과 액션 간의 제어 흐름을 모델링하기 위한 요소를 제공한다. 액티비티 다이어그램은 누가 어떤 작업을 담당하고 있는지 표현할 수도 있다. 고급 모델링 요소(CPRE 파운데이션 레벨에서는 다루지 않음)는 데이터 흐름을 모델링하기 위한 수단을 제공한다.

*프로세스 모델*은 비즈니스 프로세스 또는 기술 프로세스를 설명하는 데 사용된다. UML 액티비티 다이어그램 또는 BPMN [OMG2013] 과 같은 특정 프로세스 모델링 언어를 사용하여 표현할 수 있다. CPRE 파운데이션 레벨에서는 프로세스 모델링에 UML 액티비티 다이어그램만 사용한다.

3.4.5 상태 및 동작 모델링(L2)

상태 및 동작에 중점을 둔 모델은 이벤트에 대한 상태 의존적 반응 또는 구성 요소 상호작용의 역학 측면에서 시스템 또는 도메인 구성 요소의 동작에 대한 요구사항을 지정한다.

*상태 머신*은 한 상태에서 다른 상태로 전환을 일으키는 이벤트와 상태 전환이 발생할 때 수행해야 하는 작업을 모델링한다. *상태차트* [Hare1988] 는 계층적 또는 직교적으로 분해된 상태를 가진 상태 머신다. 상태차트를 포함한 상태 머신은 *상태 머신 다이어그램* (*상태 다이어그램*이라고도 함)을 사용해 모델링 언어 UML [OMG2017] 로 표현할 수 있다.

3.4.6 요구공학의 추가 모델 유형 (L1)

CPRE 파운데이션 레벨에서 모델의 이해와 적용은 선택된 중요한 모델 유형으로 제한된다. 요구공학에서 사용하는 추가 모델 유형들이 있다. CPRE 파운데이션 레벨에서는 이들과 그 용도를 아는 것만으로도 충분하다.

*목표 모델*은 설정된 목표, 하위 목표 및 이들 간의 관계를 나타낸다. 목표 모델에는 목표 달성에 필요한 작업 및 리소스, 목표 달성을 원하는 행위자, 목표 달성을 방해하는 장애물도 포함될 수 있다.

SysML [OMG2019] 에서는 시스템과 액터에 대해 정형화된 블록을 사용해 컨택스트 다이어그램을 표현하도록 *블록 정의 다이어그램*을 조정할 수 있다. 블록 정의 다이어그램은 시스템의 개념 주체와 주체 간의 관계 측면에서 시스템의 구조를 모델링하는 데에도 사용할 수 있다.

*도메인 스토리 모델*은 일반적으로 도메인별 기호 [HoSch2020] 를 사용해 액터가 도메인의 장치, 가공물 및 기타 항목과 상호작용하는 방식에 대한 시각적 스토리를 지정해 기능 및 흐름을 모델링하는 데 사용할 수 있다. 이는 시스템이 작동할 애플리케이션 도메인을 이해하기 위한 수단이다.

*인터랙션 모델*은 객체 또는 액터 간의 동적 상호작용을 모델링한다. UML *시퀀스 다이어그램*은 개체 간의 상호작용을 지정하는 데 널리 사용되는 수단이다.

3.5 용어집 (L2)

한 명 이상이 참여하는 모든 요구공학 작업에는 용어에 대한 공유된 이해 부족으로 일부 사람들이 같은 용어를 다른 방식으로 해석할 위험이 있다. 이러한 문제 완화를 위해 관련 용어에 대한 공통화된 이해를 용어집에 기록한다.

*용어집*은 문맥별 용어, 주어진 문맥에서 특별한 의미가 있는 일상용어, 약어 및 약어에 대한 정의의 핵심 모음이다. 동의어(같은 것을 나타내는 다른 용어)는 그렇게 동의어라고 표시해야 한다. 동음이의어(같은 용어를 다른 용어로 사용하는 것)는 피하거나 동음이의어라고 표시해야 한다.

용어집에는 다음과 같은 규칙이 적용된다:

- 용어집을 중앙에서 관리한다.
- 시스템 개발의 전체 과정동안 용어집을 유지 관리한다.
- 용어집을 담당할 사람 또는 소규모 그룹을 지정한다.
- 용어집에 일관된 스타일과 구조를 사용한다.
- 이해관계자를 참여시키고 용어에 대한 동의를 구한다.
- 관련된 모든 사람이 용어집을 사용할 수 있게 한다.
- 용어집 사용을 필수로 설정한다.
- 작업 산출물에 용어집이 올바르게 사용되었는지 확인한다.

3.6 요구사항 문서 및 문서 구조 (L2)

요구사항 명세서(3.1.1)는 여러 요구공학 작업 산출물로 구성된다. 따라서 일관되고 유지 관리 가능한 요구사항 모음을 만들기 위해서는 이러한 문서를 잘 정의된 구조로 구성하는 것이 중요한다.
요구사항 문서에는 요구사항 외에도 용어집, 인수 조건, 프로젝트 정보 또는 기술 구현의 특성과 같은 추가 정보와 설명이 포함될 수 있다 .

요구사항은 기존 문서가 아닌 다른 문서 구조로 구성될 수도 있다.

자주 사용되는 문서는 다음과 같다:

- 이해관계자 요구사항 명세서
- 사용자 요구사항 명세서 (이해관계자 요구사항 명세서의 하위 집합으로, 사용자의 요구사항만 포함)

- 시스템 요구사항 명세서
- 비즈니스 요구사항 명세서
- 비전 문서

자주 사용되는 대체 문서 구조는 다음과 같다:

- 제품 백로그
- 스프린트 백로그
- 스토리 맵

문서 구조의 선택과 선택한 구조의 내부 조직은 다음에 따라 달라진다:

- 선택한 개발 프로세스 (5)
- 개발 유형 및 도메인
- 계약서 (고객이 특정 문서 구조의 사용을 지정할 수 있음)
- 문서의 크기

문서 템플릿은 요구사항 명세서를 구성하는 데 도움이 될 수 있다. 템플릿은 문헌 [Vole2020], [RoRo2012] 및 표준 [ISO29148] 에서 확인할 수 있다. 템플릿은 이전 유사한 프로젝트에서 재사용하거나 고객이 지정한 것일 수도 있다. 조직에서 내부 표준으로 템플릿을 만들 수도 있다.

3.7 요구공학의 프로토타입 (L1)

요구공학에서 *프로토타입*은 예제를 통해 요구사항을 지정하고 요구사항을 검증하는 수단이다. 특히 관련 이해관계자가 자연어 기반, 템플릿 기반 또는 모델 기반 작업 결과물을 작성하고 검토하고 싶지 않은 경우 프로토타입을 사용할 수 있다.

탐색적 프로토타입 ([LiSZ1994])은 이해를 공유하고, 요구사항을 명확히 하거나, 다양한 수준의 요구사항을 검증하는 데 사용된다. 이들은 사용 후에는 폐기된다.

- *와이어프레임*은 주로 디자인 아이디어와 사용자 인터페이스 개념을 논의하고 검증하는 데 사용되는 간단한 재료 또는 스케치 도구로 제작된 정확도가 낮은 프로토타입다.
- *목업(Mock-ups)은* 중간 정도의 충실도를 갖춘 프로토타입이다. 디지털 시스템을 지정할 때 실제 화면과 클릭 흐름을 사용하지만 실제 기능은 없다. 주로 사용자 인터페이스를 지정하고 검증하는 역할을 한다.

- *네이티브 프로토타입*은 이해관계자가 프로토타입을 사용해 시스템의 프로토타입 부분이 예상대로 작동하고 움직이는지 확인할 수 있을 정도로 시스템의 중요한 부분을 구현하는 정확도가 높은 프로토타입이다.

정확도에 따라 탐색용 프로토타입은 비용이 많이 드는 작업 결과물이 될 수 있으므로 비용과 얻는 가치 사이에는 항상 절충점이 있다.

진화형 프로토타입([LiSZ1994])은 개발할 시스템의 핵심을 구성하는 파일럿 시스템이다. 최종 시스템은 여러 번의 반복을 통해 파일럿 시스템을 점진적으로 확장하고 개선하면서 진화한다.

3.8 작업 산출물 및 요구사항에 대한 품질 기준 (L1)

요구사항이 좋은 요구사항으로 여겨지려면 특정 품질 기준을 충족해야 한다. 가치 지향적 접근 방식을 사용하는 최신 요구공학(원칙 1, 2 참조)에서 품질 기준의 충족 정도는 이 요구사항에 의해 생성된 가치와 일치해야 한다. 즉, 요구사항이 모든 품질 기준을 완벽하게 준수할 필요는 없지만, 요구사항의 가치가 높을수록 품질 기준의 관련성이 높아져 실패 위험을 줄일 수 있다.

적절성과 이해가능성은 단일 요구사항에 대한 가장 중요한 품질 기준이다. 이러한 요건이 없으면 다른 모든 기준의 충족 여부와 관계없이 해당 요건은 쓸모가 없거나 심지어 해로울 수 있다.

*단일 요구사항*에 대한 품질 기준:

- 적합성 (사실적이고 합의된 이해관계자의 니즈를 설명)
- 필수성
- 명확성
- 완료성 (독립성)
- 이해가능성
- 검증가능성

3.1.1 에 설명된 대로 요구사항은 일반적으로 단일 또는 여러 요구사항을 포괄하는 다양한 작업 산출물에 기록된다. 위의 품질 기준은 작업 산출물에서 고품질의 단일 요구사항을 생성하는 데 사용된다. 두 가지 이상의 요구사항을 충족하는 작업 산출물의 경우 다음의 품질 기준을 추가로 고려해야 한다.

*다양한 요구사항*을 포괄하는 작업 산출물에 대한 품질 기준:

- 일관성
- 비중복성
- 완료성 (알려진 요구사항 및 관련 요구사항이 누락되지 않음)
- 수정가능성
- 추적가능성
- 준수성

4 요구사항 정교화 프랙티스 (L3)

목표: 요구사항 출처를 식별하고, 요구사항을 도출하고, 충돌을 식별 및 해결하고, 요구사항을 검증하기 위한 프랙티스의 사용을 이해한다

소요시간: 4시간 30분

용어: 요구사항 출처, 시스템 경계, 시스템 정황, 요구사항 도출, 요구사항 유효성 검사, 이해관계자, 카노 모델, 충돌 해결

학습 목표

EO 4.1.1 관련 요구사항에 초점을 맞추기 위해 시스템 경계를 결정한다 (L3)
EO 4.1.2 구현할 시스템에 대한 관련 출처를 기억한다 (L1)
EO 4.1.3 이해관계자를 식별하고 이해관계자 목록을 작성한다 (L3)
EO 4.1.4 이해관계자 관리의 이점을 이해한다 (L2)
EO 4.2.1 카노(Kano) 모델이 올바른 요구사항을 도출하는 데 어떻게 도움이 되는지 이해한다 (L2)
EO 4.2.2 수집 기법과 디자인 및 아이디어 도출 기법의 차이점을 이해한다 (L2)
EO 4.2.3 주어진 상황에 적절한 유도 기법 선택 방법을 이해한다 (L2)
EO 4.3.1 다양한 갈등 유형을 기억한다 (L1)
EO 4.3.2 갈등 해결을 위해 필요한 활동을 이해한다 (L2)
EO 4.3.3 적절한 갈등 해결 기법 적용 방법을 이해한다 (L2)
EO 4.4.1 요구사항 문서를 검증해야 하는 이유를 이해한다 (L2)
EO 4.4.2 요구사항 검증을 위한 4 가지 중요한 측면을 기억한다 (L1)
EO 4.4.3 요구사항 검증을 위해 적절한 기술을 적용하는 방법을 이해한다 (L2)

4.1 요구사항 출처 (L3)

요구사항의 품질과 완성도는 관련된 요구사항 출처에 따라 크게 달라진다. 관련 출처가 누락되면 요구사항을 불완전하게 이해하거나 불완전한 요구사항으로 이어질 수 있다. 요구사항 출처를 식별하는 것은 지속적인 재검토가 필요한 반복적이고 순환적인 프로세스다.

개발할 시스템의 정황에 대한 공통 이해(2 의 원칙 3)는 관련 요구사항 출처를 식별할 수 있는 전제 조건이다. 시스템 경계와 정황 경계 사이의 영역을 (시스템) 정황이라고 한다 (2 에서원칙 4). (시스템) 정황은 개발할 요구사항의 특성을 이해하고 요구사항의 원본 출처를 식별하는 데 필요하다.

요구사항 출처는 다음 3 가지 유형으로 분류된다:

- 이해관계자
- 문서
- 시스템

시스템의 이해관계자(의미는 [Glin2020] 참조, 2 의 원칙 2 참조)가 요구사항의 주요 출처이다. 일반적인 이해관계자 역할은 다음과 같다. [BiSp2003]:

- 사용자 (최종 사용자라고도 함)
- 스폰서
- 관리자
- 개발자
- 당국
- 고객

또한 시스템의 *영향*을 받는 사람이나 조직은 (간접) 이해관계자로 간주해야 한다.

개발 벤처를 시작할 때 이해관계자를 체계적으로 파악하고 그 결과를 개발 전반에 걸쳐 지속적인 활동으로 관리해야 한다. 여기에는 이해관계자의 역할과 해당 역할에 속한 사람 모두를 식별하는 것이 포함된다.

사용자 인터페이스가 있는 모든 시스템의 경우, 시스템의 *최종 사용자*는 요구사항 엔지니어가 특히 관심을 갖는 이해관계자 그룹이다. 최종 사용자는 그룹으로 집계되어야 한다 (예: 유사한 역할, 업무 또는 책임에 따라).

최종 사용자를 개별적으로 식별할 수 있는 경우, 각 그룹에서 대표자를 선정해야 한다. 아니면 관련 최종 사용자 그룹을 대표하는 페르소나를 정의할 수 있다 [Coop2004].

관련 이해관계자 및 이해관계자 역할을 식별할 수 있는 잠재적 출처는 다음과 같다:

- 일반적인 이해관계자 그룹 및 역할 체크리스트
- 조직 구조
- 비즈니스 프로세스 문서
- 시장 보고서
- *추가* 이해관계자 식별을 위한 초기 이해관계자

이해관계자는 최소한 다음 정보가 포함된 최신 이해관계자 목록에 문서화해야 한다:

- 이름
- 기능 (역할)
- 추가 개인 및 연락처 데이터
- 프로젝트 진행 중 시간 및 공간적 가용성
- 관련성

- 전문 분야 및 범위
- 프로젝트의 목표와 관심사

이해관계자의 권리와 의무가 명확하지 않거나 이해관계자의 요구사항이 충분히 반영되지 않은 경우 이해관계자와의 문제가 발생할 수 있다. 이해관계자 관계 관리([Bour2009])는 이해관계자와의 문제에 효과적으로 대응할 수 있는 방법이다.

대부분의 시스템 환경에서는 더 많은 출처를 사용할 수 있다. 또한 대부분의 이해관계자가 '무의식적인' 요구사항에 대해 말하지 않기 때문에 성공적인 새 시스템을 위해서는 이러한 요구사항도 고려해야 한다 (4.2).

요구사항에 대한 추가 출처는 다음과 같다:

- 기존 및 레거시 시스템
- 프로세스 문서
- 법률 또는 규제 문서
- 회사별 규정
- 잠재적 미래 사용자에 대한 (마케팅) 정보

완전히 다른 도메인의 유사한 상황을 살펴보면 요구사항에 대한 또 다른 소스를 찾을 수 있다.

4.2 요구사항 도출 (L2)

도출 과정에서 요구사항 엔지니어의 임무는 이해관계자의 요구와 필요를 이해하는 동시에 적절한 기술을 적용하여 모든 관련 요구사항 출처로부터 요구사항을 수집하고 이를 도출하는 것이다. 도출의 핵심은 암묵적인 요구, 소망, 기대를 명시적인 요구사항으로 바꾸는 것이다.

요구사항을 도출하려면 각 요구사항의 성격과 중요성을 파악하는 것이 중요한다. 이는 프로젝트마다 그리고 시간이 지남에 따라 변경될 수 있다. 카노 모델([KaeA1984])은 요구사항을 3 가지 관련 범주로 분류한다:

- 감동자 (동의어: 흥분 요인, 무의식적 요구사항)
- 만족자 (동의어: 성능 요소, 의식적 요구사항)
- 불만족자 (동의어: 기본 요인, 무의식적 요구사항)

이러한 범주의 요구사항을 도출하는 데는 여러 가지 기법이 있다. 다음과 같이 분류한다:

- 수집 기법
- 디자인 및 아이디어 창출 기법

*수집 기법*은 요구사항 도출을 위해 만들어진 기법으로 [BaCC2015] 다양한 출처를 조사해 만족자와 불만족자를 도출하는 데 도움이 된다.

크게 4 가지 범주로 구분할 수 있다:

- 질문 기술
- 협업 기술
- 관찰 기술
- 가공물 기반 기술

*디자인 및 아이디어 생성 기법*은 요구사항을 도출하는 동안 창의성을 자극하기 위한 것이다. 문제 해결을 위한 아이디어를 창출하고 디자인 아이디어를 탐색하는 것을 목표로 한다 [Kuma2013]. 이는 종종 새롭거나 혁신적인 요구사항으로 이어질 수 있다. 이러한 기법의 대표적인 예로는 브레인스토밍 [Osbo1979], 유추, 프로토타이핑(예: Mock-ups), 시나리오 및 스토리보드가 있다.

디자인 및 아이디어 창출과 관련된 더 넓은 개념은 *디자인 사고*이다. 요구사항 도출에 사용할 수 있는 다양한 기법을 제공하는 *d.school* [Sdsc2012] 및 *Designing for Growth* [LiOg2011] 등의 다양한 접근 방식이 존재한다.

도출 기법은 기능 및 품질 요구사항, 제약 조건 등 모든 종류의 요구사항을 감지할 수 있어야 한다. 실제로 품질 요구사항과 제약 조건은 종종 주목을 덜 받는다.

*품질 요구사항*을 도출하려면 ISO 25010 표준([ISO25010])과 같은 품질 모델을 체크리스트로 사용해야 한다. 이 모델은 요구사항을 정량화하는 데에도 유용할 수 있다.

잠재적 솔루션 공간에서 발생할 수 있는 제약을 고려하면 *제약 조건*을 찾을 수 있다. 기술적, 법적, 조직적, 문화적, 환경적 문제 등이 그것이다.

올바른 도출 기법을 선택하는 것은 다음과 같은 다양한 요인에 따라 달라지는 중요한 핵심 역량이다:

- 시스템 유형
- 소프트웨어 개발 수명주기 모델
- 관련자들
- 조직 설정

일반적으로 다양한 유도 기법을 조합해 최상의 결과를 얻을 수 있다. [CaDJ2014] 는 기술을 선택하기 위한 체계적인 접근 방식을 나타낸다.

4.3 요구사항 관련 충돌 해결 (L2)

도출 기법 자체만으로는 결과물인 요구사항 세트의 전체적인 일관성, 완전성, 적합성 등을 보장할 수 없다. (3.8). 그러나 최종적으로 모든 이해관계자가 자신과 관련된 모든 요구사항을 이해하고 동의해야 한다. 일부 이해관계자가 동의하지 않는 경우, 이 상황은 그에 따라 해결해야 하는 갈등으로 인식해야 한다. 갈등의 유형과 정황적인 정보에 따라 적절한 갈등 해결 기법을 선택해야 한다. 이를 위해서는 요구사항 갈등의 성격과 관련된 이해관계자의 태도에 대한 심층적인 이해가 필요하다.

갈등을 식별하고 해결하기 위한 작업은 다음과 같다:

- 갈등 식별
- 갈등 분석
- 갈등 해결
- 갈등 해결 문서화 (결정된 사항)

서로 다른 갈등 유형을 구분하는 것이 유용하다 [Moor2014]. 다음 유형의 갈등은 종종 요구사항 엔지니어의 주의를 요한다:

- 주제 충돌
- 데이터 충돌
- 이해 상충
- 가치 충돌
- 관계 갈등
- 구조적 충돌

충돌을 성공적으로 해결하기 위해 일반적인 기술을 적용할 수 있다:

- 동의
- 타협
- 투표
- 무효화
- 변형의 정의

또한, 다음과 같은 몇 가지 보조 기법이 있다:

- 모든 사실 고려
- 플러스 마이너스 흥미
- 의사결정 매트릭스

4.4 요구사항 검증 (L2)

요구사항을 검증하는 것은 성공적인 시스템을 위한 중요한 단계이다 (원칙 6 에서 2). 요구사항 품질을 미리 확보하면 이후 낭비되는 노력을 줄일 수 있다. 요구사항 검증은 현재 작업 중인 작업 산출물과 그 안에 포함된 개별 요구사항의 품질을 확인한다는 것을 의미한다 (자세한 내용은 3.8 참조).

요구사항 검증에서 고려해야 할 중요한 측면은 다음과 같다:

- 올바른 이해관계자 참여
- 결함 식별과 수정의 분리
- 다양한 관점에서의 유효성 검사
- 반복적인 유효성 검사

다음과 같은 유효성 검사를 위한 몇 가지 기술이 있다 (예: [GiGr1993], [OleA2018]). 이러한 유효성 검사 기법은 종종 다음과 같이 분류된다:

- *리뷰 기법*의 예:
  - 워크스루
  - 인스팩션
- *탐색 기법*의 예:
  - 프로토타이핑
  - 알파 테스트 및 베타 테스트
  - A/B 테스팅 [KoTh2017]
  - 최소 실행 가능한 제품(MVP) 구축
- *샘플 개발*

이러한 기술은 형식과 들어가는 노력 면에서 다르다. 어떤 기술을 선택할지는 소프트웨어 개발 수명주기 모델, 개발 프로세스의 성숙도, 시스템의 복잡성 및 위험 수준, 법적 또는 규제 요구사항 또는 감사 추적의 필요성 등의 요인에 따라 달라진다.

5 프로세스 및 작업 구조 (L3)

목표: 요구공학 프로세스의 개념을 설명하고 적절한 프로세스 구성을 적용한다

소요시간: 1시간 15분

용어: 프로세스, 요구공학 프로세스

학습 목표

EO 5.1.1 요구공학 프로세스에 영향을 미치는 중요한 요인을 파악한다 (L1)
EO 5.1.2 이러한 요소가 어떻게 그리고 왜 영향을 미치는지 이해한다 (L2)
EO 5.2.1 요구공학 프로세스 구성 시 고려해야 할 측면을 이해한다 (L2)
EO 5.3.1 일반적인 요구공학 프로세스 구성을 파악한다 L1)
EO 5.3.2 요구공학 프로세스 구성 단계를 이해한다 (L2)
EO 5.3.3 간단한 시스템 및 개발 설정의 경우, 적절한 요구공학 프로세스 구성을 선택해 적용한다 (L3)

주어진 맥락에서 수행해야 할 요구공학 작업을 구체화하고 구조화하기 위해서는 프로세스가 필요하다. 만능 요구공학 프로세스는 없으므로(1.4), 주어진 개발 및 시스템 상황에 맞는 맞춤형 요구공학 프로세스를 구성해야 한다.

요구공학 프로세스는 다양한 참여자(예: 고객, 사용자, 요구사항 엔지니어, 개발자, 테스터) 간의 정보 흐름과 커뮤니케이션 모델을 형성하고 사용 또는 생산할 작업 산출물을 정의한다. 따라서 요구공학 프로세스는 요구사항을 도출, 문서화, 검증 및 관리하기 위한 프레임워크를 제공한다.

5.1 영향 요인 (L2)

요구공학 프로세스의 구성에는 여러 가지 요인이 영향을 미친다. 주요 요인은 다음과 같다:

- 전체 프로세스 적합성: 요구공학 프로세스는 전체 시스템 개발 프로세스에 적합해야 한다.
- 개발 정황
- 이해관계자의 역량 및 가용성
- 공통된 이해
- 개발할 시스템의 복잡성과 중요도
- 제약 조건
- 사용 가능한 시간 및 예산
- 요구사항의 변동성
- 요구사항 엔지니어의 경험

영향 요인에 대한 분석은 요구공학 프로세스를 구성하는 방법에 대한 정보를 제공한다. 영향 요인은 또한 가능한 프로세스 구성의 공간도 제한한다. 예를 들어, 이해관계자가 프로젝트 시작 단계에만 참여하는 경우 지속적인 이해관계자 피드백을 기반으로 하는 프로세스를 선택할 수 없다.

5.2 요구공학 프로세스 측면 (L2)

요구공학 프로세스를 구성할 때 고려해야 할 3 가지 결정적인 측면이 있다 [Glin2019].

시간 측면: 순차 vs. 반복

순차 프로세스에서는 요구사항이 프로세스의 단일 단계에서 미리 지정된다. 반복 프로세스에서는 일반적인 목표와 일부 초기 요구사항으로 시작하여 매 반복마다 요구사항을 추가하거나 수정하는 방식으로 요구사항이 점진적으로 지정된다.

순차 요구공학 프로세스 선택 기준:

- 시스템의 개발 프로세스는 계획에 따라 진행되며 대부분 순차적이다.
- 이해관계자는 자신의 요구사항을 알고 있으며 이를 미리 지정할 수 있다.
- 시스템의 설계 및 구현을 아웃소싱하기 위해서는 포괄적인 요구사항 명세가 계약의 기초가 되어야 한다.
- 규제 당국은 개발 초기 단계에 포괄적이고 공식적으로 공개된 요구사항 명세를 요구한다.

반복 요구공학 프로세스 선택 기준:

- 시스템의 개발 프로세스는 반복적이고 민첩하다.
- 대부분의 요구사항은 미리 알 수 없지만 시스템 개발 과정에서 나타나고 발전하게 된다.
- 이해관계자는 잘못된 시스템 개발의 위험을 완화하기 위한 수단으로 짧은 피드백 루프를 구축할 수 있다.
- 개발 기간에 따라 한두 번 이상의 반복이 가능하다.
- 요구사항을 쉽게 변경할 수 있는 기능이 중요하다.

목적 측면: 규범 vs. 탐색

규범적 요구공학 프로세스에서 요구사항 명세는 계약을 구성하며, 모든 요구사항은 구속력이 있으므로 반드시 구현되어야 한다. 탐색적 요구공학 프로세스에서는 목표만 선험적으로 알려져 있고 구체적인 요구사항은 탐색해야 한다.

규범 요구공학 프로세스 선택 기준:

- 고객은 시스템 개발을 위해 고정 계약을 요구한다.
- 기능과 범위가 비용과 마감일보다 우선한다.
- 특정 시스템의 개발은 입찰이나 아웃소싱을 할 수 있다.

탐색 요구공학 프로세스 선택 기준:

- 이해관계자는 처음에는 요구사항에 대해 막연한 생각만 가지고 있다.
- 이해관계자가 적극적으로 참여하고 지속적인 피드백을 제공한다.
- 마감일과 비용은 기능과 범위보다 우선한다.
- 실제로 어떤 요건을 어떤 순서로 이행해야 할지는 사전에 명확하지 않다.

타겟 측면: 고객 vs. 시장 측면

고객 맞춤형 요구공학 프로세스에서 시스템은 고객이 주문하고 공급업체가 개발한다. 시장 중심 요구공학 프로세스에서 시스템은 특정 사용자 분류를 타깃으로 하는 시장용 제품 또는 서비스로 개발된다.

고객 맞춤형 요구공학 프로세스 선택 기준:

- 이시스템을 주문하고 개발 비용을 지불한 조직이 주로 사용한다.
- 주요 이해관계자는 주로 고객의 조직과 관련이 있다.
- 이해관계자 역할에 따라 개인을 식별할 수 있다.
- 고객은 계약서 역할을 할 수 있는 요구사항 사양을 원한다.

시장 중심 요구공학 프로세스 선택 기준:

- 개발 조직은 특정 시장 부문에서 시스템을 제품이나 서비스로 판매하고자 한다.
- 잠재 사용자는 개별적으로 식별할 수 없다.
- 요구사항 엔지니어는 사용자를 대상으로 하는 예상 요구사항과 일치하도록 요구사항을 설계해야 한다.
- 제품 소유자, 마케팅 담당자, 디지털 디자이너, 시스템 설계자가 주요 이해관계자이다.

힌트 및 주의 사항

- 위에 제시된 기준은 고정된 규칙이 아닌 *휴리스틱*이다. 예를 들어, 고객과 공급업체 간의 계약은 일반적으로 포괄적인 요구사항 명세를 기반으로 하기 때문에 시스템 개발 아웃소싱은 탐색 요구공학 프로세스보다는 규범 요구공학 프로세스를 사용하는 것이 바람직하다. 그러나 탐색 요구공학 프로세스를 기반으로 아웃소싱 계약을 협상하는 것도 가능하다.

- 순차 요구공학 프로세스는 요구사항 변경에 대해 정교한 프로세스가 마련되어 있는 경우에만 작동한다.
- 순차 요구공학 프로세스는 긴 피드백 루프를 의미한다: 잘못된 시스템 개발에 대한 리스크를 줄이려면 요구사항을 집중적으로 검증해야 한다.
- 요구공학 프로세스를 정의할 때 *순차와 규범을* 함께 선택하는 경우가 많다.
- 탐색 요구공학 프로세스는 보통 반복적인 프로세스이기도 하다 (그 반대의 경우도 마찬가지).
- 시장 중심 프로세스에서 사용자의 피드백은 제품이 실제로 타겟 사용자 분류 요구를 충족시킬 수 있는지 검증할 수 있는 유일한 수단이다.
- 시장 중심 측면은 순차적이고 규범적인 측면과 잘 어울리지 않는다.

5.3 요구공학 프로세스 구성 (L3)

구체적인 시스템 개발 상황에서는 요구공학 담당자가 적용할 요구공학 프로세스를 구성해야 한다. 5.1 의 영향 요인 분석을 기반으로 5.2 에 설명된 프로세스 측면을 적절히 조합하여 [Glin2019] 을 사용할 수 있다. 아래에서 세 가지 일반적인 조합을 설명한다.

참여 요구공학 프로세스: 반복 & 탐색 & 고객 맞춤형 프로세스

주요 적용 사례:	공급업체와 고객이 긴밀하게 협업하고, 이해관계자가 요구공학와 개발 프로세스 모두에 적극적으로 참여한다.
일반적인 작업 산출물:	사용자 스토리 또는 작업 설명이 포함된 제품 백로그 및 프로토타입
일반적인 정보 흐름:	이해관계자, 제품 소유자, 요구사항 엔지니어 및 개발자 간의 지속적인 상호작용; 사용자의 피드백 포함 가능

계약 요구공학 프로세스: 일반적으로 순차(때때로 반복)& 규범 & 고객 맞춤형 프로세스

주요 적용 사례:	요구사항 명세는 요구사항 단계 이후 명세에는 관여하지 않으면서 이해관계자와의 상호작용이 거의 없는 사람들이 시스템을 개발하기 위한 계약적 근거가 된다.
일반적인 작업 산출물:	자연어 기반 요구사항과 모델로 구성된 전형적인 시스템 요구사항 명세
일반적인 정보 흐름:	주로 이해관계자에서 요구사항 엔지니어로 전달된다.

제품 중심 요구공학 프로세스: 반복 & 탐색 & 시장 중심

주요 적용 사례:	조직이 제품이나 서비스로 판매 또는 배포하기 위해 소프트웨어를 명시하고 개발한다.
일반적인 작업 산출물:	제품 백로그, 프로토타입
일반적인 정보 흐름:	제품 소유자, 마케팅, 요구사항 엔지니어, 디지털 디자이너, 개발자 그리고 고객/사용자의 빠른 피드백 간의 상호작용

앞서 언급한 구성 중 어느것도 적합하지 않은 시스템 및 개발 환경이 있을 수 있다는 점에 유의해야 한다. 예를 들어, 규제 제약으로 인해 ISO/IEC/IEEE 29148 [ISO29148] 과 같은 특정 표준을 준수하는 프로세스를 사용해야 할 수 있다.

요구공학 프로세스를 구성할 때는 5 단계 절차를 사용하는 것이 좋다:

1. 영향 요인 분석 (5.1)
2. 측면 기준 평가 (5.2)
3. 프로세스 구성 (5.3)
4. 작업 산출물 결정 (3)
5. 적절한 프랙티스 선택

6 요구사항 관리 관행 (L2)

목표: 요구사항 관리의 필요성과 이점을 이해한다.

소요시간: 2 시간

용어: 요구사항 관리, 변경 관리, 추적성, 요구사항 속성, 요구사항 수명주기, 우선순위 지정

학습 목표

EO 6.1.1 요구사항 관리가 무엇이며 왜 필요한지 파악 (L1)
EO 6.2.1 요구사항 작업 산출물에 상태/수명주기 모델이 필요한 이유를 설명한다 (L2)
EO 6.3.1 주어진 프로젝트 상황에서 요구사항 버전 관리 개념이 어떻게 적용되는지 설명한다 (L2)
EO 6.4.1 요구사항 구성 및 기준선 사용법을 숙지한다 (L1)
EO 6.5.1 요구사항에 속성의 목적을 파악한다 (L1)
EO 6.5.2 주어진 프로젝트 상황에서 요구사항에 대한 적절한 속성 집합이 어떤 모습인지 설명한다 (L2)
EO 6.5.3 보기의 목적을 설명하고 요구사항의 다양한 보기를 설명한다 (L2)
EO 6.6.1 요구사항 추적성의 근거를 확인한다 (L1)
EO 6.6.2 암묵적 추적성과 명시적 추적성의 차이점을 요약한다 (L1)
EO 6.6.3 명시적 추적성을 문서화할 수 있는 방법을 파악한다 (L1)
EO 6.7.1 순차(계획 중심) 및 애자일 접근법의 변경사항을 처리하는 방법을 파악한다 (L1)
EO 6.8.1 우선순위 지정 이유를 파악하고 의미있는 평가 기준을 숙지한다 (L1)
EO 6.8.2 요구사항 우선순위를 지정하는 단계를 설명한다 (L1)
EO 6.8.3 우선순위 지정 기법의 다양한 분류를 설명한다 (L1)

6.1 요구사항 관리란 무엇인가? (L1)

요구사항 관리는 다양한 작업 산출물에 기록된 기존 요구사항을 관리하는 프로세스이다. 특히 여기에는 요구사항 저장, 변경 및 추적이 포함된다 [Glin2020]. 요구사항 관리는 선택한 개발 프로세스와 정황에 따라 다양한 방식과 수준에서 이루어질 수 있다 (예: [Leff2011], [Rupp2014], [WiBe2013] 참조). 정황에 관계없이 요구사항 관리의 임무는 프로젝트의 모든 역할이 효과적이고 효율적으로 작업할 수 있도록 요구사항을 유지하는 것이다.

6.2 수명주기 관리 (L2)

수명주기 관리란 모든 작업 산출물의 수명주기 상태를 추적하는 프로세스를 말한다. 문서화된 각 요구사항과 각 작업 산출물에는 고유 수명주기가 있으며, 생성되고 난 후, 리뷰, 재작업, 통합, 동의 등의 과정을 거치기 전에 평가 및 정제된다. 어떤 작업 산출물이 어떤 상태인지 식별할 수 있게

하려면 허용된 각 수명주기 상태와 상태 전환을 정의하는 수명주기 모델이 필요하다. 작업 산출물의 실제 상태는 (일반적으로) 전환 이력을 포함해야 하여 항상 명확해야 한다.

6.3 버전 제어 (L2)

요구사항의 버전 관리는 모든 작업 산출물이 진화하는 동안 이를 추적하는 프로세스를 의미한다. 작업 산출물의 모든 변경 사항은 새 버전에 반영되어야 한다. 버전 관리를 통해 작업 산출물의 이력을 원본까지 추적하고 작업 산출물을 이전 버전으로 복원할 수 있다. 이를 위해 버전 관리에는 세 가지 조치가 필요하다:

- 작업 산출물의 버전을 고유하게 식별하는 버전 번호
- 변경된 사항 히스토리
- 작업 산출물 저장 컨셉

모든 작업 산출에 대해 버전 관리를 고려해야 한다 [WiBe2013]. 버전 번호는 일반적으로 버전과 증분의 최소 두 부분으로 구성된다.

6.4 구성 및 기준선 (L1)

요구사항 *구성*은 요구사항을 포함하는 일관된 작업 산출물 집합이다. 각 구성은 특정 용도에 맞게 정의되며 많아봐야 각 작업 산출물의 버전 하나를 포함한다 [Glin2020]. 예를 들어, 구성의 목적은 일련의 작업 결과물을 검토하거나 개발 노력을 쉽게 추정하기 위한 것이다.

*기준선*은 프로젝트의 릴리스 계획 또는 기타 배포 단계에 사용되는 안정적이면서 변경 제어가가 가능한 작업 산출물의 *구성*이다 [Glin2020].

구성에는 다음과 같은 속성이 있다:

1. 논리적 연결
2. 일관성
3. 고유성
4. 불변성
5. 재설정 기준

6.5 속성 및 보기 (L2)

*속성*은 작업 산출물의 중요한 메타데이터를 문서화하는 데 필요하며 일반적으로 프로젝트나 제품 수명주기 동안 여러 중요한 질문에 답하는 데 사용된다.

속성을 사용해 요구사항을 특성화하는 목적은 팀원 및 기타 이해관계자가 프로젝트의 어느 시점에서든 필요한 요구사항에 대한 정보를 얻을 수 있도록 하기위한 것이다.

관련 속성 집합을 정의하는 것은 프로젝트의 다양한 이해관계자가 필요로 하는 정보에 따라 달라진다. 기존 표준(예: [ISO29148])은 가장 관련성이 높은 속성에 대한 개요를 제공한다.

*보기*는 전체 요구사항 집합에서 발췌한 것으로, 현재 관심있는 콘텐츠만 들어 있다. 기술적 관점에서 볼 때 보기는 필터와 설정한 정렬의 조합으로, 선택한 조합을 저장해 다른 참가자가 사용하거나 재사용할 수 있다.

여기서는 세 가지 유형의 보기를 구분한다:

- *선택적 보기*
- *투영적 보기*
- *종합적 보기*

대부분의 경우 요구사항 보기는 보고서를 만들기 위한 선택적 보기, 투영적 보기 및 종합적 보기의 조합이다.

6.6 추적성 (L1)

추적성 [GoFi1994] 은 요구사항의 *출처*(예: 이해관계자, 문서, 근거 등) *로 거슬러* 올라가 *후속 작업 산출물(예: 테스트 케이스)*은 물론, 이 요구사항이 의존하는 *다른 요구사항까지* 추적할 수 있는 기능다.

추적성은 요구사항 관리의 전제 조건이며 표준, 법률 및 규정에서 명시적으로 요구하는 경우가 많다. 추적성을 구현한다는 것은 기본적으로 서로 다른 추상화 수준(3.1.2), 세부 수준(3.1.3)에서 서로 다른 작업 산출물(0) 간의 분석, 규정 준수 및 정보를 이유로 모든 관련 전임자 및 후임자에 대해 종속성을 유지을 의미한다.

추적성은 작업 결과물을 구조화 및 표준화하여 *암묵적*으로 문서화하거나, 다양한 형식의 고유 식별자를 기반으로 작업 결과물을 서로 연관시켜 *명시적*으로 문서화할 수 있다 [HuJD2011]. 일반적인 표현 형식은 하이퍼링크, 참조, 매트릭스, 표 또는 그래프이다.

6.7 변경 처리 (L1)

요구사항은 고정되어 있지 않다. 요구사항 변경은 여러 이유로 발생하며, 공식적으로 *변경 요청*을 작성하거나 *제품 백로그*에 새 항목을 추가하는 등 적절하게 처리해야 한다 (원칙 7 에서 2).

변경 사항의 의사 결정, 계획 및 구현 제어는 개발 접근법과 변경이 발생하는 시점에 따라 달라진다.

순차 접근법에서는 변경에 대한 결정을 변경 제어 위원회(프로젝트의 경우) 또는 변경 자문 위원회(운영의 경)에서 내리는 경우가 많다. 보다 *반복적인* 접근 방식에서는 제품 소유자가 제품 백로그에 변경 사항을 포함시키고 그에 따라 새 항목의 우선순위를 지정한다.

6.8 우선순위 지정 (L1)

모든 요구사항이 똑같이 중요한 것은 아니다 [Davi2005]. 평가 및 우선순위 지정은 다음 제품 릴리스나 증분과 가장 관련성이 높은 요구사항을 결정하는 데 사용된다.

요구사항의 *평가*는 우선순위를 정하는 기준이 되며, 비즈니스 가치, 긴급성, 노력, 종속성 등과 같은 여러 평가 기준을 사용해 결정되는 경우가 많다.

요구사항의 *우선순위*는 특정 기준에 따라 다른 요구사항과 비교 단일 요구사항의 중요성을 설명한다 [Glin2020]. *우선순위* 지정 자체는 단일 또는 여러 기준에 따라 수행되며, 이는 주로 선택한 우선순위 지정 기법에 따라 달라진다.

우선순위를 정하는 단계:

- 우선순위를 정하기 위한 주요 목표와 제약 조건 정의
- 원하는 평가 기준 정의
- 참여해야 하는 이해관계자 정의
- 우선순위를 정해야 하는 요구사항 정의
- 우선순위 지정 기법 선택
- 우선순위 지정 수행

*우선순위 지정 기법*은 다음과 같이 분류할 수 있다:

- *임시* 우선순위 지정 기법
- *분석* 우선순위 지정 기법

7 도구 지원 (L2)

목표: 요구공학 도구의 역할과 구현 측면에 대한 개요 제공한다.

소요시간: 30분

용어: 도구, 요구공학 도구

학습 목표

EO 7.1.1 다양한 유형의 요구공학 도구를 파악한 (L1)
EO 7.2.1 요구공학 도구를 도입할 때 고려해야 할 사항을 설명한다 (L2)

7.1 요구공학 도구 (L1)

요구공학 프로세스는 전용 작업 및 활동 지원 도구로 지원 가능하다. 요구공학 프로세스는 상당히 개별적인 특성을 가지므(5), 기존 요구공학 도구는 요구공학의 특정 측면에만 초점을 맞추며, 모든 활동을 지원하는 경우는 드물다. 요구사항 엔지니어는 도구를 선택하기 전에 요구공학 프로세스 중 어떤 작업과 활동을 어떻게 지원지 결정해야 한다. 지원하는 도구를 다음과 같이 구분한다:

- 요구사항 관리:
  - 요구사항 속성의 정의 및 저장
  - 요구사항 우선순위 지정
  - 버전 및 형상 관리
  - 요구사항 추적
  - 요구사항 변경 관리
- 요구공학 프로세스 관리:
  - 요구공학 프로세스 측정 및 보고
  - 제품 품질 측정 및 보고
  - 요구공학 작업흐름 관리
- 요구사항에 대한 지식 문서화:
  - 요구사항 공유
  - 요구사항에 대한 공통된 이해 형성
- 요구사항 모델링
- 요구공학에서의 협업
- 요구사항 테스트/시뮬레이션

도구는 위에서 언급한 기능이 혼합되어 제공되는 경우가 많다. 모든 요구공학 작업을 적절하게 처리하기 위해 서로 다른 도구를 결합할 수 있다.

때로는 오피스나 이슈 추적 도구와 같은 다른 종류의 도구를 사용해 요구사항을 문서화하거나 관리하기도 한다. 이러한 도구에는 한계가 있으므로 요구공학 프로세스가 통제되고 요구사항이 조정되어 매우 안정된 경우에만 사용해야 한다.

7.2 도구 도입 (L2)

요구공학 도구의 선택은 다른 용도의 도구를 선택하는 것과 비슷하다. 성공적인 선택을 위해서는 목표, 상황 및 요구사항이 미리 준비되어야 한다 [Fugg1993].

적절한 요구공학 절차와 기법을 도입한 다음 적절한 도구를 찾을 수 있다. 도구 도입에는 명확한 요구공학 책임과 절차가 필요하다. 요구공학 도구를 도입하는 과정에 다음과 같은 측면이 고려되어야 한다:

- 라이선스 비용 외 모든 수명주기 비용 고려
- 필요한 리소스 고려
- 리스크 방지를 위한 파일럿 프로젝트를 사용
- 정의된 기준에 따른 도구 평가
- 직원 대상 도구 사용법 교육

참조

[AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Goal Decomposition and Scenario Analysis in Business Process Reengineering. CAiSE (Conference on Advanced Information Systems Engineering), 1994, 94–104.

[BaCC2015] K. Baxter, C. Courage, K. Caine: Understanding Your Users: A Practical Guide to User Research Methods, 2nd edition. Morgan Kaufmann, Burlington, 2015.

[BiSp2003] K. Bittner, I. Spence: Use Case Modelling. Pearson Education, Boston, 2003.

[Bour2009] L. Bourne: Stakeholder Relationship Management: A Maturity Model for Organisational Implementation. Gower Publishing Ltd, Birlington, 2009.

[CaDJ2014] D. Carrizo, O. Dieste, N. Juristo: Systematizing requirements elicitation technique selection. Information and Software Technology 2014, 56(6): 644–669.

[Cock2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, Boston 2001.

[Cohn2004] M. Cohn: User Stories Applied – For Agile Software Development. Addison-Wesley, Boston, 2004.

[Coop2004] A. Cooper: The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity. Que, Indianapolis, 2004.

[Davi2005] A. M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. Dorset House Publishing, New York, 2005.

[Davi1993] A. M. Davis: Software Requirements – Objects, Functions, & States, 2nd edition, Prentice Hall, New Jersey, 1993.

[DeMa1978] T. DeMarco: Structured Analysis and System Specification. Yourdon Press, New York, 1978.

[Fugg1993] A. Fuggetta: A classification of CASE technology. IEEE Computer 1993, 26 (12): 25–38.

[GiGr1993] T. Gilb, D. Graham: Software Inspection. Addison Wesley, Boston, 1993.

[Glin2019] M. Glinz: Requirements Engineering I. Course Notes, University of Zurich, 2019. https://www.ifi.uzh.ch/en/rerg/courses/archives/hs19/re-i.html#resources. 2020 년 7 월에 마지막으로 방문했습니다.

[Glin2020] M. Glinz: A Glossary of Requirements Engineering Terminology. Version 2.0. https://www.ireb.org/en/downloads/#cpre-glossary. 2020 년 7 월에 마지막으로 방문했습니다.

[GoFi1994] O. Gotel, A. Finkelstein: An Analysis of the Requirements Traceability Problem. 1st International Conference on Requirements Engineering, Colorado Springs, 1994. 94–101.

[GoRu2003] R. Goetz, C. Rupp: Psychotherapy for System Requirements. 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), London, 2003. 75–80.

[[GRL2020] Goal oriented Requirement Language. University of Toronto, Canada https://www.cs.toronto.edu/km/GRL. 마지막 방문일 2020 년 5 월.

[Hare1988] D. Harel: On Visual Formalisms. Communications of the ACM 1988, 31 (5): 514–530.

[HoSch2020] S. Hofer, H. Schwentner: Domain Storytelling — A Collaborative Modeling Method. Available from Leanpub, http://leanpub.com/domainstorytelling. 2020 년 7 월에 마지막으로 방문했습니다.

[HuJD2011] E. Hull, K. Jackson, and J. Dick: Requirements Engineering. Springer, 3rd Ed, 2011.

[ISO29148] ISO/IEC/IEEE 29148: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering, International Organization for Standardization, 2018.

[ISO19650] ISO 19650: Organization and Digitization of Information about Buildings and Civil Engineering Works, including Building Information Modelling (BIM)– Information Management Using Building Information Modelling – Part 1 and 2, International Organization for Standardization, 2018.

[ISO25010] ISO/IEC/IEEE25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE) –– System and software quality models. International Organization for Standardization, Geneva, 2011.

[Jack1995] M. A. Jackson: Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. Addison-Wesley, New York, 1995.

[Jack1995b] M. Jackson: The World and the Machine. 17th International Conference on Software Engineering 1995 (ICSE 1995). 287–292.

[KaeA1984] N. Kano et al.: Attractive quality and must-be quality. Journal of the Japanese Society for Quality Control 1984, 14(2): 39–48. (in Japanese)

[KoTh2017] R. Kohavi, S. Thomke: The Surprising Power of Online Experiments – Getting the most out of A/B and other controlled tests. 하버드 비즈니스 리뷰, 2017 년 9~10 월: 74–82.

[Kuma2013] V. Kumar: 101 Design Methods – A Structured Approach for Driving Innovation in Your Organization. John Wiley & Sons, Hoboken, 2013.

[Laue2002] S. Lauesen: Software Requirements. Styles and Techniques. Addison-Wesley, Harlow, 2002.

[Leff2011] D. Leffingwell: Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, Boston, 2011.

[LiOg2011] J. Liedtka, T. Ogilvie: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia University Press, 2011.

[LiSZ1994] H. Lichter, M. Schneider-Hufschmidt, H. Zullighoven: Prototyping in Industrial Software Projects – Bridging the Gap Between Theory and Practice. IEEE Transactions on Software Engineering 1994, 20 (11): 825–832.

[MFeA2019] D. Méndez Fernández, X. Franch, N. Seyff, M. Felderer, M. Glinz, M. Kalinowski, A. Volgelsang, S. Wagner, S. Bühne, K. Lauenroth: Do We Preach What We Practice? Investigating the Practical Relevance of Requirements Engineering Syllabi – The IREB Case. CIbSE 2019: 476–487.

[Moor2014] C. W. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts, 4th edition. John Wiley & Sons, Hoboken, 2014.

[MWHN2009] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak: Easy Approach to Requirements Syntax (EARS). 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, Georgia, 2009. 317–322.

[OleA2018] K. Olsen et al.: Certified Tester, Foundation Level Syllabus – Version 2018. International Software Testing Qualifications Board, 2018.

[OMG2013] Object Management Group: Business Process Model and Notation (BPMN), version 2.0.2. OMG document formal/2013-12-09

http://www.omg.org/spec/BPMN. 2020 년 7 월에 마지막으로 방문했습니다.

[OMG2017] Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5.1. OMG document formal/2017-12-05.

https://www.omg.org/spec/UML/About-UML/. 2020 년 7 월에 마지막으로 방문했습니다.

[OMG2019] Object Management Group: OMG Systems Modeling Language (OMG SysML™), Version 1.6. OMG Document formal/19-11-01.

https://www.omg.org/spec/SysML/. 2022 년 1 월에 마지막으로 방문했습니다.

[Osbo1979] A. F. Osborn: Applied Imagination, 3rd revised edition. Charles Scribner´s Sons, New York, 1979.

[RoRo2012] S. Robertson and J. Robertson: Mastering the Requirements Process, 3rd edition. Addison-Wesley, Boston, 2012.

[Rupp2014] C. Rupp: Requirements-Engineering und Management, 6. Auflage. Hanser, München, 2014. (in German).

[Sdsc2012] Stanford d.school: An Introduction to Design Thinking. Hasso Plattner Institute of Design, Stanford, 2012. https://dschool-old.stanford.edu/groups/designresources/wiki/36873. 2020 년 7 월에 마지막으로 방문했습니다.

[vLam2009] Axel van Lamsweerde: Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009.

[Vole2020] Volere: Requirements Resources. https://www.volere.org. 2020 년 7 월에 마지막으로 방문했습니다.

[WiBe2013] K. Wiegers and J. Beatty: Software Requirements, 3rd edition. Microsoft Press, Redmond, 2013.