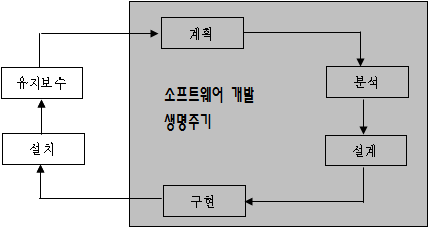
**Chapter 1**

**System analysis and design**



**Phases of SDLC :** planning -> analysis -> design -> implementation

계획 -> 분석 -> 설계 -> 구현

SDLC는 시스템을 설계하고, 구현하고, 사용자에게 전달 함으로써 어떻게 정보시스템이 Business needs를 지원할 지 이해하는 과정이다.

* **The system development life cycle (SDLC**) is the process of understanding how an information system can support a business needs by designing a system, building it, and delivering it to users.

시스템 개발 생명 주기에서 핵심 인물은 시스템 분석가이다.

* The key person in the system development life cycle is system analysist.

Code-and-Fix방법은 생명주기가 없다.

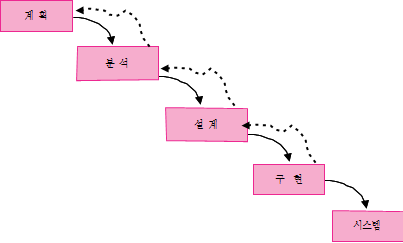
* Code-and-Fix method doesn’t have a life cycle.

**System Development Methodologies**

1. **Structured development**
   1. waterfall development (폭포수 모형)
   2. parallel development (병렬 개발 모형)
2. **Rapid application development (RAD)**
   1. phased development (단계적 모형)
   2. prototyping
   3. throwaway prototyping
3. **Agile development**

**3.1** extreme programming

**3.2** scrum



**Waterfall development (폭포수 모형)**

1. 각 단계에 중복이나 상호작용이 없음

* There is no duplication or interaction.

1. 각 단계는 다음 단계 시작 전에 끝나야 함.

* Each phase must be ended before the next phase

1. 개발 전 완전한 설계가 필요.

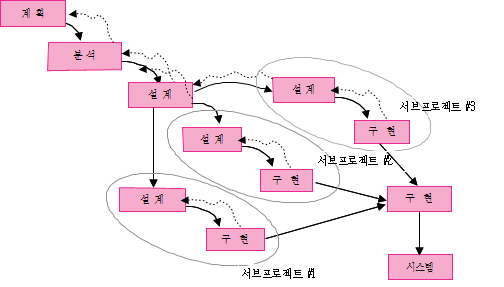
* It needs a complete design before the development.

1. 에러가 날 확률이 적다. (분석 전문가가 참여하기 때문)

* There is a little possibilities that an error is occurred. (because an analysist is participated.)

1. 한 번 다음 단계로 넘어가면 돌아올 수 없다.

* Once you go to next phase you never come back to previous phase.



**Parallel development (병렬 개발 모형)**

1. 분석과 시스템 납품 사이의 기간을 최소화 시킬 수 있음.

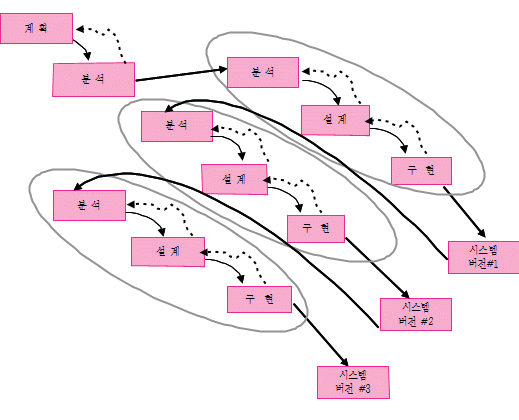
* It can reduce the time to deliver a system.

1. 각각의 프로젝트는 구현 단계에서 서로에게 영향을 줄 수 있음.

* Each projects are dependent to each other in the implementation phase.

1. 구현 마지막 단계에서 통합을 위한 노력이 큼.

* At the end of the project it can require significant integration efforts.



**Phased development (단계적 모형, EX: Microsoft)**

1. 유용한 시스템을 빠른 시간 내에 사용자에게 돌려줌.

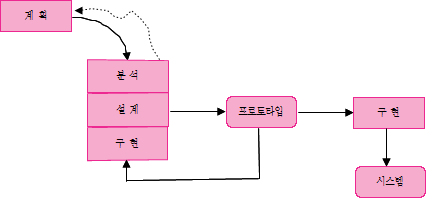
* It has the advantage of quickly getting a useful system into the hands of the users.

1. 중요한 요소들은 미리 초기의 버전에 포함된다.

* An important and fundamental requirements are bundled into the first version of the system.

1. 의도적인 미완성 시스템으로 작업한다.

* Users begin to work with systems that are intentionally incomplete.



**Spiral prototyping (Quick and Dirty)**

1. 사용자의 요구를 정확하게 추출 할 수 있다.

* It can extract users’ demand more precisely

1. 시스템이 완벽하지 않아도 사용자에게 제공이 가능하다.

* It very quickly provides a system with which the users can interact. Even if it is not ready for widespread organizational use at first.

1. 문제점이 있으면 기존의 모형을 폐지시켜야 함.

* Once a problem is founded, you must abandon current model and redesign it.

1. 개발 후에 치명적인 문제가 발견될 수 있다.

* There is a possibility that a critical problem is founded after the development.



**Throwaway prototyping**

1. 구현단계로 넘어갈 때, 프로토타입을 폐기한다.

* After designed, the design prototypes are thrown away.

1. Spiral prototyping 보다 더 안정적인 시스템을 개발한다.

* It usually produces more stable systems.

**Waterfall : 자동차 만든 후 실내등이 않켜지면**

**Parallel development : general design - > 여러 개의 design 구현**

**-🡪 한 곳에서의 설계 결정이 다른 곳에 영향을 준다. 프로젝트 끝에서 완성이 무지 힘들다.**

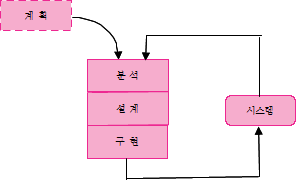
**Rad (프로토 타입 제공) – managing user expection**

**Rad1 phased development : 차수를 나누어 개발을 진행(시스템 버전1, 1.1, 1.2, …. 2.0, ….)**

**사용자가 미완성된 시스템을 이용 업무를 진행한다. 따라서 가장 중용한 기능을 먼저 구현해야 한다.**

**Rad2 prototyping : - 가장 기본 적인 문제가 나중에 발견될 수 있다. – 자동차를 만들고 10,000km뛰고 난 후 엔진오일을 갈려고 했더니 엔진 전체를 들어내어야 한다.**

**Rad 3 throwaway – 분석과 설계단계가 끝날 때 까지 설계 프로토타입을 만들고 계속반복해서 완성이 되면 버린다.**



**Agile development**

1. Agile 원칙은 객체지향 시스템을 개발할 때 매우 유용하다.

* Agile principles are very useful in object-oriented systems development.

1. Agile 정의만큼 문서의 양이 매우 적기 때문에 감사에서 문제가 발생할 수 있다.

* It may causes a problem regarding the auditability because of lack of documentations.

**시스템 분석가의 능력**

**@변화의 대리인**

**@조직을 향상시키기위한 확인**

**@다른 사람들을 훈련시키고 동기부여함**

**@요구되는 능력(skills needed):**

**@기술**

**@비즈니스: 비즈니스 프로세스를 알아야 함**

**@분석: 문제를 풀 수 있어야한다**

**@커뮤니케이션: 기술과 비기술의 청중(technical & non-technical audiences\_**

**@대인관계: 리더쉽과 management**

**@윤리: 잘 다루고 기밀 정보를 보호해야한다.**

**객체지향 시스템 분석과 설계(Analysis & Design)**

**@데이터와 프로세스간 균형을 이루어야한다**

**@UML과 Unifiend Process를 사용한다**

**@OOAD(object-oriented systems analysis&design)의 특성**

**@use-case Driven**

**@건축가의 중심**

**@반복 및 증가**

**객체지향 시스템의 특징**

**@클래스와 객체**

**@객체: 클래스의 인스턴스화(객체 자체)**

**@Attributes: 클래스를 설명하는 정보**

**@State: describes its values and relationships at a point in time**

**@methods(함수) 및 메시지**

**@Methods: the behavior of a class**

**@Messages: information sent to an object to trigger a method(procedure call)**

**@캡슐화와 정보 은닉**

**@캡슐화: 데이터와 프로세스의 combination**

**@정보 은닉: 기능 숨기기**

**@상속**

**@보통 클래스들은 만들어진다(superclasses)**

**@subclasses은 데이터 상속과 superclass로부터 methods를 상속한다**

**@다형성과 동적 바인딩(Polymorphism & dynamic binding)**

**@다형성: 같은 메시지는 다른 의미를 가진다.**

**@동적 바인딩: 객체의 타입은 실행시간까지 결정되지 않는다.(->동적 바인딩을 함으로써 상속이나 다형성이 가능)**

**@정적 바인딩 vs 동적 바인딩**

**@정적 바인딩: 컴파일 시간에 성격이 결정**

**@동적 바인딩: 실행 시간에 성격이 결정**

**객체지향 시스템의 분석과 설계**

**@Use-case driven**

**@use-case는 시스템의 동작을 정의**

**@각 use-case는 하나의 비즈니스 프로세스를 중점으로 둔다.**

**@아키텍쳐 중심**

**@외적인 기능 관점**

**@유저의 관점에 집중**

**@구조적인 static 관점**

**@attributes, methods, classes와 관계에 집중**

**@동작 관점**

**@클래스와 resulting behaviors 사이의 메시지에 집중**

**@반복과 증가**

**@지속적인 테스트 및 개선**

**@분석가는 시간이 지남에 따라 시스템을 더 잘 이해 해야한다.**

**OOSAD의 혜택**

**(object-oriented systems analysis & design)**

**@복잡한 시스템을 관리하기 쉬운 모듈로 작게 쪼갤 수 있다.**

**@개인적으로 모듈에 수행할 수 있다.**

**The Unified Process(통합 프로세스)**

**@객체 지향 분석 및 설계를 위해 다양한 UML 기술을 사용하는 시기와 방법을 설명하는 특정 방법론**

**@단계와 workflows로 구성되어 있는 2차원 프로세스**

**@단계는 개발 기간이다.**

**@Workflows는 각 단계에서 발생하는 업무이다.**

**@단계와 workflows의 활동은 겹칠 것이다.**

**Unified Process 단계**

**@처음**

**@타당성 분석 수행**

**@Workflows는 다양하나 비즈니스 모델링과 요구수집에 집중되어 있다.**

**@Elaboration**

**@분석과 설계에 집중**

**@다른 workflows가 포함될 수도 있다.**

**@Construction**

**@프로그래밍에 집중(이행)**

**@Transition-Focus on testing & 전개**

**Engineering Workflows**

**@비즈니스 모델링, 요구, 분석, 설계, 구현, 테스팅, 전개**

**Supporting Workflows**

**@프로젝트 관리, 구성과 조치 변화, 환경, Operations과 지원, 인프라 관리**

**Unified Process(통합 프로세스)의 확장**

**@Staffing, 예산관리, 계약 관리, 유지, Operations, 지원등을 포함하지 않는다.**

**@상품이 전개된 후 문제 해결을 위해 Production Phase를 추가한다.**

**@New Workflows:**

**@Operations & Support**

**@인프라 구조**

**@존재하는 workflows 수정**

**@workflow 테스트**

**@workflow 전개**

**@Environment workflow**

**@Project Management workflow**

**@Configuration & change management workflow**

**@UML(Unified Modeling Language)**

**@분석에서 구현에 이르기까지 모든 시스템 개발 프로젝트를 모델링 할 수 있을 정도로 풍부한 객체지향 용어 및 다이어그램 기법의 공통 어휘를 제공한다.**

**@UML Structure Diagrams와 UML Behavior Diagrams 보통 2가지가 있다.**

**UML Structure Diagrams**

**@정보 시스템 안 데이터와 정적 관계를 표현한다.**

**@클래스, 객체, 패키지, Deployment, Component, Composite structure**

**UML Behavior Diagrams**

**@비즈니스 정보 시스템을 나타내는 인스턴스 또는 객체간의 동적 관계를 나타낸다.**

**Extreme programming**

1. Communication (의사소통)

* The developers must provide rapid feedback to the end users.

1. Simplicity (단순함)

* Extremely programming (XP) requires to follow the KISS principles.

1. Feedback (피드백)
2. Courage (격려)

* Developers must have a quality-first mentality.

1. 분석과 문서화의 결여 때문에 큰 시스템을 개발하기 어렵다.

* Owing to the lack of analysis and documentation, developing large systems with XP is impossible

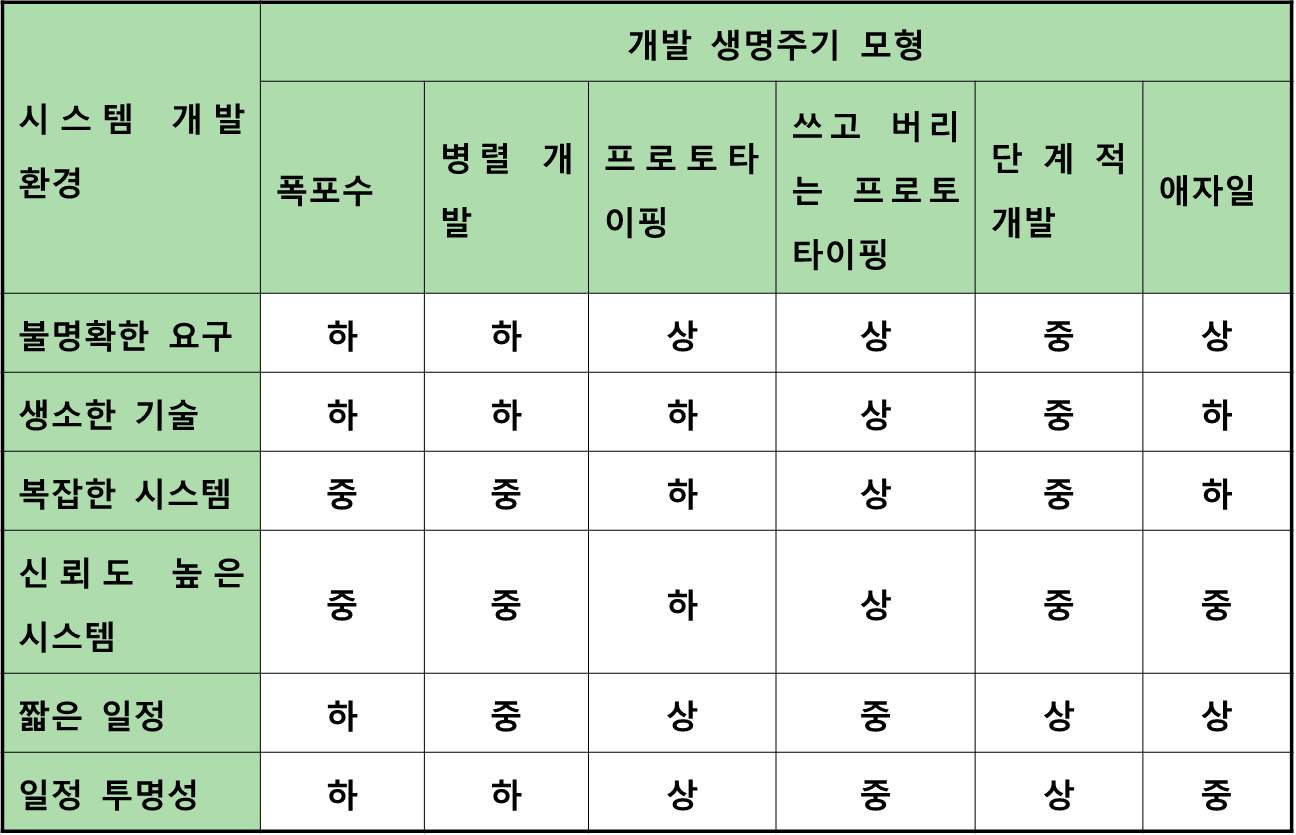
**Scrum**

1. Scrum 팀은 리더가 없다.

* Scrum teams don’t have a designated team leader.

1. 팀의 규모가 대체로 작기 때문에, 대규모 시스템을 개발하기 어렵다.

* It’s difficult to develop very large, mission-critical systems because of the small team.



1. **요약**

**@모든 시스템 개발 프로젝트는 본질적으로 SDLC(시스템 개발 생명 주기)라는 동일한 프로세스를 따른다.**

**@시스템 개발 방법론은 SDLC 구현에 대한 공식화된 접근 방식이다.**

**@시스템 분석가는 다양한 스킬이 필요하고 다양한 역할을 수행한다.**

**@객체지향 시스템은 전통적인 시스템과는 다르다**

**@OOSAD(Object-Oriented Systems and Analysis and Design)은 use-case 중신, 아키텍쳐 중심, 반복 및 증분 정보 시스템 개발 접근법을 사용한다.**

**@Unified Process는 일련의 단계 및 workflows로 설명 된 2차원 시스템 개발 프로세스이다.**

**@UML은 표준 다이어그램 기술 세트이다.**

**Chapter 2**

**Project management**

프로젝트 관리는 올바른 기능을 가지고 최소한의 비용으로 주어진 시간 안에 시스템 개발을 계획하고 관리하는 과정이다.

* Project management is the process of planning and controlling the development of a system within a specified time frame at a minimum cost with the right functionality.

SWOT 분석은 시스템 설계 전에 하는 것이 적절하다.

* SWOT analysis is adequate before system design.

시스템을 위한 비즈니스 요구를 인식하고, 시스템 성공여부에 관심이 있는 사람은 프로젝트 스폰서이다.

* Who recognizes the strong business need for a system and has an interest seeing the system succeed is project sponsor.

**시스템 요청서 (System Request) 5가지 구성요소**

1. 프로젝트 스폰서 (project sponsor)
2. 비즈니스 필요성 (business need)
3. 비즈니스 요구 (business requirement) (무엇을 하는가)
4. 비즈니스 가치 (business value)
5. 제한 사항 (special issue or constraints)

**타당성 분석**

1. **기술적 타당성 (Technical feasibility)**

* 응용분야에 익숙한가?

(Familiarity with the functional area)

* 기술에 익숙한가? (Familiarity with the technology)
* 프로젝트 크기 (Project size)
* 호환성 (Compatibility)

1. **경제적 타당성 (Economic feasibility)**

* 비용 및 수익요소 파악

(Identifying costs and benefits)

* 비용 및 수익 금액 산정

(Assigning values to costs and benefits)

* 순 현재가치 계산

(Determining Net Present Value (NPV))

* 투자자본 수익률 계산

(Determining Return on Investment (ROI))

* 손익분기점 계산

(Determining the Break-Even point)

1. **조직적 타당성 (Organizational feasibility)**

**현재 가치 (Present Value (PV)) =**  (n = year)

* The amount of an investment today compared to that same amount in the future, taking into account inflation and time.

**순 현재 가치 (Net PV) =**

* The present value of benefit less the present value of costs

**투자자본 수익률 (Return on Investment)**

**=**

* The amount of revenues or cost savings results from a given investment.

**손익분기점 (Break-Even point) =**

* The point in time at which the costs of the project equal the value it has delivered.

타당성 분석은 프로젝트 팀이 시스템에 관하여 중요한 결정을 했을 때 여러 번 개정되어야 한다.

* Feasibility study should be revised several times during the project at points where the project team makes critical decisions about the system.

프로젝트 선택 (Project selection)

* 타당성 분석이 완료되면, 그 프로젝트는 승인위원회로 시스템 제안서와 함께 제출된다. 그 위원회는 그 프로젝트를 승인할 지, 거절할 지, 추가적인 정보가 가능할 때까지 보류한다. 포트폴리오 관리(Portfolio management)는 조직 내에 존재하는 여러 종류의 프로젝트에 적용된다. 결정은 위원회에서 하기 때문에, 좋은 전망이 있어도 여러 가지 이유로 그 프로젝트는 거절당할 수 있다.

작업 분할 구조 (Work Breakdown Structure)

* 최소한 WBS는 작업기간, 작업 현재 상태, 작업 의존성을 포함하고 있어야 한다.
* WBS must include the duration of the task, the current status of the task and the task dependencies.

1. **Gantt Chart**

* 많은 작업들은 헷갈리게 된다. 그래서 작업의 숫자는 20 ~ 30개로 제한하는 것이 제일 좋다.

1. **Network Diagram**

**Key Definitions**

**@프로젝트 관리(Project management)는 적절한 기능을 통해 최소 비용으로 지정된 기간 내에 시스템 개발을 계획하고 제어하는 프로세스이다,**

**@프로젝트 관리자(Project manager)는 신중하게 조정해야 하는 수백 가지 작업과 역할을 관리**

**프로젝트 관리에서의 Four Key Steps**

**@프로젝트 규모 식별(Identifying project size)**

**@Project size, cost, time의 균형**

**@시간과 노력을 위해 예상 값을 할당하는 과정**

**@추정의 근원(Sources of estimates)**

**@사용중인 방법론**

**@실제 이전 프로젝트**

**@숙련된 개발자**

**@견적(추정)은 범위로 시작하여 프로젝트가 진행됨에 따라 더욱 구체적으로 나타난다.**

**@시스템 size를 추정 -> 요구되는 노력 추정 -> 요구되는 시간 추정**

**@작업 계획 작성 및 관리(Creating and managing the workplan)**

**@방법론**

**@표준 작업 목록 사용**

**@Top-down 접근법**

**@높은 수준의 작업 식별**

**@점점 더 작은 단위로 나누기**

**@업무 분류 체계 구성**

**@Project Workplan**

**@작업 분석 구조의 모든 작업의 목록**

**@작업 이름, 작업 기간, 현재 작업 상태, 작업 의존성, 이정표(날짜)(Milestone)**

**@Tracking Project Tasks**

**@Gantt Chart**

**@바 차트 형식**

**@언제든지 프로젝트를 모니터링하는데 유용**

**@PERT Chart**

**@Flowchart 형식**

**@작업 의존성 및 중요 경로 설명**

**@프로젝트 staffing(Staffing the project)**

**@Staffing Attributes**

**@Staffing level은 프로젝트의 생명시간에 따라 바뀐다**

**@staff(직원)를 추가하는 것은 추가 인력보다 더 많은 시간이 들 수도 있다.**

**@계층적 구조에서 8~10보고 팀을 사용하면 복잡성을 줄일 수 있다.**

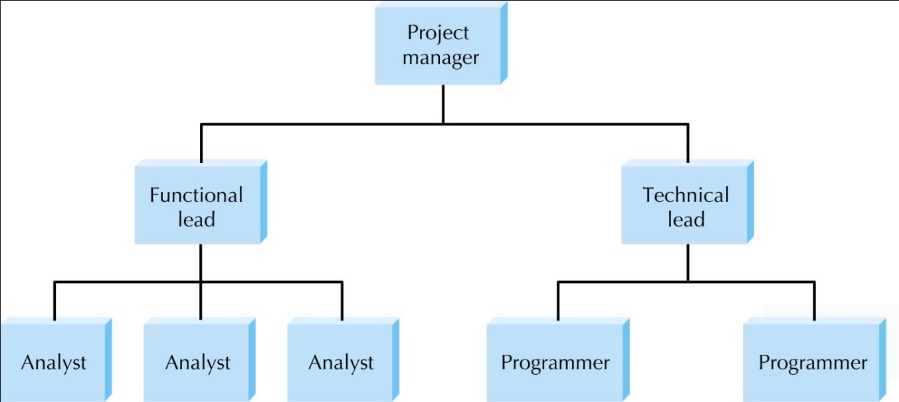
**@Key Definitions**

**@staffing plan은 프로젝트에서 일하는 사람들의 종류를 묘사한다.**

**@project charter는 프로젝트의 objectives와 규칙을 설명한다**

**@functional lead(기능적 책임자)는 분석가들의 그룹을 관리한다.**

**@technical lead(기술 책임자)는 프로그래머 및 기술 직원의 진행 상황을 감독합니다.**

****

**@Motivation(동기)**

**@동기는 사람들의 성과에 가장 큰 영향을 미친다**

**@고유 보상 사용(Use intrinsic rewards)**

**@인식, 성취, 작품 자체, 책임, 진보, 새로운 기술을 배울 수 있는 기회**

**@금전적 보상을 신중하게 사용해라**

**@Documentation(문서)**

**@Project binder**

**@모든 결과물과 모든 내부 커뮤니케이션을 포함**

**@지속적인 업데이트**

**@리스크 관리**

**@위험 평가**

**@인력 부족, 범위 추가, 형편없는 설계, 지나치게 낙관적(optimistic) 추정**

**@위험을 줄이기위한 조치**

**@수정된 평가**

**@프로젝트 활동 조정(Coordinating project activities)**

**요약**

**@프로젝트 관리는 새 시스템의 성공적인 개발에 중요하다.**

**@프로젝트 관리는 계획, 통제, 제시간에 보고하는 것, 노동, 비용을 포함한다**

**@Project 시작**

**@타당성 분석**

**@프로젝트 선택**

**@전통적인 프로젝트 관리 도구**

**@프로젝트 노력 추정**

**@workplan 수립 및 관리**

**@프로젝트 staff**

**@프로젝트의 환경과 하부구조 workflows를 관리**

**Chapter 3**

**Requirements Determination**

System request

+feasibility analysis result

+project plan

System

System

proposal

System specification

**System development life cycle**

Feasibility analysis result

System request

**Project plan**

**Planning =** project initiation + project management

**Project plan** = workplan + staff + techniques

System request

**Functional requirements**

Project plan

Feasibility analysis result

**Nonfunctional requirements**

**Requirement Determination** = Functional requirements +

Nonfunctional requirements

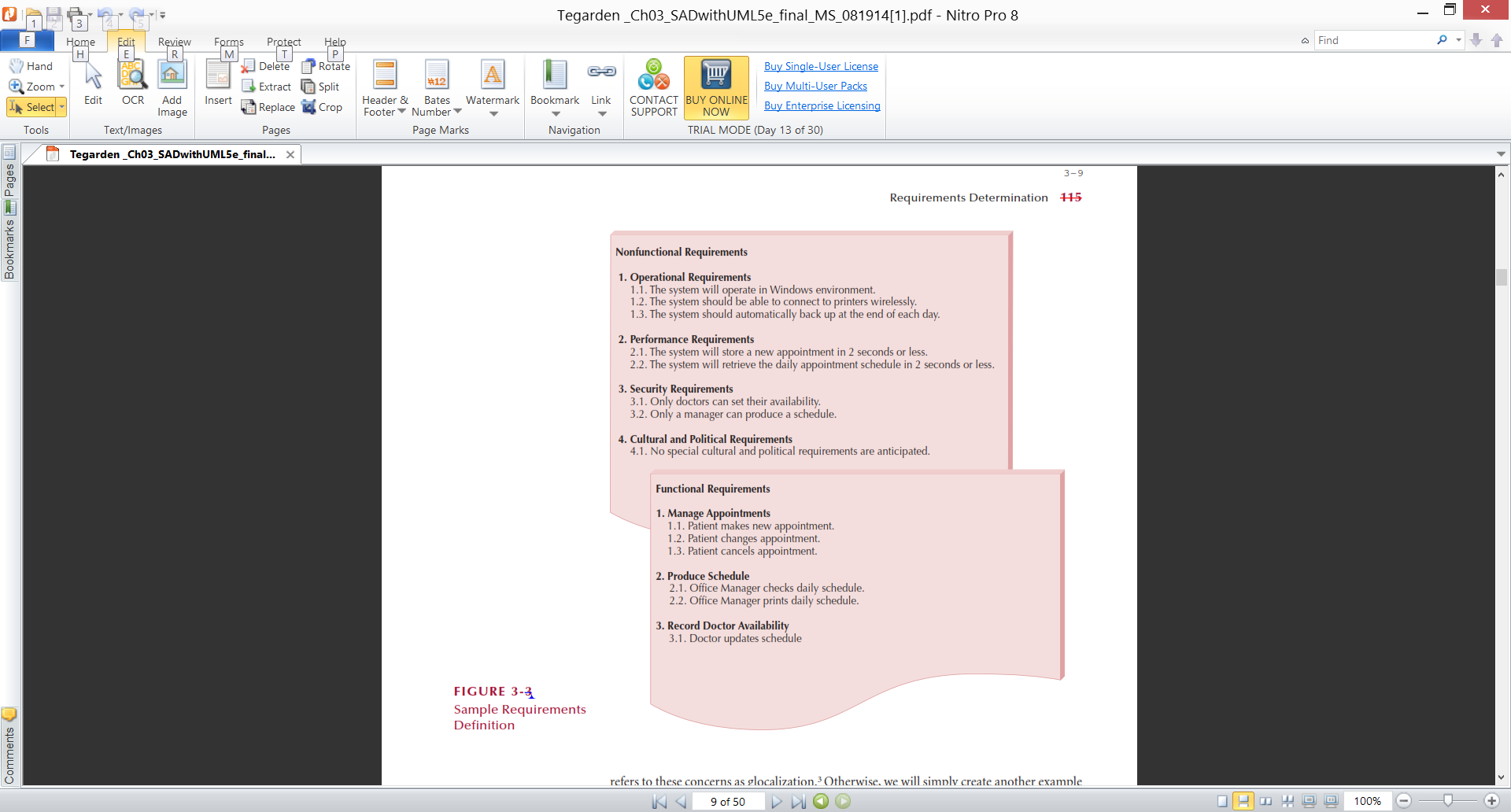
Nonfunctional requirements는 설계단계(Design)에서 반영됨.

Functional requirement는 시스템이 수행해야 할 프로세스나 그 시스템이 가지고 있어야 할 정보와 직접적 연관이 있다.

* A functional requirements relates directly to a process a system has to perform or information it needs to contain.

Nonfunctional requirement는 성능이나, 유용성 같은 시스템이 가지고 있어야 할 행동적 특성을 가리킨다.

* Nonfunctional requirements refer to behavioral properties that the system must have, such as performance and usability.



시간

보안

성능

디자인

등등..

Use case를 만드는 데 기반이 된다.

요구(Requirement)가 실제 비즈니스 요구(Business Need)는 반영하지만, 현재 시스템의 목적에 맞지 않을 때, 그 요구는 미래의 요구 목록에 추가되거나 낮은 우선권이 주어진다.

* When a requirement reflects a real business need but is not within the scope of the current system or current release, it is either added on a list of future requirements or given a low priority.

**요구 분석 전략 (Requirement Analysis Strategies)**

1. **Problem Analysis**

* 현 시스템의 문제점과 개선방안을 물어보는 방법

1. **Root Cause Analysis**

* 문제의 원인을 분석 (not 개선방안)

1. **Duration Analysis**

* 업무 수행 시간 분석

1. **Activity-Based Costing**

* Duration Analysis에서의 비용 분석

1. **Informal Benchmarking**

* 다른 곳을 베껴서 분석

1. **Outcome Analysis**

* 고객이 진짜로 원하는 것을 분석
* EX) 사고가 났을 때, 고객이 원하는 건 보험인가, 아니면 빠른 자동차 수리인가?

1. **Technology Analysis**

* 새로운 기술의 도입이 도움이 되나?

1. **Activity Elimination**

* 구조조정: 업무과정 생략 가능?

**요구 수집 전략 (Requirements Gathering Techniques)**

1. **회의 (Interview)**
2. **결합 응용 설계 (Joint Application Development, JAD)**
   1. **Project managers, users, developers가 함께 일할 수 있도록 허락한다.**
   2. **Scope creep이 50% 감소할거다.**
   3. **너무 구체적이거나 너무 모호한 욕들은 피해라**
   4. **촉진자(Facilitator)**
      1. **회의 안건(meeting agenda)을 설정하고 토론을 안내한다.**
      2. **토론에는 참여하지 않는다.**
   5. **학자(Scribe)**
      1. **Making copies, recording notes함으로써 facilitator를 도와준다**
   6. **3주동안 5~10일 지속**
   7. **인터뷰와 함께 질문을 준비**
   8. **형식적인 안건과 기본규칙(스케쥴 따르기, 다른 사람 존중)**
   9. **Facilitator(촉진자) 활동**
      1. **세션을 계속 추적**
      2. **기술 용어 및 전문 용어에 대한 도움말**
      3. **레코드 그룹 입력, 문제 해결에 도움**
3. **설문 (Questionnaires)**

* 조직 바깥 사람들에게 유리

1. **문서 분석 (Document Analysis)**

* 문서화가 제대로 되어있지 않은 프로젝트가 많음.

1. **관찰 (Observation)**

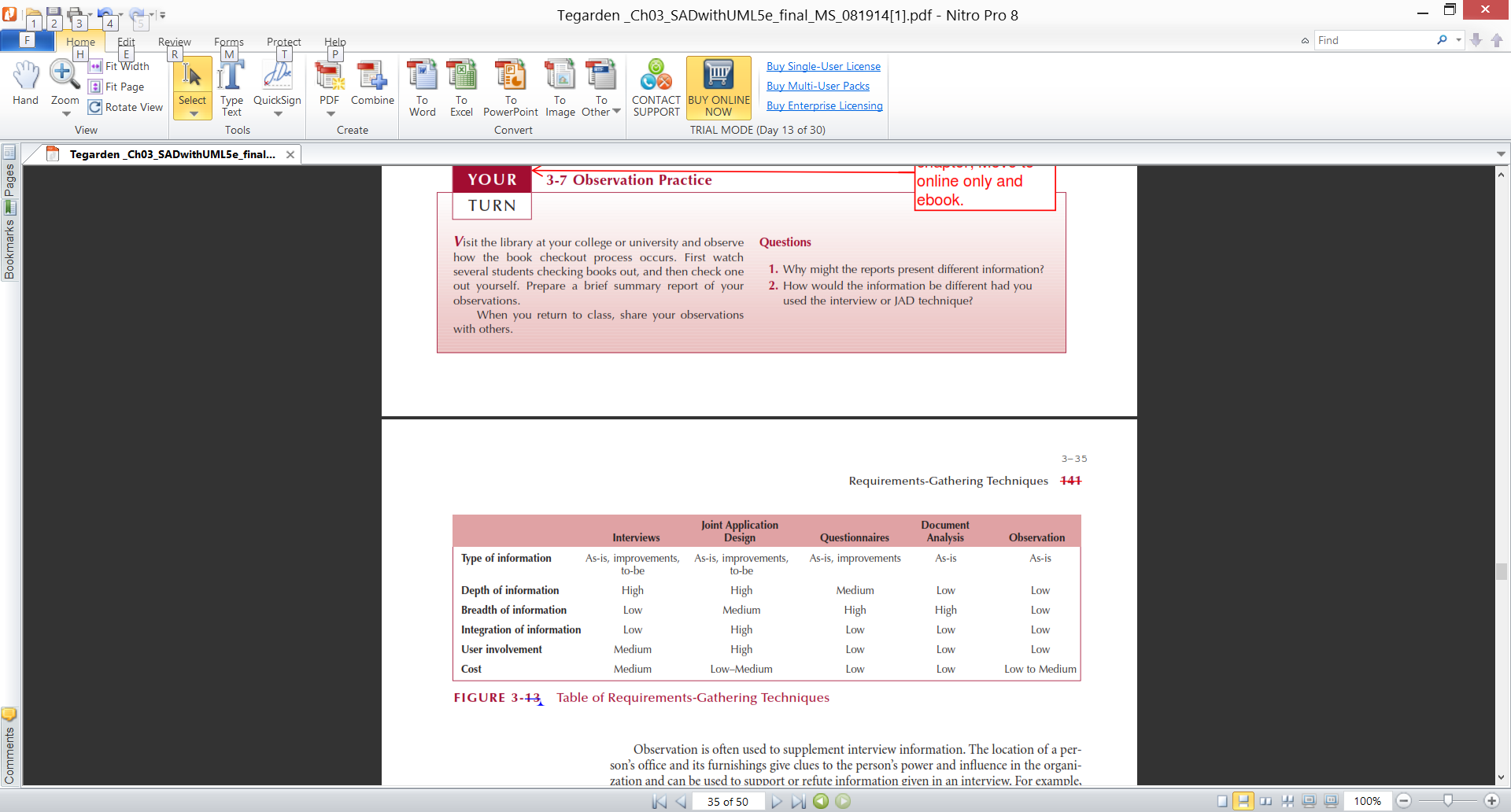
* 분석가가 직접 실제 상황을 보기 때문에 효과적이다. 또한 회의나 설문을 통한 간접적인 정보를 직접 확인할 수 있다.

1. **개념도 (Concept Maps)**

* 적은 개수의 핵심 개념에 집중할 때 유용하다.
* ****

1. **User story**

* Very useful for agile development



**@IT person 단독으로 하면 비즈니스 value가 없는 시스템을 만들고 Biz people들 단독으로 수행하면 잘못된 biz process를 그대로 전산화한다. -> 매우 비효율적 시스템을 만든다.**

* **따라서 같이 해야한다. (사업/분석가가 같이 해야 최고)**

**@기능 비기능을 정한다.**

**@요구 수집 기술을 이용한다**

**@분석가는 요구를 검증하고 변경하고 우선순위를 정한다. 사용자와 함께 Scope를 제한한다.**

**@필요하지만 이번 시스템에 포함되지 않는 것은 향후 계획으로 넘긴다.**

**@정확한 사용자를 파악하기 힘들며 부적절한 요구 명세가 있을 수 있다.**

**@처음에는 요구가 모두 나타나지 않으며 점진적, 반복적으로 해결**

**@검증이 힘들지만 아키텍쳐(functional, structural, behavioral model)를 만들 때 점검한다.**

**요약**

**@functional and non-functional requirements 결정의 토론**

**@Requirements 분석 전략**

**@문제 분석, root cause 분석, 기간 분석, 비용기반 활동 분석, 다른 것 밴치마킹 분석, 결과물 분석, 기술 분석 과 활동 제거**

**@Requirements gathering techniques(요구사항수집기술)**

**@인터뷰, JAD(joint application development), 설문, 문서 분석과 관찰**

**@대체 요구사항 문서화 기술**

**@concept maps, story cards, 업무 목록**

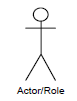
**@system proposal**

**Chapter 4**

**Business Process and**

**Functional Modeling**

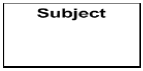
**Elements of Use-Case Diagrams**

1. **Actor** 

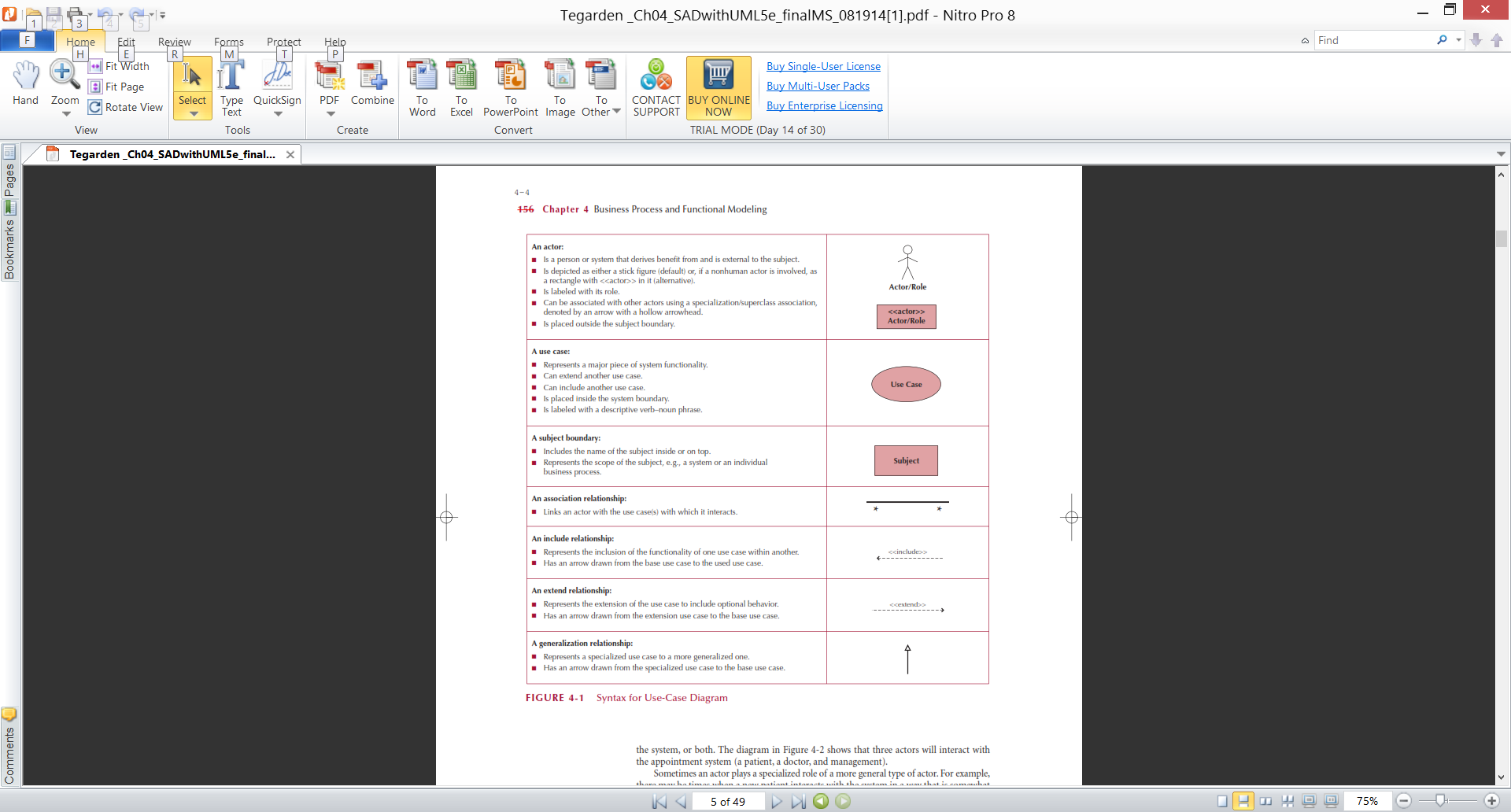
* Is a person or system that derives benefit from and is external to the subject.
* Is labeled with its role.
* Is placed outside the subject boundary

1. **Use Case** 

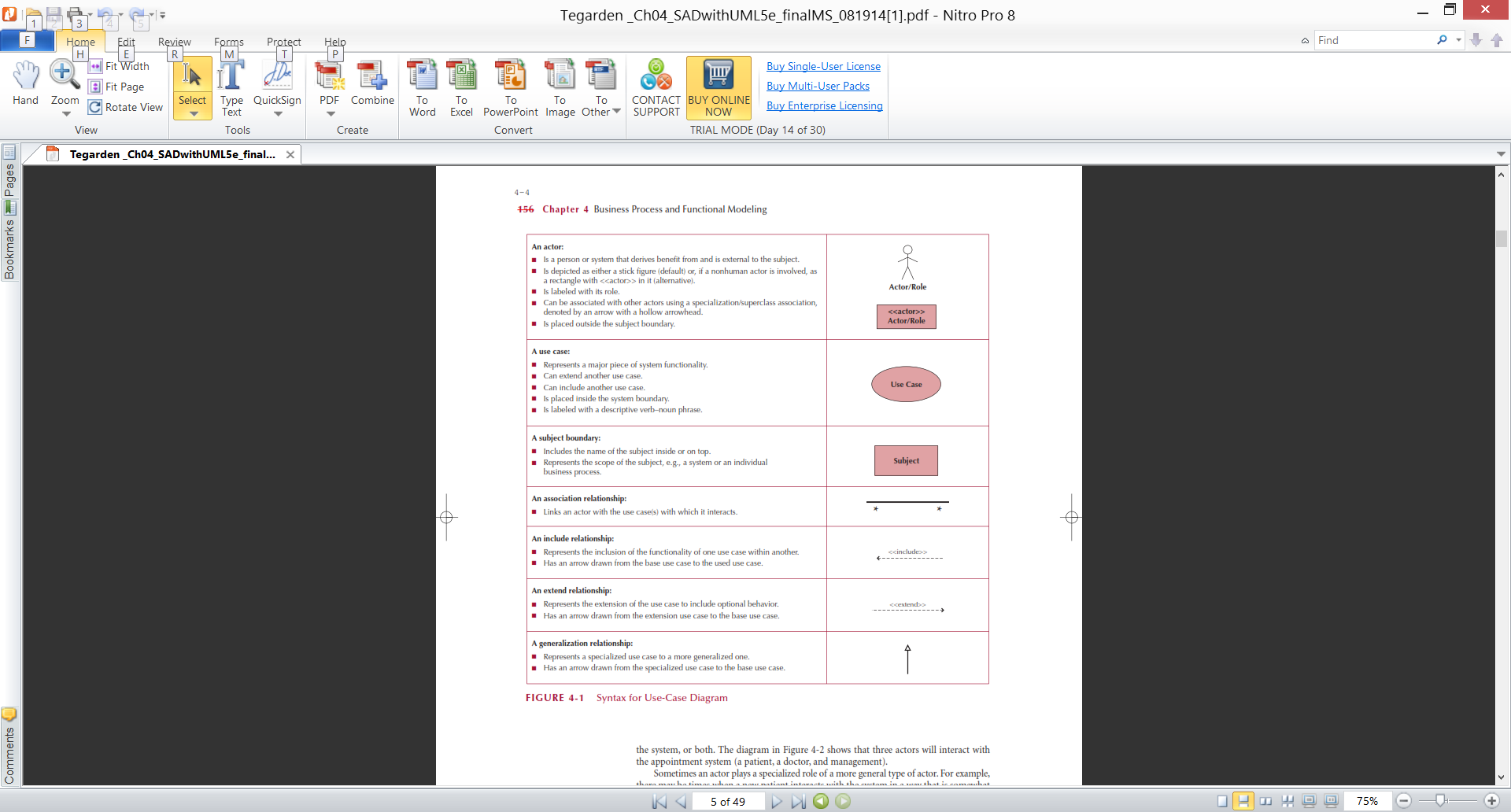
* Represent a major piece of system functionality.
* Is placed inside the system boundary.
* Is labeled with a descriptive verb-noun phrase.

1. **Subject boundary** 
2. **Association relationship**

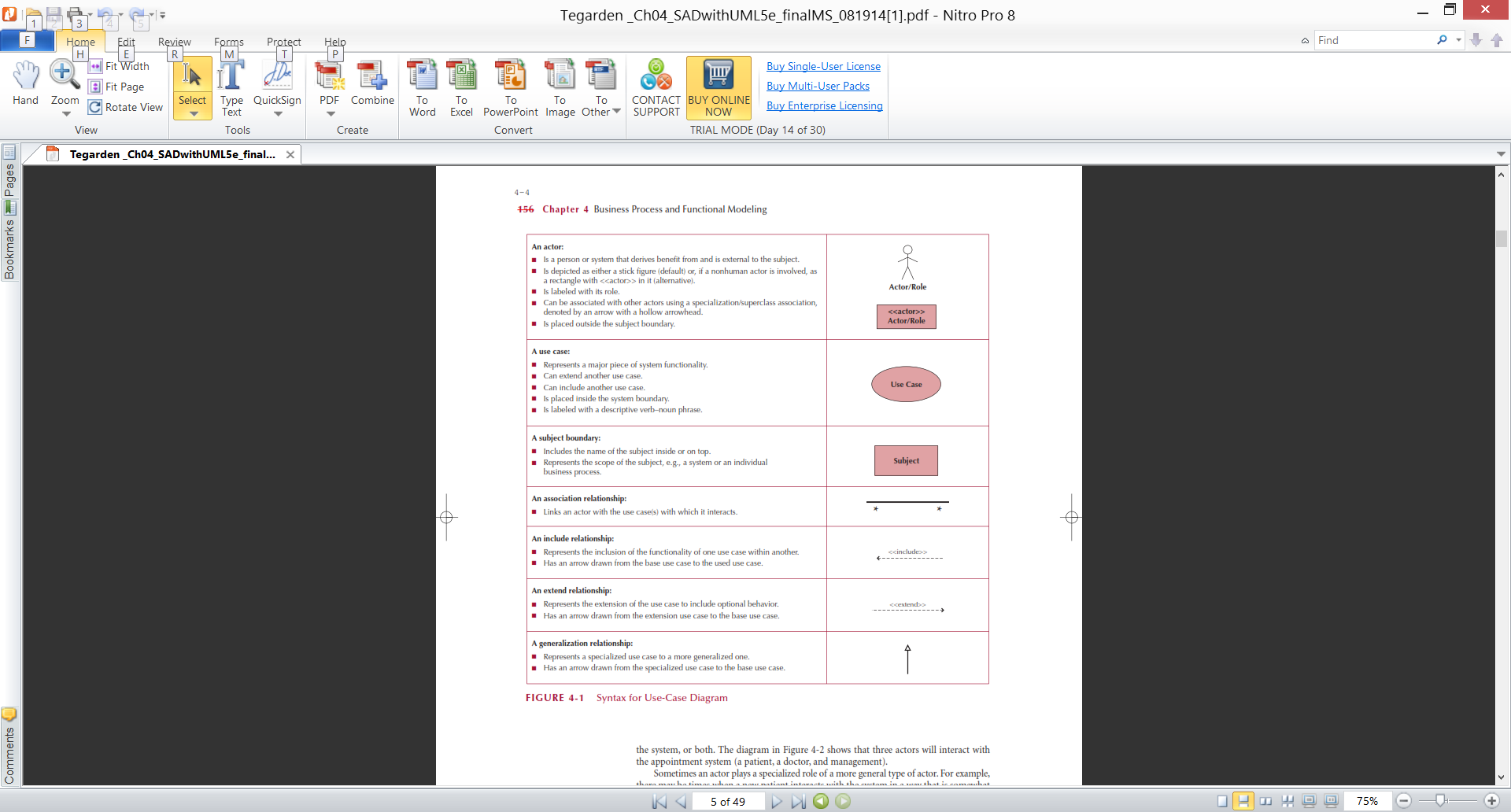
* Links an actor with the use case with which it interacts.

1. **Include relationship** 

* Has an arrow drawn from the base use case to the used use case.

1. **Extend relationship** 

* Represents the extension of the use case to include optional behavior.

1. **Generalization relationship** 

* Has an arrow drawn from the specialized use case to the base use case.

**Identifying the Major Use Case**

1. **Review Requirements definition**
2. **Identify Subject’s Boundaries**

* 주제의 범위를 확실하게 정한다. 하지만 추후에 바뀔 수 있음.

1. **Identify Primary Actors & Goals**

* 주요 Actor 및 Actor의 목적을 확인한다.

1. **Identify Business Processes & Major Use Cases**

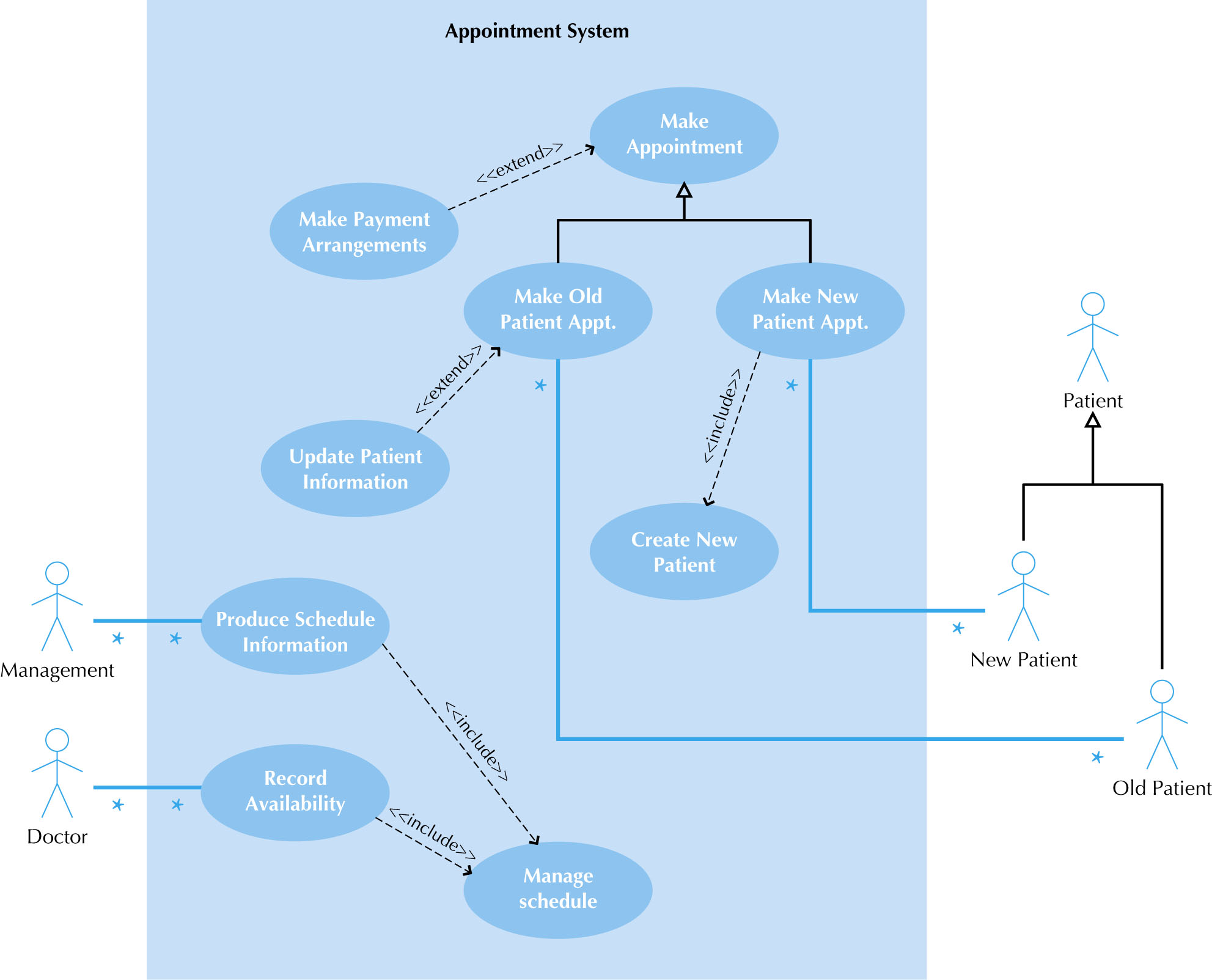
* 하나의 Use Case에 집중하는 게 아니라 전체적인 Use Case를 확인하는 과정임.

1. **Review Current Set of Use Cases**

**Creating Use-Case Diagram**

1. Place & Draw Use Cases
2. Place & Draw Actors
3. Draw Subject Boundary
4. Add Associations

**Use Case Diagram for the Manage Appointment Use Case**



**Elements of Activity Diagram**

1. **Actions and Activities**

* The only difference between an action and an activity is that an activity can be decomposed further into a set of activities and/or actions, whereas an action represents a simple nondecomposable piece of the overall behavior being modeled.

1. **Object Nodes**

* Essentially, object nodes represents the flow of information from one activity to another activity.

1. **Control Flows and Object Flows**

* Control flows can be attached only to actions or activities
* Object flows must be attached to an action or activity on one end and an object node on the other end.

1. **Initial, Final-activity, Final-flow node** DRW000014842f39

* An initial node portrays the beginning of a set of actions or activities
* A final-activity node is used to stop all control lows and object flows in an activity
* A final-flow node is similar to a final-activity node, except that it stops a specific path of execution through the business process but allows the other concurrent or parallel paths to continue.

1. **Decision, Merge node** DRW000014842f3b

* A decision node is used to represent a test condition to ensure that the control flow or object flow only goes down one path.

A decision node is labeled with the decision criteria to continue down the specific path.

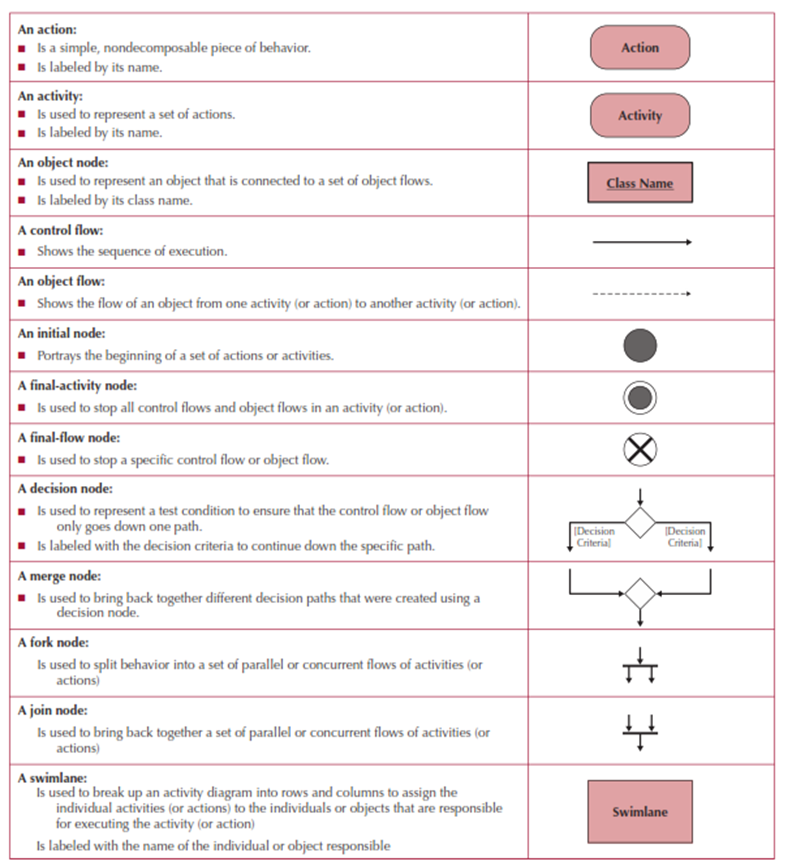
1. **Fork, Join node** DRW000014842f3d

* Used for a set of parallel or concurrent flows of activities or actions

1. **Swimlane**

* Is used to break up an activity diagram into rows and columns to assign the individual activities or actions to the individuals or objects that are responsible for executing the activity

**Syntax for an Activity Diagram**



**Activity Diagram for the Manage Appointment Use Case**

