

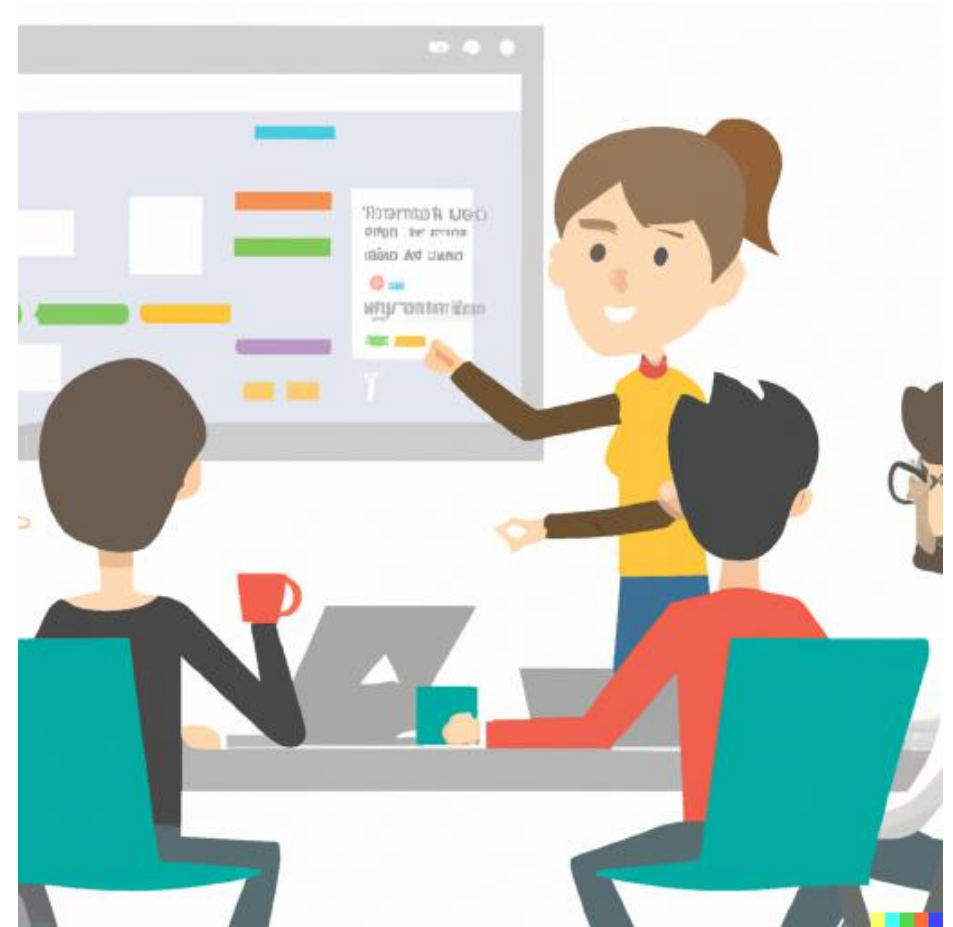
소프트웨어공학

소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유



소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유

- 특별한 상황이 아니라면 요구사항 명세는 개발의 핵심
 - 소프트웨어를 성공적으로 개발하려면 다양한 이해관계자의 요구사항을 적절히 반영하는 것이 중요



소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유

- 복잡하고 큰 규모의 소프트웨어 개발, 모델링

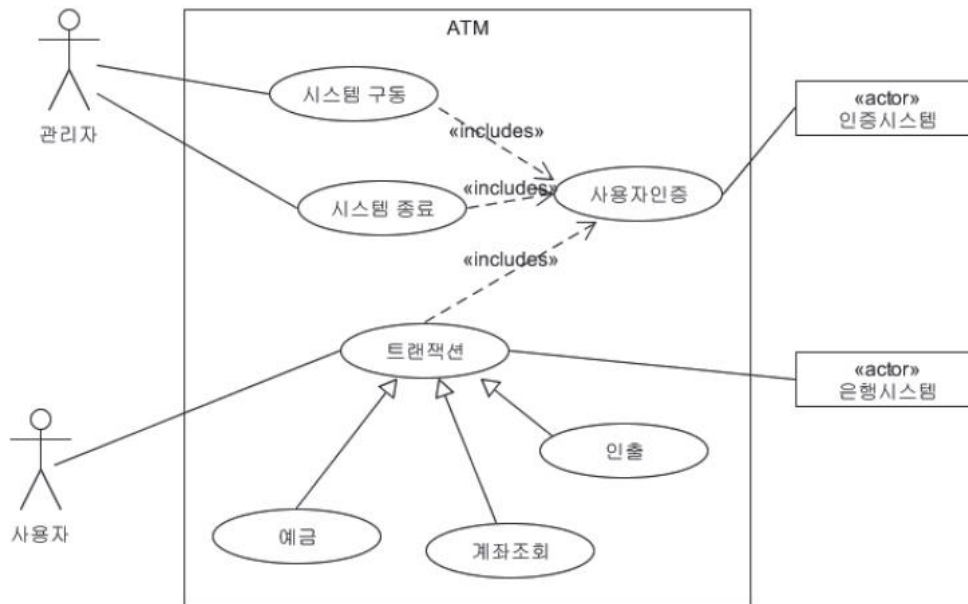


그림 2-2 ATM 시스템 유스케이스 다이어그램 예제

표 2-4 「인출」 유스케이스

유스케이스명	인출
액터명	주 액터: 사용자 부 액터: 인증시스템, 은행시스템
개요	사용자가 자신의 계좌로부터 예금 출금을 하기 위해 ATM을 이용한다.
사전조건	<ul style="list-style-type: none"> • 사용자가 은행 카드를 소지하고 있어야 한다. • 은행시스템과 인증시스템과의 네트워크 연결이 정상적이어야 한다. • 시스템은 사용자가 요구한 출금액만큼의 현금을 가지고 있어야 한다.
사후조건	<ul style="list-style-type: none"> • 사용자가 카드를 돌려받는다. 사용자가 돈을 출금하고 은행계좌에 출금액이 반영된다. • 사용자가 카드를 돌려받는다. 사용자가 돈을 출금하지 못하고 은행계좌는 변동이 없다. • 사용자가 카드를 돌려받지 못한다. 사용자가 돈을 출금하지 못하고 은행계좌는 변동이 없다. 또한 은행을 접촉하라는 메시지를 받는다.
기본흐름	<ol style="list-style-type: none"> 1. 사용자는 은행 카드를 시스템에 제시한다.(트리거) 2. 시스템은 카드 정보를 읽고 비밀번호를 요구한다. 3. 사용자는 비밀번호를 입력한다. 4. 「사용자인증」 유스케이스를 실행한다. 5. 시스템은 사용자에게 트랜잭션 타입을 요구한다. 6. 사용자는 출금 트랜잭션을 선택한다. 7. 시스템은 출금액을 사용자에게 요구한다. 8. 사용자는 출금액을 입력한다. 9. 시스템은 은행시스템에 출금을 요구한다. 은행시스템은 사용자의 계좌에 출금을 반영한다. 10. 시스템은 돈을 사용자에게 내주고 출금 사실을 로그기록에 반영 한다 11. 시스템은 은행카드를 사용자에게 되돌려 준다.
대체흐름 1	2a. 시스템이 인식하지 못하는 카드가 입력되는 경우 2a.1 시스템은 사용자에게 카드 판별할 수 없다는 사실을 알리고 유스케이스를 종료한다.
대체흐름 2	9a. 잔금이 모자라는 경우 9a.1 사용자에게 알리고 단계 7을 다시 수행한다.

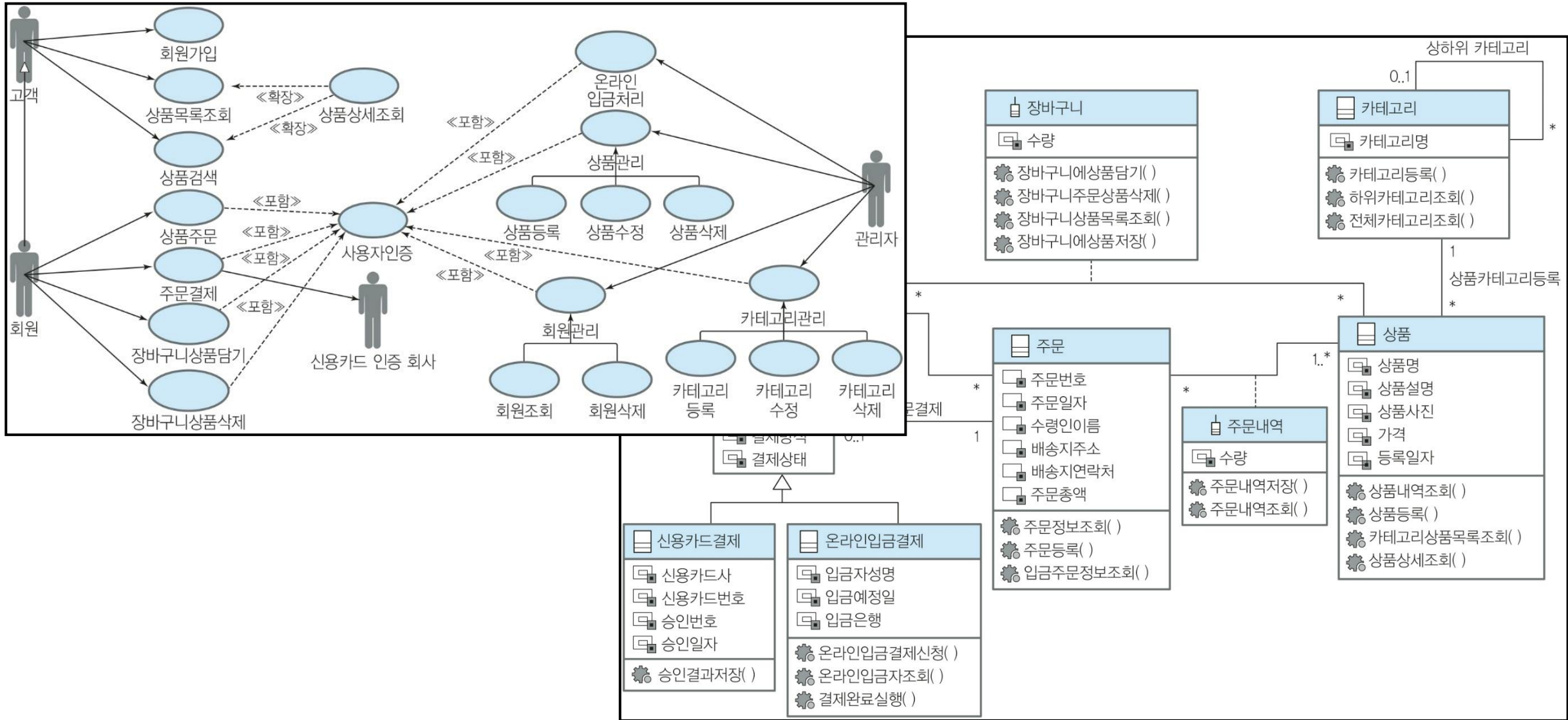
소프트웨어 공학 및 UML(Unified Modeling Language)

- 캡스톤 디자인 교과목 제안서 예시

목차

1 프로젝트 수행 목적	3
1.1 프로젝트 정의	3
1.2 프로젝트 배경	3
1.3 프로젝트 목표	4
2 프로젝트 개요	5
2.1 프로젝트 설명	5
2.2 프로젝트 구조	5
2.3 시나리오	6
2.4 기대효과	11
2.5 제약조건	11
2.6 관련기술	12
2.7 개발도구	12
3 프로젝트 추진 체계 및 일정	14
3.1 역할 분담	14
3.2 작업 흐름도	14
3.3 개발 일정	16
4. 참고자료	17

소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유



소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유

- 다양한 이해관계자들이 앞으로 개발하려고 하는 소프트웨어의 실제 개발 내용을 이해하기 쉬움

☞ 의사소통 과정에서의 오류, 모순 및 개발 과정에서의 시행착오를 최대한 줄여줄 수 있다.

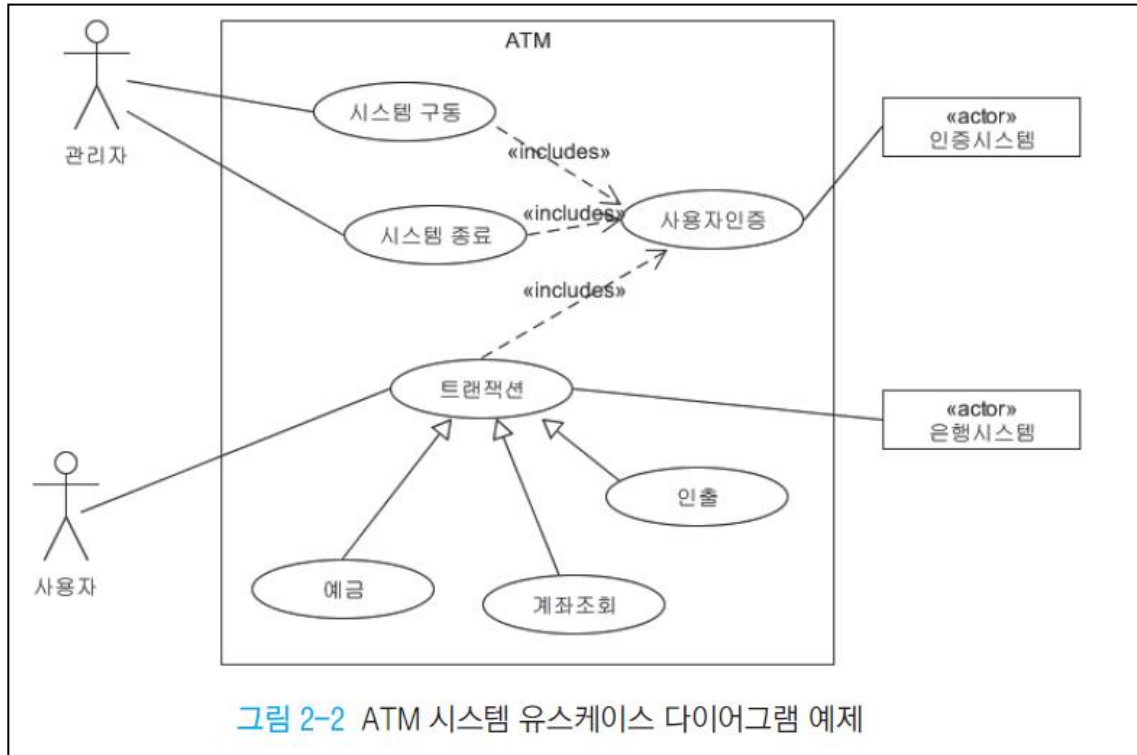
- 특히 최근의 복잡하고 규모가 큰 시스템 개발의 경우,

앞으로 개발할 소프트웨어에 대해 해결해야 할 문제가 정리되지 않으면 해결책도 명확하게 정리할 수 없고,
시행 착오를 줄이면서도 적은 비용으로 보다 빨리 좋은 소프트웨어를 개발할 수 있도록

과학적, 공학적 원리와 지식, 방법을 활용해야 함



소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유



001: Withdraw Cash

설명고객이 ATM으로부터 자신의 계좌에 입금되어 있는 현금을 출금한다.
이벤트흐름

2.1 기본흐름

1. 사용자는 ATM 기계에 카드를 넣는다.
2. ATM은 사용자에게 초기 화면을 출력한다.
3. 사용자는 ATM기계의 인출 버튼을 클릭한다.
4. ATM은 인출 화면을 출력한다.
5. 사용자는 ATM에 인출금액을 입력하고 확인 버튼을 클릭한다. (A1)
6. ATM은 사용자에게 영수증을 발급여부를 물어본다. (E1)

1.2.2 대안 흐름

A1. 통장 잔액이 부족한 경우, 인출 불가 화면을 띄운다.

1.2.3 예외 흐름

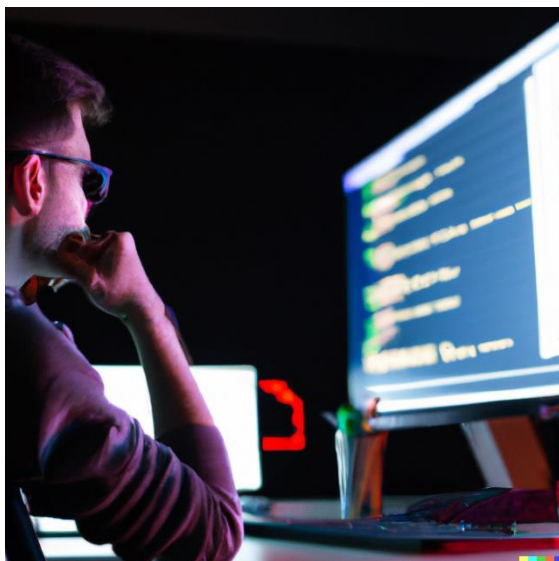
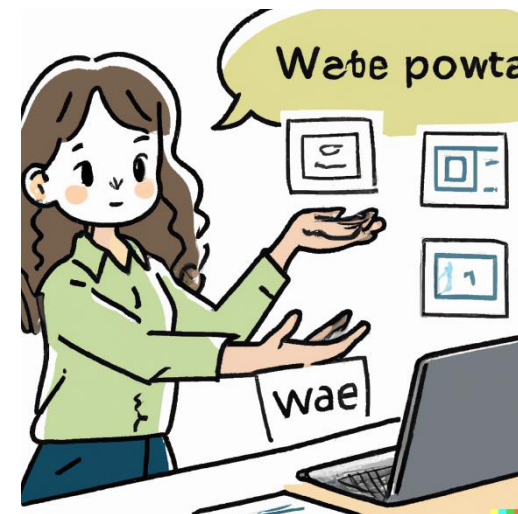
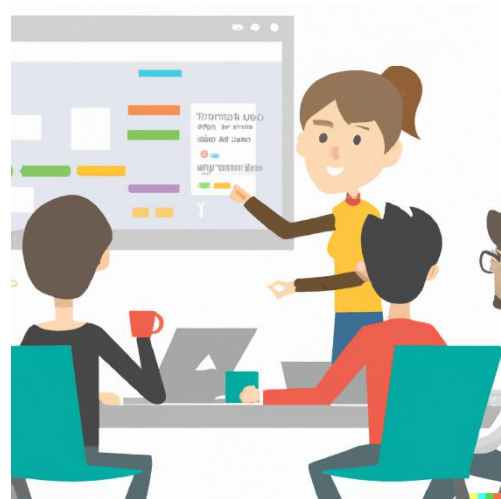
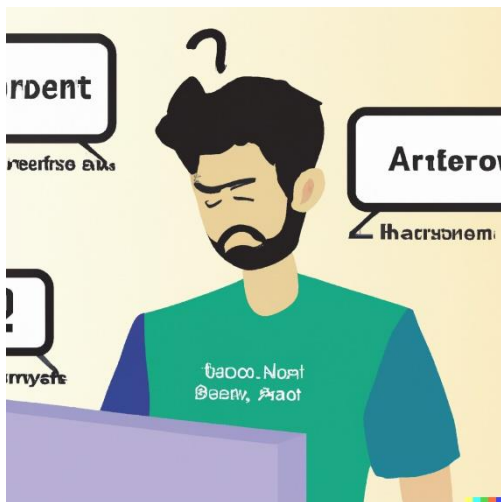
E1. 영수증 인쇄지가 부족한 경우, 시스템 에러 화면을 띄운다.

소프트웨어 공학 및 UML(Unified Modeling Language)을 잘 알아야하는 이유

- 다양한 실제 이해당사자들이 소프트웨어를 사용했을 때 얻을 가치를 고민해 볼 수 있고 개발하려고 하는 소프트웨어에 대해 종합적인 통찰을 얻을 수 있다.

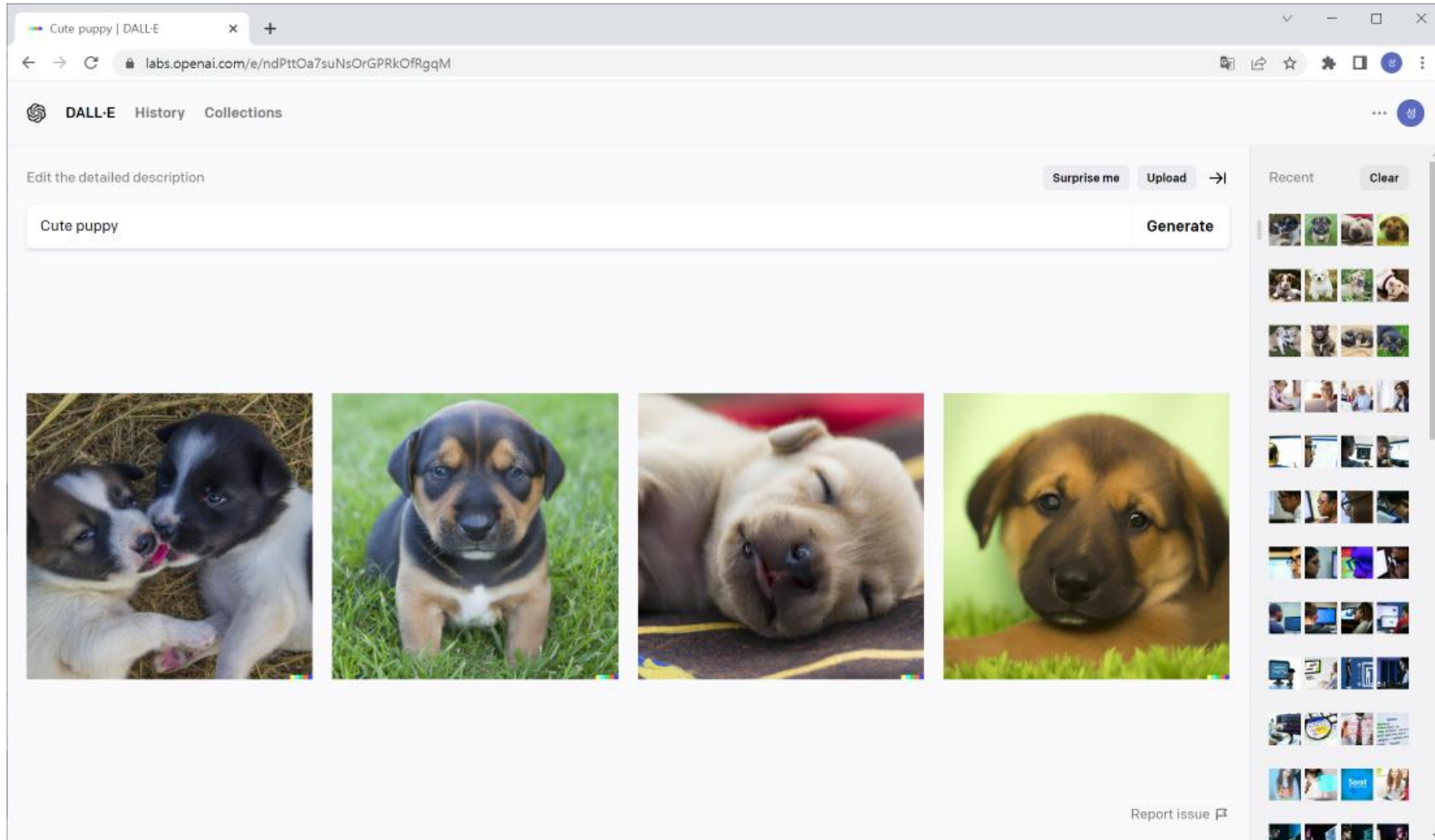


DALL·E로 그린 그림



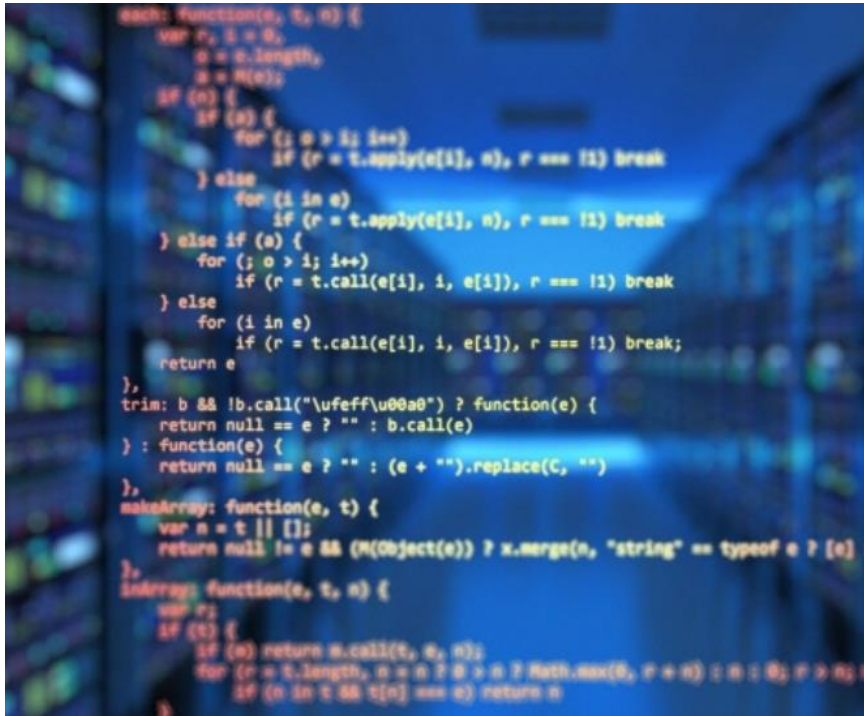
DALL·E로 그린 그림

HTML, CSS, JavaScript, 뷰(Vue.js), 리액트(React.js), JSP, 스프링, 스프링부트, 데이터베이스, 클라우드, AI



DALL·E로 그린 그림

인공지능은 소프트웨어!



```
each: function(e, t, n) {  
  var r, i = 0;  
  o = e.length;  
  a = n(n);  
  if (n) {  
    if (n) {  
      for (; o > i; i++)  
        if (r = t.apply(e[i], n), r === !1) break  
    } else  
      for (i in e)  
        if (r = t.apply(e[i], n), r === !1) break  
  } else if (n) {  
    for (; o > i; i++)  
      if (r = t.call(e[i], i, e[i]), r === !1) break  
  } else  
    for (i in e)  
      if (r = t.call(e[i], i, e[i]), r === !1) break;  
  return e  
},  
trim: b && !b.call("\uffff\u00a0") ? function(e) {  
  return null == e ? "" : b.call(e)  
} : function(e) {  
  return null == e ? "" : (e + "").replace(C, "")  
},  
makeArray: function(e, t) {  
  var n = t || [];  
  return null != e && (N(Object(e)) ? x.merge(n, "string" == typeof e ? [e]  
),  
isArray: function(e, t, n) {  
  var r;  
  if (n) {  
    if (n) return n.call(t, e, n);  
    for (r = t.length, n = n ? Math.max(0, r + n) : n < 0; r > n; r++)  
      if (n in t && t[r] === n) return n  
  }  
}
```



```
if ($(window).scrollTop() > header1_initialDistance) {  
  if (parseInt(header1.css('padding-top'), 10) > header1_initialPadding) {  
    header1.css('padding-top', '' + $(window).scrollTop() - header1_initialDistance)  
  }  
} else {  
  header1.css('padding-top', '' + header1_initialPadding + 'px');  
}  
  
if ($(window).scrollTop() > header2_initialDistance) {  
  if (parseInt(header2.css('padding-top'), 10) > header2_initialPadding) {  
    header2.css('padding-top', '' + $(window).scrollTop() - header2_initialDistance)  
  }  
} else {  
  header2.css('padding-top', '' + header2_initialPadding + 'px');  
}
```


AI 학습 : 많은 양의 데이터부터 일반적인(공통적인) 특징을 찾는 것



AI 학습 : 많은 양의 데이터부터 일반적인(공통적인) 특징을 찾는 것

■ 머신러닝(Machine Learning)

- 계속된 교육과 경험을 통해 보다 정확한 판단을 하고, 다른 것도 쉽게 받아들일 수 있는 사람의 능력을 흉내내는 기술

예) 많은 양의 데이터로부터 일반화된 규칙을 찾아(학습) 임의의 새로운 데이터를 분류 및 예측하는 알고리즘 등

■ 자연어 처리(Natural Language Processing (NLP))

- 사람이 말하는 것 또는 글로 쓴 것을 이해하고 대응하는 (사람의) 능력을 흉내 내는 기술

예) 키워드 중심 인터넷 서치 엔진과 검색, 기계 번역 등

■ 컴퓨터 비전Computer Vision

- 눈에 들어 오는 모든 것을 인지하고 대응하는 사람의 능력을 흉내 내는 기술

예) 이미지 분류, 객체 인식, 위험 상황 인지 등

최근 인공지능 성능이 좋아진 원동력



Computing Power

강력한 병렬 및 분산처리 능력

Big Data Power

인터넷, IOT, Sensor 기술을 통한 데이터 수집 능력

Algorithm, Open Source Software

개방, 공유, 협업의 성과



 **T h a n k y o u**

TECHNOLOGY

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Velit ex
plicabo ipsum, labore sed tempora ratione asperiores des
cenderat bore sed tempora rati jgert one bore sed tem!