1. **UML(Unified Modeling Language)의 이해**

통합된 모델링 언어, 요구사항을 약속한 언어로 기술하기 위해 만들어진 언어이다. 소프트웨어는 다양한 층의 이해관계와 위계질서 아래 탄생하는 예술작품이다. 객체지향은 현실의 모방이 아니다. 현실의 질서를 현실과 다른 질서 속에서 유용하기 위한 창조활동이다. 이를 위해서는 각 이해관계자들의 현실 인식이 필요하다. 각자의 인식과 이해 관계는 복잡하게 얽혀 있다. UML은 이러한 복잡한 인식, 이해를 명료화하기 위한 도구이다.

UML 작성은 문제에 대한 고객사, 개발자, 경영진 등 각자의 관점을 종합해 모두가 이해할 수 있는 양식으로 재창조하는 행위이다. 따라서

1. 읽기 쉬워야 한다.

**UML은 시각화 언어이다.** 시각화 언어라는 말은 장문의 문장을 일일이 분석할 필요가 없다는 말이다. 많은 UML은 그림과 도형으로 표현된다. 한 자리에 어렵게 모인 이해관계자들이 머리를 싸매서 나온 결과물이 다시 머리를 싸매야 읽을 수 있는 문서라면 굳이 작성할 필요가 없지 않을까? UML은 직관적이고 시각적으로 와 닿아야 한다.

1. 구체적이어야 한다.

**UML은 명세화 언어이다.** UML작성은 개념을 잡는 일이다. 나쁘게 얘기하면 생산량 자체에는 직접적인 영향을 미치지 않는다. 매출에 직접적인 영향을 미치지 않는데 비용은 들면서 개념을 잡겠다는, 더 나쁘게 말하면 비용은 드는데 모여서 뜬구름 잡는 소리나 하는, 시간을 축내는 일처럼 보여질 수 있다. 그렇게 해서 나온 결과물이 진짜 뜬구름을 잡고 있다면 욕먹는다. 그래서 UML은 구체적이어야 한다. 소프트웨어 개발 과정인 분석, 설계 단계에서 필요한 모델을 정확하고 완전하게 **명세화(Specification)해야 하는 일이다.** 명세화된 UML은 앞서 말했듯 그림(기호)을 통해 시각적으로 표현된다.

1. 소스코드와 호환 가능해야 한다.

UML은 C, Java, C++, Visual Basic, C# 등 다양한 언어로 표현될 수 있다. 달리 말해 **UML로 설계된 모델은 코드로도 구축할 수 있어야 한다.** 또 **이미 구축된 코드도 UML로 역변환될 수 있어야 한다.** 이를 리버스 엔지니어링이라고 한다. 그래서 외워야 하는 사항은 **“UML은 구축언어이다.”**이다.

1. 문서화할 수 있어야 한다.

**UML은 문서화 언어이다.** 아까 긴 문장을 분석하지 않기 위해 읽기 쉬운 시각화 언어로 구축한 게 UML이라고 했는데 문서화 언어라는 건 또 무슨 소리일까? 생각해보라. 기껏 시간, 비용을 들여 나온 작품이라는 게 애들도 이해할 정도로 쉬운 졸라맨이 공을 던지고 있는 그림이라면(3장에서 설명할 유스케이스 다이어그램에서 실제로 졸라맨이 공을 던지는 그림을 사용한다.) 윗 사람들이 언짢아 할 수 있다.는 건 농담이고 사실 그렇게 쉽게 해놔도 못 알아먹는 사람이 있을 수 있고 또 그렇게 나타낸 그림이 실제 요구사항과 얼마나 맞아 떨어지는지 확인하고 대조할 필요도 있다. 그렇기에 지금 이 글처럼 문서에 잡설이 많이 들어갈 수 있다. 좋게 얘기하면 구체화하는 거고 나쁘게 얘기하면 사족이다. 어찌됐든 불편할 사람이 없도록 구체적인 문서를 마련할 필요가 있는 것이다. 요즘은 편리하게도 StarUML, 투게더 등의 케이스 툴을 이용해 설계한 내용을 자동으로 문서화할 수 있다고 한다.

종합.

UML은 다양한 이해관계자들이 서로 무슨 말을 하는지, 요구가 뭔지 파악하기 위해 마련한 합의이며, 이행해야 할 리스트이다. 따라서 한 분야의 문외한도 알아들을 만큼 읽기 쉬워야 하고, 이를 토대로 실제 구현을 할 수 있을 정도로 구체적이어야 하며, 코드로 구현하거나 역으로 코드가 UML로 구현될 수 있어야 하고, 못 알아듣는 사람이 없도록 문서화가 되어 있어야 한다. 결론은, UML은 **시각적 언어, 명세화 언어, 구축언어, 문서화 언어**라는 사실을 외우면 된다.

다음으로 알아야 할 사실은 UML이 객체지향 방법론을 실현하기 위한 툴이라는 것이다. 하지만 객체지향은 객체지향 문서에 정리할 생각이기에 굳이 반복하지 않겠다. 만약 UML 을 배우고 싶은데 객체지향을 모른다면 객체지향부터 공부하고 UML을 공부하는 편이 나을 것이다. 사실 “객체지향을 잘 안다.”의 기준이 실질적으로 객체지향 프로그램을 잘 구축할 수 있는지 없는지에 있다면 나도 객체지향을 잘 모른다. 여러 번 관련 책과 레퍼런스를 찾아봤기에 개념에 조금 익숙해져 있을 뿐이라고 생각한다. 객체지향 언어와 인터페이스, 추상 클래스 등을 공부해 본 경험이 있는 정도라면 계속 UML 공부를 진행하고 그게 아니라면 객체지향 언어와 개념 공부를 우선하는 게 좋을 것 같다.

**모델링 방법론**

1994년경 객체지향 방법론은 50개가 넘었다고 한다. UML 개요를 공부한 시점인 지금 공부해봐야 많아서 다 못 외우고 다 못 쓴다. 그래서인지 1994년에 이렇게 많은 방법론을 통합할 필요성이 대두됐다. 그래서 나온 것 중에 하나가 이 UML이다. UML은 부치, 야콥슨, 럼바에 의해 고안된 객체지향 방법론이다. 럼바의 경우 정보처리기사 필기 문제에 선지로 출제된 적이 여러 번 있으니 그쯤 알고 우선은 넘어가는 걸 추천한다. 더 자세히 공부하려면 Chat-gpt에게 물어보시라. 이런식으로 중요한 이름으로는 부치 방법론, 야콥슨의 OOSE, 럼바의 OMT, 등이 있다. 개인적으로는 지금 이 부분을 읽고 외워봐야 휘발될 확률이 높으니, 더 공부하다가 필요성이 느껴지면 다시 찾아보려고 한다.