실무형 코드분석 가이드 문서

(2020.09.14. (재)이노베이션아카데미 멘토단)

이 문서는 신입, 이직시 새로운 환경과 도메인에서 빠르게 코드를 분석해 나가는 과정에 목적이 있습니다.

많은 개발자 환경에 서로 다른 의견이 있을 수 있습니다.

본문서는 인터넷 서비스를 유지하고 , 개발을 진행하는 서비스업을 기준으로 제시합니다.

1. 개발 언어에 대한 기본 문법을 확인 합니다.
2. framework 및 코딩 컨벤션을 우선 확인 합니다.
    a. 기존 업무 시퀀스를 확인 합니다.
3. 코드 시퀀스와 업무 시퀀스를 맵핑 합니다.
4. 3번을 기준으로 개발코드 기준 시퀀스 다이어그램을 작성합니다.
5. 업무 진행시 수정 , 추가 , 삭제가 가장 많이 일어 나는 구간을 체크 합니다.
    a. 해당 부분을 코드 시퀀스 다이어 그램을 기준으로 분리 합니다.
6. 5번 진행시 서비스 전체에서 가장 기본이 되는 기준 BM(비즈니스모델) 을 확인 합니다.
7. 해당 시퀀스들을 기준으로 코드를 다시 맵핑을 해 봅니다.

1. 개발 언어에 대한 기본 문법을 확인 합니다.

처음 접하는 경우 언어의 경우 , 기본적인 변수 선언 그리고 타입에 대한 정의등이 다르기 때문에 개발 진행시 자기도 모르는 버그를 유발 할 수 있습니다.
특히 오토 캐스팅을 지원하는 언어와, 자료형에 다른 형태의 자료형을 함께 사용이 가능한 언어들은 개발자로 하여금 버그 포인트를 못찾고, 디버깅을 힘들게 하는 요소 중에 하나입니다.

2. Framework 및 코딩 컨벤션을 우선 확인 합니다.

a. framework => open source 기반
b. framework => 자체 개발 기반
c. 막 코딩

개발 코드들은 개발의 효율성을 기본적으로 가지고 가야 되기에 framework를 기본적으로 사용합니다.

레거시 코드는 오래 될 수록 자체 개발 된 모델링 기법을 가지고 있을 확률이 큽니다.

기본적으로 framework 의 구조에 대한 이해가 있다면, 오래된 자체 제작 된 framework를 빠르게 이해 할 수 있습니다. 하지만 주니어의 경우 경험치 부재로 기본 구조를 익히는 부분을 힘들어 하는 경우도 많습니다.
레거시 코드로 작성된 프로그램을 해석 하기 힘든 경우는 이외에 여러가지 이유가 많습니다.

framework 의 동작 방식을 이해하고 해당 작업을 위한 코딩 컨벤션을 확인하도록 합니다.
코딩컨벤션이 없이 작성된 레거시 코드들은 개발자 마다 다른 색을 가지고 있을 경우가 많습니다.
최대한 기본적인 부분들을 다양하지만 눈에 빠르게 익혀 두는것이 좋습니다.

open source 기반의 framework 의 경우 튜닝을 통해서 자사의 서비스에 맞추는 경우가 많습니다.

사수가 있다면 해당 부분에 대한 부분을 문의 해서 정리를 해 둡니다.
만약 사수 , 문서 등 공유 받을 수 없다면 기본 구조와 다른 표현 방법들을 빠르게 찾아서 config 부분을 확인 해야 합니다.

3. 코드 시퀀스와 업무 시퀀스를 맵핑 합니다.

주니어의 경우 레거시 시스템을 기준으로 돌아 가는 기본적인 서비스 구조를 먼저 파악해야 합니다.

a. 서비스 - 유저 시퀀스의 흐름에 따른 코드 매핑
b. 운영 - 유저의 서비스 사용후 남은 로그 기준으로 운영 방법에 대한 시퀀스
   대두분 어드민툴, 운영툴 이라고 부르는 내부적으로 존재하는 서비스 운영 툴에서 어떻게 시스템을 컨트롤 하는지를 확인 합니다.

업무 시퀀스를 확인하는 가장 빠른 방법은

a. 실제로 서비스를 해 보면서 구간마다 하나의 시퀀스르 정의하고 노트에 정리 합니다.
b. 사수에게 서비스, 운영에 대한 전체적인 흐름에 대한 내용을 인수 인계 받습니다.
c. a의 경우는 일반적인 서비스 회사는 가능합니다.
   b의 경우는 SI 처럼 내부에서만 할 수 있는 서비스들의 경우에 효율적입니다.

4. 3번을 기준으로 개발코드 기준 시퀀스 다이어그램을 작성합니다.

업무의 흐름에 기반한 시퀀스를 개발 코드와 매핑하는 작업을 합니다.
그리고 개발 코드의 컨트롤로, 모델, 뷰, 라이브러리, 상수, 공통 함수 등.. framework 형태의 모습으로 시퀀스 다이어그램을 작성합니다.

이런 문서가 있다면 디버깅 및 개발하는 속도가 아주 빨라 집니다. 하지만 경력이 쌓이게 되면 이런 문서가 없이도 가능하기 때문에 작성하는 부분을 생략하기도 합니다.
그렇기에 인수인계시, 작성되는 경우가 많고 신규 입사자들의 경우, 코드를 라인 디버깅을 해야 되는 경우가 많습니다.

5. 업무 진행시 수정 , 추가 , 삭제가 가장 많이 일어 나는 구간을 체크 합니다.

해당 부분을 코드 시퀀스 다이어 그램을 기준으로 분리 합니다.
시퀀스에서 가장 빈번이 일어나는 구간을 찾아서, 리팩토링 대상으로 지정합니다.

업무의 대부분은 작업한 부분에 대해서 다시 수정 요청이 들어 오는 경우가 많습니다.
기준 BM 이 크게 변하지 않고, 서서히 변경 되기 때문입니다.

변경이 많이 일어나는 부분을 간섭이 가장 적은 형태로 코드를 리팩토링 할 수 있도록 합니다.

6. 5번 진행시 서비스 전체에서 가장 기본이 되는 기준 BM을 확인 합니다.

서비스 전체에 흐름을 기준으로 하는 스캘레톤 코드들 이 있습니다.
기본적으로는 framework 가 구조는 잡고 있지만, C 언어의 메인 처럼 기준이 되는 MVC 스켈레톤 코들를 파악 합니다.

많은 코드들은 그 기준 BM 코드를 기준으로 발전하게 되어 있습니다.

기준 코드를 중심으로 확장 코드들 확인 후 다음과 같은 부분들을 찾아서 확인해 봅니다.
서비스 업데이트 시 발생하는 많은 부분들입니다.

a. 기획의도
b. 기획서
c. 디자인

d. UX, UI
e. 운영데이터
f. CS 처리 시퀀스
g. 각 파트별 로그
h. 신규 업데이트를 위한 데이터
i. 리팩토링

이후 신규 서비스 진행시 위의 경우들을 함께 예측 해 봅니다.

7. 해당 시퀀스들을 기준으로 코드를 다시 맵핑을 해 봅니다.

기획서를 기준으로 새롭게 생성된 코드들을 6번에서 나열한 9가지 기준으로 코드들을 확인해 보고 실제 시퀀스에 맞게 맵핑을 해 봅니다.
신규 서비스들은 이러한 기존 틀 위에서 조금씩 확장하는 구조로 개발이 이루어 지게 되어 있습니다.
일반적인 회사들은
1년단위의 큰 틀에서 업무가 반복적인 패턴을 가지고 있습니다.
그렇기때문에 서비스 개발 부분에서도 비슷한 패턴을 가지게 됩니다.

규모가 큰 회사일수록 각 파트별 팀단위로 구분이 되어 있으며, 팀단위 협업을 원활히 하기 위한 많은 룰이 있을것입니다.

이 룰 안에서 업무를 효율적으로 하기 위한
메일 쓰기, 이슈트레킹 작성, 코드 리뷰…
개발 외에 글쓰기가 아주 많습니다.

이런 부분들을 위해서도 글을 잘 다듬고 전달 하고자 하는 내용을 심플하고 명확히 작성하는 방법을 배워야 합니다.