

Process Analyzing을 통한 System Software의 구성

단국대학교 모바일시스템공학 32154231 정영환

단국대학교 모바일시스템공학 32161681 박소빈

Develop Contents

1. Intro

2. Body

A. What is Software?

B. What is Program?

C. What is Program configuration?

D. Items of development

- i. What is the demand?
- ii. What do developers need?
- iii. What IDE will be used?
- iv. What procedures will be used?

E. Actual Development

F. Problem Debugging

G. Building programs through packaging

3. Ref

INTRO

우리가 사용하고 있는 높은 수준에 응용프로그램이나 어플리케이션은 넓은 의미에서는 운영체제에서 실행되는 모든 소프트웨어를 의미한다. 따라서 워드나 스프레드시트, 웹브라우저를 포함한 컴파일러나 링커 등도 응용프로그램으로서 기능을 한다. 하지만 여기서 언급하는 좁은 의미의 SW는 OS위에서 직접적으로 동작하는 소프트웨어 프로그램을 의미한다면 컴파일러나 링커 같은 LOW level 단의 시스템 소프트웨어를 제외한 응용프로그램으로 한정한다.

이렇게 한정할 경우 응용소프트웨어는 시스템 소프트웨어의 여집합이라고 생각할 수 있다. 이를 우리는 Application 혹은 App이라고 부르기로 한다.

-응용프로그램 제품군-

응용프로그램 제품군은 하나의 꾸러미로 묶인 여러 개의 응용프로그램을 말한다. 단적인 예를 들면 워드와 PPT를 포함한 마이크로소프트 오피스 프로그램을 예로 들 수 있겠다.

하나의 꾸러미로 묶인 응용프로그램들은 대개 사용자가 사용하기 쉽게 소프트웨어간 호환성이 높고 높은 수준의 유저 인터페이스를 제공한다. 이와 함께 편의를 위해 응용프로그램 사이에 상호작용하는 기능까지도 포함한다.

예를 들어 워드 파일에 ppt를 참조할 수 있고 ppt의 결과가 스프레드 시트에 탑재될 수 있음을 의미한다.

우리가 이번에 진행하는 절차기반의 시스템 소프트웨어의 구성은 사용자의 요구를 파악하며 Power Measurement 장비에서 측정되는 계측값을 재 구성하고 LTE Band의 채널적인 요소와 가드밴드를 바탕으로 직접적인 ROW data를 특정하여 파일간 호환성 + 유저간 편의성 + 손쉬운 접근성을 보장하는 응용프로그램을 개발하는데 의의를 둔다.

Body

1. What is Software?

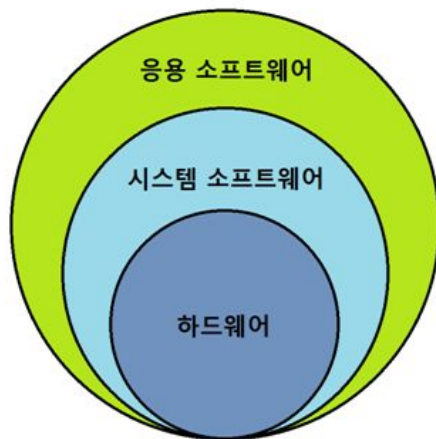
컴퓨터의 시스템을 구성하는 주요 요소 중 하나이다.

(컴퓨터 = 하드웨어 + 소프트웨어 + 펌웨어)

데이터 등 무형의 구성요소를 가리키며 비트 단위로 연산되는 과정을 거쳐 일련의 Process를 처리하는데 목적을 둔다.

상단에도 언급했다시피 소프트웨어는 시스템소프트웨어와 응용소프트웨어로 나뉘어지고 이 문서에서 다루는 주된 내용은 응용 소프트웨어에 한정된다.

그런의미에서 소프트웨어 개발은 사용자의 요구에 맞추어 사용자의 편의를 고려한 목적지향성 절차를 자동화 하는 과정이라고 볼수 있다.



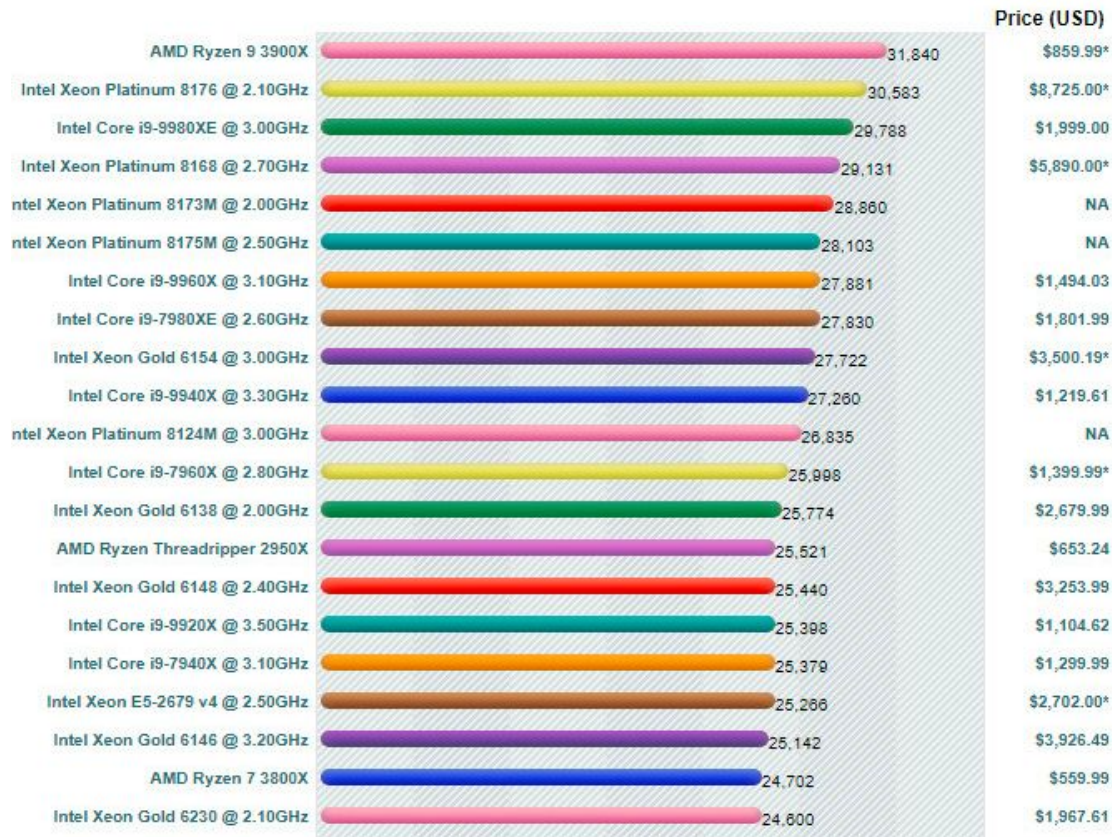
2. What is Program?

프로그램이란 컴퓨터를 활용해 어떠한 결과를 얻기 위해 차례대로 작성된 논리적인 기계라고 볼 수 있다. 기계를 작동시키면 설계된 대로 작동하는 것처럼 프로그램도 마찬가지로 수동적으로 적혀진 명령의 집합과 그 dependencies를 바탕으로 작동하는 바이다.

다만 그 과정이 CPU 클럭과 연관 혹은 비동기식으로 처리되어 초당 수천만번의 연산 속도를 보여주어 자동으로 보이는 것에 불과하다.

그렇다면 이를 바탕으로 생각하면 프로그램의 성능은 CPU 조금 더 넓게 말하면 컴퓨터 자원의 할당과 프로세서의 성능에 의해 좌우된다고 말해도 과언이 아니겠다.

PassMark - CPU Mark
High End CPUs - Updated 1st of August 2019



논외로 CPU란 중앙처리장치 Central Processing Unit의 줄임말로 모든 데이터 처리의 핵심에 서있다.
이러한 CPU의 성능은 일반적으로 4가지에 의해 결정된다.

1. 아키텍처
2. 클럭
3. 코어 수
4. 캐시메모리

이러한 요소들의 집합으로 구성된 CPU는 상단 그림에서도 알 수 있다시피 성능에 따른 편차가 크기 때문에 같은 응용프로그램이 더라도 환경에 따라 다른 성능을 보여줄 수 있다는 것을 염두에 두어야 한다.

해당 시스템을 디자인한 환경이다

프로세서: Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz (4 CPUs), ~2.3GHz
메모리: 8192MB RAM
페이지 파일: 6471MB 사용됨, 2879MB 사용 가능
DirectX 버전: DirectX 12

<https://ark.intel.com/content/www/kr/ko/ark/products/85213/intel-core-i5-5300u-processor-3m-cache-up-to-2-90-ghz.html>

해당 링크에서 구동 환경에 대한 자세한 정보를 알 수 있으며 이보다 나은 환경에서 구동되는 응용 프로그램은 더 빠르고 좋은 퍼포먼스를 보여줄 수 있음을 명시한다.

3. What is Program Configuration?

여기서 '구성'이란 어떻게 목표한 바에 가장 부합하도록 프로젝트를 수행할 수 있는가에 대한 의사결정을 의미한다. 우리는 깔끔하고도 효율적인 코드라는 'python'의 특성을 극대화할 수 있는 방법을 고민해야 할 필요가 있다. 일반적으로 '구성'이란 로직과 의존성이 깔끔한 코드를 만드는 것을 의미할 뿐만 아니라 파일시스템에 어떻게 파일과 폴더를 구성해서 접근하기 쉽고 유연성이 있으면서도 빠른 결과물을 산출할 수 있는가에 초점을 맞추도록 한다.

어느 모듈에 어느 기능이 들어가야 하는가? 는 가장 중요한 문제이다. 프로젝트에서 데이터는 어떻게 흘러가야 하는가? 는 더더욱 어려운 문제이며 어떤 특징과 기능은 통합되어야 하고 어떤 기능은 분리되어 작업되어야 하는가? 그리고 방금한 그 작업이 프로그램의 성능에 어떤 영향을 주는가? 에 관해서도 고려할 수 있어야 한다. 이러한 질문에 앞으로 하나씩 답을 하면서 우리가 만들어야 할 최종적인 프로그램의 뼈대를 구성하고 구성한 결과를 바탕으로 제품에 대한 평가를 내릴 가이드 라인을 제시할 수 있게 된다.

이하 알파벳 인덱스는 해당 프로그램을 구성하면서 고려한 요소들이며 이로 말미암아 원활하고 구조감 있는 프로젝트 진행에 도움이 되었다.

A. 저장소의 구조

중요한 문제이다. 코딩스타일, API, 디자인, 자동화같이 건강한 개발 사이클에 필수적인 요소들처럼 저장소의 구조도 프로젝트의 아키텍처에 결정적인 요소로 작용한다.

정확한 위치를 선언해 주고 선언된 장소를 기반으로 체계적이고 논리적인 데이터 접근은 마치 DATABASE의 SQL선언문처럼 논리적인 작업을 통해 결과를 도출하고 도출된 결과를 바탕으로 확실히 가시적인 결과물을 만들어 내는데 도움을 줄 수 있다.

요약하자면 파일의 종류를 분류하고 집적하되 체계적으로 정리하고 논리적으로

구성하는것에 의의를 둔다.

B. 코드 구성

파이썬의 임포트 방식과 잘 만들어진 모듈은 프로그램 개발이 원활하게 돕는 역할을 하여 더욱 원활한 프로젝트 진행을 할 수 있게 한다.

여기서 말하는 '쉽다' 라는 말은 쉽게 엉망이 되기도 한다는 것을 의미한다.

코드를 가독성 있게 작성하는데 피해야할 몇가지 요소가 존재한다.

- i. 복잡한 순환참조
- ii. 숨겨진 링크연결
- iii. 전역구문의 과도한 사용
- iiii. 스파게티 코드
- v. 라비올리코드

그런의미에서 코드가 갖는 복잡성을 단순화 시키고 의존성을 줄이면서 몇가지 요소들을 결합해 정확한 목표를 구현하는것에 목적을 두는것으로 코드 구성을 명료화 한다.

C. 모듈

Python의 모듈은 사용가능한 추상레이어중 하나로 동작한다. 이는 코드를 기능파트와 저장파트로 나눌수 있도록 해준다.

예를 들어 프로젝트의 레이어중 하나는 사용자 인터페이스를 담당하고 다른하나는 저 수준의 데이터 처리를 담당한다. 이렇게 레이어를 분리시키는 방법은 인터페이스기능과 저 수준의 데이터 처리를 가능하게 하는 파일을 세분화 하여 분리 하는것으로부터 시작한다.

이를 `import ...fromimport....` 형태로 구분하며 서로간에 의존성을 상충시킨다. 이는 `import` 구문에 즉시 사용 가능한 `import sys`와 `import os`를 포함한 서드파티 모듈을 의미한다.

이러한 모듈은 타 코드와는 다른 특징을 보이는데 타 언어에서 의미하는 `include file` 형태의 지시문은 전처리 장치 즉 `preprocessor`가 해당 파일의 모든 코드를 가져와 호출자에서 복사해 사용하는 방식으로 진행된다면 python에서는 모듈에 포함된 코드를 `namespace`에서 독립적으로 실행하는 것으로 구분된다. 이는 전체 코드가 오작동하는 경우를 미연에 방지하고 원활한 데이터 처리를 가능하게 함을 의미한다.

D. 패키지

파이썬은 패키징 하는것에 있어서 간단하고 명료하게 작동한다. 이는 파이썬의 모듈 구조를 단순하게 디렉토리 형태로 상충화 시킨것으로 인식 할 수 있다.

`__init__.py`가 있는 모든 디렉토리는 파이썬 패키지로 분류되며 여러 다른 모듈과 비슷하게 불러와 지며 구성되고 재 생산된다.

이 파일들은 패키지 전체의 모든 정의를 모으는 용도로 사용된다.

E. 객체지향 프로그래밍

python에서 모든것을 객체로 분류된다. 그리고 객체처럼 다룰수가 있다. 이말은 함수는 일급객체라고 선언하는것과 같은 의미를 가진다. 클래스 문자열 심지어 타입까지도 파이썬에서는 객체로 취급된다. 다른 객체 처럼 타입이 있고 함수의 인자 값을 받을수도 있다. 메소드와 속성을 포함 할수도 있다. 이러한 점을 고려하여 파이썬을 객체 지향 프로그래밍이라고 하는 것은 옳다.

하지만 자바와 달리 파이썬은 객체 지향 프로그래밍을 프로그램 패러다임으로 두지 않는다.

따라서 클래스를 정의하지 않아도 실행이 가능한 수준에 이른다.

게다가 모듈에서도 살펴보았듯이 파이썬이 모듈과 namespace를 분리 하여 다루는 방식은 개발자로 하여금 자연스럽게 캡슐화와 추상레이어의 분리를 가능하게 해준다.

따라서 클래스의 생성에 집착하지 않아도 되는 편안한 환경을 조성한다.

어떤 아키텍처,어플리케이션에서는 파이썬 프로세스에 동시에 발생하는 외부 요청에 응답하기 위해 복수의 인스턴스가 발생하거나 구문의 실행이 멈춘 상태로 정지 되어 있는 경우도 있다. 이 경우에는 고정된 정보를 계속 붙잡고 있을 필요성이 있을때 사용된다.

이러한 문제를 피하기 위해서는 모호한 문맥과 부작용을 최소화 하는 함수를 사용하는것이 좋고 함수 내부에 있는 persistant 레이어 항목을 남용하면 함수의 의미가 모호해져 맥락의 변화를 일으키게 되므로 저장된 값을 일정하게 유지하는것이 중요하다.

F. 동적타이핑

파이썬은 동적 타이핑이 되는 언어라고 불린다. 이 말은 변수가 고정된 타입을 가지고 있지 않으며 다른 정적타입의 변수들과는 다른 속성을 보여주게 된다.

어떤 태그나 이름이 객체에 쓰여지면 메모리의 한 조각이 아니기 Eo문에 함수가 될수도 있고 str이 될수도 있으며 int가 될수도 있다.

이런 동적 타이핑은 약점으로 여겨지기도 하며 디버깅이 어려워 지는 문제를 초래 하기도 한다.

이외에 시스템에 대한 정보나 python의 개발 사항에 대한 정보는 아래 링크에서 추가적으로 확인 가능하다.

<http://docs.python.org/2/library/>

<http://www.diveintopython.net/toc/index.html>

4. Items of Demand

A. What is the Demand?

- 1> Band 별 Power Data의 측정 및 정리
- 2> 정리된 데이터의 이분화 해소
- 3> Row 형태의 데이터가 갖는 불규칙성 해소
- 4> 데이터 정제 및 재배열
- 5> 새로운 Form으로 재구성

+++++

및 업무 절차의 간소화 및 오류 가능성의 종식 요구

+++++

업무 자동화를 통한 작업 시간 단축 요구 + 능률향상

B. What the Developers need?

- 1>IDE
- 2>소요시간 및 일정
- 3>예외처리 요구사항
- 4>통합 파일의 처리 및 구동 환경의 일치 여부

C. What IDE will be used?

ATOM

자유 오픈소스형태의 OS 리눅스 윈도우용 문서 편집기 및 코드 편집기로서 NODE.JS로 작성된 플러그인 ,깃허브가 개발한 임베디드 깃 관리지원을 포함한다. 대부분의 확장패키지는 자유소프트웨어 라이선스를 취하며 커뮤니티가 만들고 관리하고 있다. 아톰은 크로미엄에 기반을 두며 커피스트립으로 작성되어 있어 IDE로 사용이 가능한 형태 이다.

2015.6.25 정식출시

ref . <https://flight-manual.atom.io/>

python

1991년 프로그램 귀도 반 로섬이 발표한 고급프로그래밍언어로 플랫폼에 독립적이며 인터프리터식 객체지향적 동적 타이핑을 지원하는 대화형언어이다.

파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형,공동체 기반 개발모델을 가지고 있으며 C언어로 구현된 C파이썬이 표준으로 매김한다.

ref. <https://docs.python.org/3/>

pandas

pandas는 파이썬으로 작성된 데이터를 분석 및 조직하기 위한 소프트웨어 라이브러리 이다. 수치형태이بل과 시계열 데이터를 조직하고 운영하기위한 데이터를 제공하는데 3조항 BSD 라이선스 조건하에서 무료로 이용이 가능하다. R에서 사용되던 data.frame형태를 차용해 DataFrame이라는 구조를 활용하고 타 플랫폼에서 사용되던 데이터 구조를 무리 없이 변환하고 적용 가능한 형태로 변환시키는데 유리하다.

특징)

- 통합 인덱싱을 활용한 데이터 조작을 가능하게 하는 데이터프레임(DataFrame) 오브젝트
- 인메모리(in-memory) 데이터 구조와 다양한 파일 포맷들 간의 데이터 읽기/쓰기 환경 지원
- 데이터 결측치의 정렬 및 처리
- 데이터셋의 재구조화 및 피보팅
- 레이블 기반의 슬라이싱, 잘 지원된 인덱싱, 대용량 데이터셋에 대한 서브셋 지원
- 데이터 구조의 칼럼 추가 및 삭제
- 데이터셋의 분할-적용-병합을 통한 GroupBy 엔진 지원
- 데이터셋 병합(merging) 및 조인(joining) 지원
- 저차원 데이터에서의 고차원 데이터 처리를 위한 계층적 축 인덱싱 지원
- date range, 빈도 변환, 이동 창 통계, 이동 창 선형회귀, 날짜 이동 등의 시계열 작업 지원
- 데이터 필터 지원

판다스 라이브러리의 주요 코드는 Cython이나 C로 작성되었으며, 퍼포먼스에 최적화되어있다.

ref . https://pandas.pydata.org/docs/user_guide/index.html

openpyxl

openpyxl은 Excel2010 파일을 읽고 쓰는 python 라이브러리이다.

openpyxl로서 PHPExcel 팀에 대한 모든 쿠도는 처음에 PHPExcel에 기반을 두고 있다.

ref. <https://openpyxl.readthedocs.io/en/stable/>

PyQt5

Qt는 현대 데스크 탑과 모바일 시스템의 많은 측면에 접근하기 위한 고도의 API를 구현하는 교차 플랫폼 C++ 라이브러리의 집합이다. 여기에는 위치 및 위치

지정 서비스 ,멀티 미디어 ,근거리 무선통신 및 블루투스 연결 + 크롬기반 웹브라우저 그리고 기존의 UI 개발이 포함된다.

PyQt5에서는 기존에 포함된 Qt의 포괄적인 파이썬의 바인딩 집합이다.
35개 이상의 확장 모듈로 구성되며 모든 플랫폼에서 파이썬을 C++로 대체 응용 개발 언어로 사용할 수 있다.

또한 PyQt5에서는 C++기반 어플리 케이션에 내장되어 해당 어플의 사용자가 기능을 구성하고 향상시킬 수 있게 설정된다.

ref. <https://pypi.org/project/PyQt5/>

D. What procedures will be used?

소프트웨어 생명주기 (software life cycle)

소프트 웨어를 체계적으로 개발하고 관리 하기 위해 개발과정을 단계별로 구분

1. 요구사항 분석

- 문제 분석 단계
- 개발할 소프트웨어의 **기능**과 **제약조건**, **목표** 등을 소프트웨어 사용자와 함께 명확히 정의
- 개발할 소프트웨어의 성격을 정확히 이해하고, 개발 방법과 필요한 개발 자원 및 예산을 예측
- **요구명세서** 작성

2. 시스템 명세

- 시스템이 무엇을 수행해야 하는가를 정의하는 단계
- **입력자료**, **처리내용**, 생성되는 **출력**이 무엇인지를 정의
- **시스템 기능 명세서** 작성

3. 설계

- 시스템 명세 단계에서 정의한 기능을 실제로 수행하기 위한 방법을 논리적으로 결정하는 단계
- 시스템 구조 설계 ; 시스템을 구성하는 내부 프로그램이나 모듈 간의 관계와 구조 설계
- 프로그램 설계 ; 프로그램 내의 각 모듈에서의 처리 절차나 알고리즘을 설계
- 사용자 인터페이스 설계 ; 사용자가 시스템을 사용하기 위해 보여지는 부분 설계

4. 프로그래밍

- 설계 단계에서 논리적으로 결정한 문제 해결 방법(알고리즘)을 프로그래밍 언어를 사용하여 실제 프로그램을 작성하는 단계
ex) 사용할 언어 선택, 프로그래밍 기법과 스타일, 프로그래밍 순서
- 프로그래밍 기법
ex) 구조화 프로그래밍, 모듈러 프로그래밍

5. 테스트

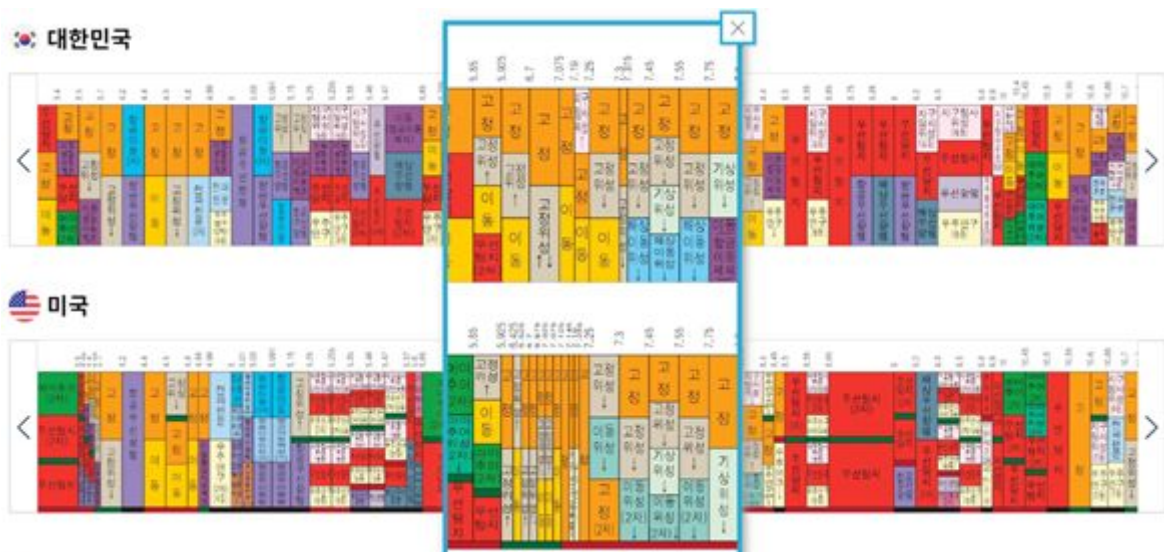
- 개발한 시스템이 요구사항을 만족하는지, 실행결과가 예상한 결과와 정확하게 맞는지를 검사하고, 평가하는 일련의 과정
- 숨어있는 오류를 최대한 찾아내어 시스템 완성도를 높이는 단계
- 1단계 : 단위 테스트(Unit Test), 2단계 : 통합 테스트(Integration Test), 3단계 : 인수 테스트

6. 유지보수

- 시스템이 인수되고, 설치된 후 일어나는 모든 활동
- 프로그램 오류 수정, 시스템 디자인 수정, 새로운 요구사항 추가, 시스템 사용환경 변화에 대한 교정 등

5. Actual Programming

주파수는 국가 재산이고 이를 분할해 통신 사업자 및 국가 기관에 대여 하는 형태로 통신사업은 출발한다. 모든 국가가 일정 주파수에 대해 반드시 같은 목적으로만 사용하는것이 아니다.



ref. <https://spectrummap.kr/radioInfo/radioDivideDiagram.do?menuNo=300499#>

이 시점에서 Row Data를 살펴보고 실제 데이터가 어떠한 형태로 이루어져 있는지 그리고 그 데이터가 무엇을 의미하는지 알아야 실질적인 코딩을 하는데 있어 도움이 될것이다.

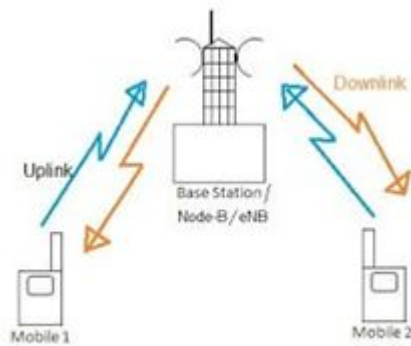
LTE B2 , B3, B41....이란 무엇인가?

실질 적인 LTE가 할당된 주파수를 나누고 채널 단위로 분할해서 목적에 따라 넘버링한 분포를 의미한다.

따라서 실제 코딩에는 해당 밴드 묶음 B2등에 들어가서 그 값을 채워 넣는 식으로 진행된다.

<LTE의 종류는 다양하다.>

Tx , Rx간의 통신 기준에 따라



Duplex : Tx와 Rx간에 쌍방이 동시에 데이터를 주고 받는것을 의미한다.

simplex : Tx에서 Rx로만 데이터를 전송하거나 그 반대

Half-Duplex : Tx에서 Rx로 전송하고 있으면 Rx는 반대 전송을 하지 못함

주파수에 따라

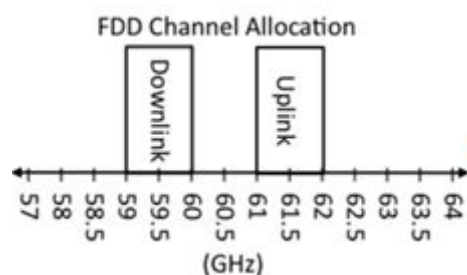
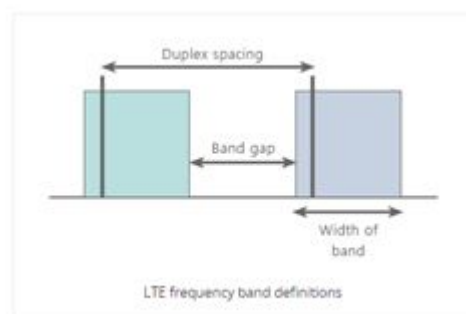
FDD(Frequency Division Duplex)

uplink와 downlink에 사용되는 주파수 대역대가 다르다.

+

동시에 device->BS/BS->device 통신 가능 (duplex)

FDD LTE bands: FDD spectrum requires pair bands, one of the uplink and one for the downlink. It is also important that there is sufficient spacing between the top of the lower band and the bottom of the upper band to allow sufficient filtering. Also the uplink to downlink channel spacing must be sufficient to allow sufficient filtering to prevent the transmitted signal from entering the receiver and desensitising it.



일반적으로 낮은 대역폭을 downlink로 높은 대역을 uplink로 사용한다.

주로 우리가 사용하는 device는 데이터를 전송보다는 다운을 많이 받는 형태를 가지기 때문에 높은 주파수 대역일수록 pathloss가 크고 멀리까지 신호가 가지 못하며 멀리 보내기 위해서는 높은 파워를 출력해야만 하는 이유에 있다.

그래서 제조사에서 정하는 일정한 파워의 범위가 존재하는 이유가 여기에 있다.

LTE 1~22 까지는 FDD방식으로 통신는데 사용되고 LTE33~41은 TDD방식으로

사용된다.

TDD(Time Division Duplex)

Uplink와 Downlink에 사용되는 주파수 대역이 같다 . 다만 시간 차이를 두고 서로 데이터를 전송 하는 개념이다.

따라서 스케줄링 방식을 시간에 맞춘다.

LTE radio channel bandwidths [1.4MHz, 3,5,10,15,20]

LTE는 OFDMA 방식이다. 이는 각 유저에게 시간축 주파수 축을 고려해서 할당을 하게 되는데 이때 RB(Resource Block)라는 최소 보낼수 있는 데이터 양을 도입한다.

일반적으로 LTE의 RB는 12개의 subcarrier(각 15kHz ; 총 약 180kHz)를 7개의 OFDM Symbol (시간 상= 0.5ms 의미)를 의미한다.

하나의 user가 OFDMA 방식에서 할당받을 수 있는 주파수와 시간의 최소 단위를 RB라고 하는데 LTE에서는 1 RB = 12 sub carrier, 0.5ms 를 의미한다.

하지만 실질적으로는 0.5ms 마다 다른 유저를 할당하는게 제조사 측에서 어렵기 때문에 통상적으로는 14 OFDM symbol(1ms)를 1RB라고 한다.

LTE의 경우 채널 bandwidth가 1.4 3 5 10 15 20 으로 다양하기때문에 더 많은 RB의 개수를 갖게 된다.

RADIO CHANNEL BANDWIDTHS SPECIFIED IN LTE	
CHANNEL BANDWIDTH	NUMBER OF RESOURCE BLOCKS
1.4 MHz	6
3 MHz	15
5 MHz	25
10 MHz	50
15 MHz	75
20 MHz	100

따라서 각 측정된 데이터들이 어떤 채널에서 어떤 통신 방식으로 어떤 복변조 값을 가지고 어떤 모드에서 작동하는지를 파악하고 그 결과 값을 데이터 테이블 형태로 정리 하는 과정을 의미할수 있다.

A. Observation of Row data

,1,12,20.50998,24.17917,6.322861,0.83,0.83,
 ,1,24,20.52176,24.2659,6.85215,0.83,0.83,
 ,12,0,20.71823,25.61649,7.667542,0.83,0.83,

데이터 들의 나열 형태로 기록된 Row데이터를 바탕으로 데이터 프레임을 만들어 틀 단위로 정형화 시켜 보자.

```
import sys, UI
import pandas as pd
import openpyxl as op
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon, QFont
from PyQt5.QtCore import QApplication, QDate, Qt
```

폴더 권한에 접근하기 위해서 sys와 해당 프로그램의 UI를 불러오기 위해 import된 외부 모듈 UI를 가져온다.

ID	Band	BW	FREQ	CHN	Mode	RBO	RBS	UE	Peak Pwr	FE	EAO	EAI	MPR
140	LTE41W	5MHz	0	39750	QPSK	1	0	20.53619	24.25912	7.839203	0.83	0.83	
141	LTE41W	5MHz	0	39750	QPSK	1	12	20.50998	24.17917	6.322861	0.83	0.83	
142	LTE41W	5MHz	0	39750	QPSK	1	24	20.52176	24.2659	6.85215	0.83	0.83	
143	LTE41W	5MHz	0	39750	QPSK	12	0	20.71823	25.61649	7.667542	0.83	0.83	
144	LTE41W	5MHz	0	39750	QPSK	12	6	20.70999	25.54453	8.869171	0.83	0.83	
145	LTE41W	5MHz	0	39750	QPSK	12	11	20.69659	25.55676	7.982254	0.83	0.83	
146	LTE41W	5MHz	0	39750	QPSK	25	0	20.72638	25.99982	9.799004	0.83	0.83	
147	LTE41W	5MHz	0	39750	16QAM	1	0	20.59857	25.05502	8.926392	0.83	0.83	
148	LTE41W	5MHz	0	39750	16QAM	1	12	20.6008	25.04013	5.679131	0.83	0.83	
149	LTE41W	5MHz	0	39750	16QAM	1	24	20.59885	25.07306	6.036758	0.83	0.83	
150	LTE41W	5MHz	0	39750	16QAM	12	0	20.75168	26.55835	7.867813	0.83	0.83	
151	LTE41W	5MHz	0	39750	16QAM	12	6	20.75436	26.42084	7.46727	0.83	0.83	

pandas의 데이터 프레임은 해당 데이터를 손쉽게 convert하고 읽을수 있도록 도와주는 역할을한다.

해당 데이터는 사내 정해진 규격의 양식의 올바른 위치에 자동으로 입력 되며 해당 값이 중복 되지 않고 적절한 오차 범위 내에 들어 오고 있는지 체크 하고 만약 벗어난다면 해당 값에 붉은 색으로 오류 처리를 발생시키는것을 목표로 한다.

실제로 여기서 사용할 데이터는 Band와 Band Width Channel RBO,RBS,UE 값이다.

해당 값외에 다른 값을 사용해서 정렬할 필요는 없다. 일반적인 파라미터에 해당하기도 하고 계측 보정치에 해당하기도 한다.

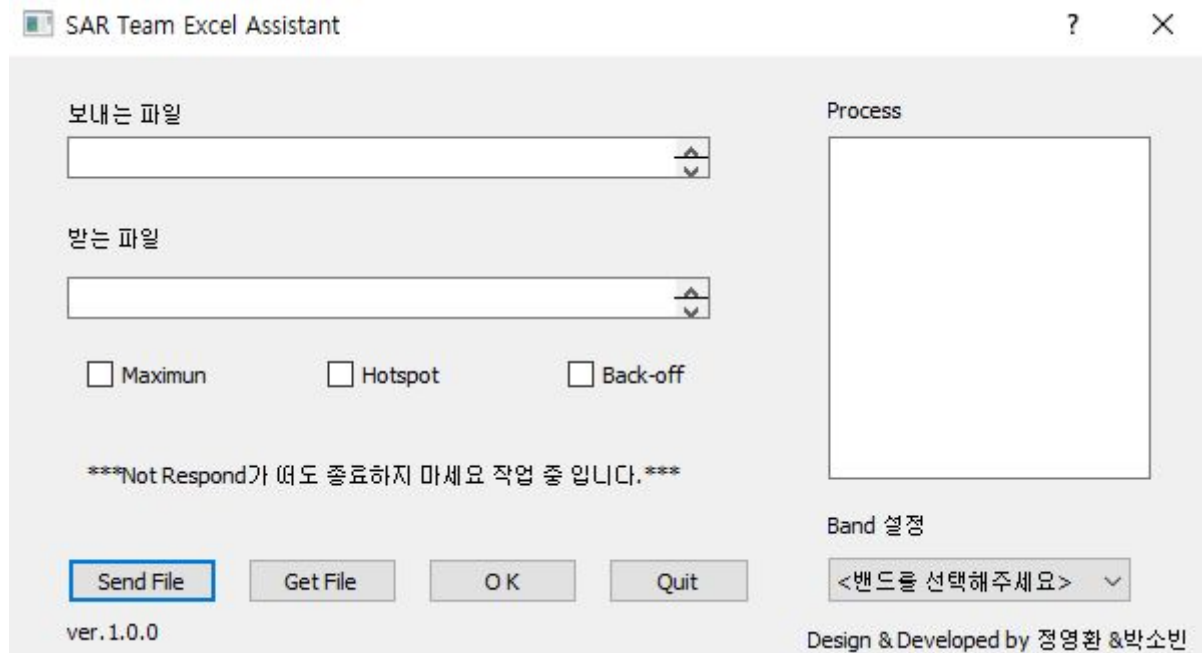
이는 technician 분들이 처리해주실 문제 이다.

B. Building UI

UI를 구성하는 과정에서는 몇가지 요점 사항이 있다.

- 1.사용자가 쉽게 사용할수 있어야 하고
2. 필요없는 기능이 없어야 하며
3. 진행상황이나 오류를 쉽게 판단할수 있어야 한다.

그런의미에서 Technician분들에게 필요한 부분과 불필요한 부분을 묻고 그 결과를 바탕으로 실질적인 업무에 사용될 UI를 구성한다.



UI를 구성하고 각 버튼에 함수를 할당해 주는 과정을 거친다.

.

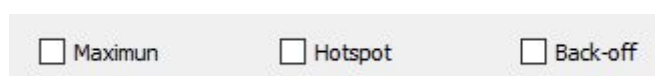
전체 프레임의 크기를 정하고

원하는 위치에 데이터가 들어갈수 있도록 시각화 하는 칸 들의 크기와 방향을 정해준다.

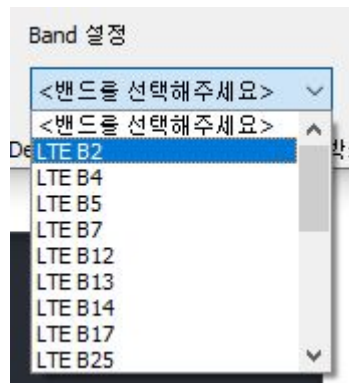
그와 함께 버튼의 방향성을 구성하고 전체 절차의 진행 정도를 파악할수 있는 창을 구성했다.



처럼 박스 형태로 sys가 접근할 폴더의 이름을 str형태로 받을수 있게끔 그리고 시각화 할수 있게끔 박스를 구성했고



기계가 측정해서 보여주지 못한 파라미터에 관해서 보정해주는 체크 박스를 만들었으며



콤보 박스를 만들어 원하는 band를 체크할수 있도록 select 할수 있게 편의를 제공했다.

이러한 일련의 과정이 없다면 개개인이 일일이 Row data를 관찰하고 그 결과를 폼 형태에 복사해 붙여 넣는 고루한 작업을 계속하게 된다.

C. 코드 구성

```
class MyApp(UI.Ui_Dialog, QDialog):
    global send_name
    global get_name

    global Max_power
    global Hotspot_power
    global back_off_power

    #####
```

큰틀의 My App을 선언해 주고 내부적으로 전역으로 사용되는 변수를 할당 해준다.

```
# 이니셜라이즈 및 셋업 과정
def __init__(self):
    super().__init__()
    QDialog.__init__(self, None, Qt.WindowStaysOnTopH
    self.setupUi(self)
    self.date = QDate.currentDate()
    self.initUI()
```

그 후로는 initialize 하고 기본 셋업 과 동시에 UI를 적용하고

면 출력에 관해서

```
def initUI(self):
    QToolTip.setFont(QFont('SansSerif', 10))
    self.center()

    #버튼과 ui 상호작용
    self.send_file.clicked.connect(self.OnOpenDocument)#
    self.send_file.setToolTip('파일 불러오기 입니다.')

    self.get_file.clicked.connect(self.OnOpenDocument2)
    self.get_file.setToolTip('파일 불러오기 입니다.')

    self.quit.clicked.connect(QCoreApplication.instance(
    self.quit.setToolTip('종료하기')

    self.ok.clicked.connect(self.okFunction)
    self.ok.setToolTip('확인')

    #체크박스
    self.max_chk.stateChanged.connect(self.chkFunction)
    self.hot_chk.stateChanged.connect(self.chkFunction)
    self.back_chk.stateChanged.connect(self.chkFunction)
    #콤보박스
    self.combo.activated[str].connect(self.comboEvent)

#####
```

위처럼 콤보 박스와 버튼 그리고 텍스트 에디터 등에 함수를 연결 시켜 주는 작업을 한다.

결과적으로는 간단한 동작에 불과 하지만 내부적으로는 메모리 접근이나 배열의 주소 순서상 문제등 많은 것들을 고려하여 함수를 빌드 해야 한다.

```
def okFunction(self):
    global Row
    send = pd.read_csv(send_name)
    target = op.load_workbook(get_name)
    Row=len(send.index)
    sheet = target[Band]
    sheet_row = len(sheet['A'])
    print('time start')
    time_a = time.time()
    r문은 1번만 사용할것
```

핵심코드의 일부인 okFunction에서는 ok버튼을 누르면 파이썬 내부적으로 일어나는 여러가지 절차들을 정의해 두었다.

읽어온 csv 파일을 데이터 프레임 형태로 저장 받고 엑셀 파일과 연동 및 저장을 하는 과정을 담는다.

```

for i in range(Row):
    나 데이터 프레임을 만들면 메모리 낭비가 심함
    x3 = send[' R80'][i]
    x4 = send[' R85'][i]
    x5 = send[' CHN'][i]
    self.textBox.append('Processing....'+str(i)+'\n')

    if (Band == 'LTE B41') or (Band == 'LTE B41(HPUE)') or (Band == 'LTE B41(IC)'):
        if (Max_power == 1) and (Hotspot_power == 0) and (back_off_power == 0):
            if x5 == 39750:
                y5 = 'E'
            elif x5 == 40185:
                y5 = 'F'
            elif x5 == 40620:
                y5 = 'G'
            elif x5 == 41055:
                y5 = 'H'
            elif x5 == 41490:
                y5 = 'I'

```

이렇게 들어온 값을중에 핵심변수를 추리고 그 데이터드을 비교 하면서 모드를 설정하고 설정된 모드를 기반으로 채널을 할당해 변수를 조율한다.

```

#시작 값을 정하고
#이게 각 밴드마다 전부 다름 그러니까 위치를 하나하나 따는게 좋음
if send[' BW'][i]== ' 5MHz ':
    st = 105

elif send[' BW'][i]== ' 10MHz ':
    st = 74

elif send[' BW'][i]== ' 15MHz ':
    st = 43

elif send[' BW'][i]== ' 20MHz ':
    st = 12

```

들어갈 변수에 밴드 단위를 설정하고 그 값이 엑셀에서 유의미한 위치에 있어야 하므로 그 위치를 설정해 준다.

```
#여기서 부터 모드별로 시작과 끝을 정함
if send[' Mode'][i]== ' QPSK ':
    fin=st+6
elif send[' Mode'][i]== ' 16QAM ':
    st = st + 7
    fin= st + 6
elif send[' Mode'][i]== ' 64QAM ':
    st = st + 14
    fin= st + 6
elif send[' Mode'][i]== ' 256QAM ':
    st = st + 21
    fin= st + 6
```

각 모드는 복변조 방식이 다르기 때문에 그 모드에 따라 시작위치와 끝을 정해주고

```
#결측값 보정 해주는 코드 임 다른 밴드에서 고장난 부분이 있으면 주
for j in range(st,fin+1):
    if send[' BW'][i]== ' 5MHz ' and x3==12 and x4 == 6:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 5MHz ' and x3==12 and x4==11:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-2
    elif send[' BW'][i]== ' 10MHz ' and x3==1 and x4==24:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 10MHz ' and x3==25 and x4==24:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-1
    elif send[' BW'][i]== ' 15MHz ' and x3==36 and x4==18:
        y3 = sheet['C'][j].value
        y4 = sheet['D'][j].value-2
    elif send[' BW'][i]== ' 15MHz ' and x3==36 and x4==35:
```

장비에서 나오는 값이 실제 계측치와 다른것을 인식해 그 위치를 보정해주는 코드를 작성해 결측값이 생기지 않게 한다.

수많은 데이터에서 의미를 갖는 값은 많지 않고 그 값들을 순서쌍 형태로 묶어 한번에 처리를 하는 과정을 통해 데이터 구조를 더 자유롭게 사용할수 있다.

6. Problem Debugging

1. 계측치가 원하는 위치에 안들어가는 문제

예외처리 부족으로 인해 몇가지 계측치가 누락되거나 중복되는 현상이 발견되어 해당 열에 대한 인덱스를 조정해주고 그 값을 지역변수 형태로 저장하여 다른 함수에서 접근하지 못하게 만들

2. 패키징을 하면 프로그램이 느려지는 문제

pyinstaller로 패키징을 했을때 전반적인 코드의 실행속도에 문제가 생김

1>Big O 연산에 의한 이중 중첩 연산량 문제

for 문으로 서치할 문제를 손코딩으로 1:1 매칭 시켜 중복성을 디버깅

2>python의 모듈 import 과정에서 메모리 적재 문제 발생

모듈을 최소화 시키고 데이터 프레임을 삭제해 메모리 누수를 지움

3>python의 단일 thread GIL로 인한 속도 저하문제

다중 쓰레드를 활용한 문제 강구(진행중)

3. 연산수가 많아 출력값이 print되는데 시간이 오래 걸리는 문제


개별연산의 개수를 줄이기 위해서 for 대신 if를 사용해 코드수를 늘리고 노동 집약적인 코드 구현


4. 측정하지 않는 채널에 관해서 장비에서 계측치가 나오는 관계로 값을 무시해 주어야 하는 문제

실측 값은 존재 하지만 테이블 상에는 누락되어야 하는 문제 이므로 해당 값에 continue를 활용해서 다시 돌게끔 조정

7. Building Program through packaging

pyinstaller를 활용한 일괄적인 onefile packaging을 구현

 program.exe

 응용 프로그램

Ref.

L. Benini, A. Bogliolo, S. Cavallucci and B. Ricco, "Monitoring system activity for OS-directed dynamic power management," *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379)*, Monterey, CA, USA, 1998, pp. 185-190.

Wes Mckinney, Julie Steele and Meghan Blanchette, Teresa Exley, "Python for Data Analysis" proceedings 2012.Oct O'REILLY pp.111-152

Vivian Siahaan-Rismon Hasiholan Sianipar "LEARNING PyQt5", first-edition proceedings 2019 MY_SQL ,pp 89-221

T. E. Oliphant, "Python for Scientific Computing," in *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10-20, May-June 2007.