CA Combination and Reconfiguration Automation Process

단국대학교 모바일시스템공학과

32154231 정영환

32161681 박소빈

Index

1. Overview & Intro

Factory Automation(이하 FA)은 컴퓨터 시스템과 산업 로봇을 연계하여 공정 전체의 자동화 무인화 생산관리의 자동화 등을 행하는 일체의 시스템을 의미한다. 2020 년 3 월 현재 수많은 회사에서는 자동화 시스템을 통한 업무, 공정을 생산성과 효율성을 검증하고 있으며 그 결과를 바탕으로 높은 신뢰도의 공정결과 생산, 예측되는 공정실패 지점, 유효한 결과 분석, 지속적인 결과의 되먹임으로 인한 프로세스 성능향상을 이룩하고 있다.

이러한 시대적 흐름에 맞추어 기존에 일선현장에서 수기방식으로 이루어지던 일련의 작업들을 자동화하고 시간과 정확성 그리고 예측가능성을 포함시켜 보다 수월한 업무 환경 구축에 도움이 되고자 해당 프로젝트를 시작하게 되었다.

우리가 오늘날 사용하고 있는 응용프로그램(이하 프로그램)은 일련의 자동화 프로세스의 집합으로 정의되며 운영체제에서 실행되는 모든 소프트웨어를 뜻한다. 좁은 의미에서는 사용자가 직접 사용하게 되는 소프트웨어를 뜻하며 넓은 범위에서 정의하면 그 외에 시스템상에서 사용자의 편의를 위해 자동으로 보정되는 시스템 프로그램까지도 일부 포함한다고 규정할 수 있겠다.

컴퓨터의 하드웨어는 소프트웨어가 있어야만 동작을 하게 되는 구조이다. 하드웨어에 시그널을 보내고 연산 결과를 돌려받아 사용자에게 결과를 도출하는 작업에 있어서 이를 시각화 해 줄 수 있는 개념을 소프트웨어라고 한다.

그리고 이를 자동화하여 메모리 액세스, 캐시관리, 유저할당, 네트워크 연결 및 호스팅과 같은 일련의 작업을 도와주는 소프트웨어를 운영체제라고 정의한다.

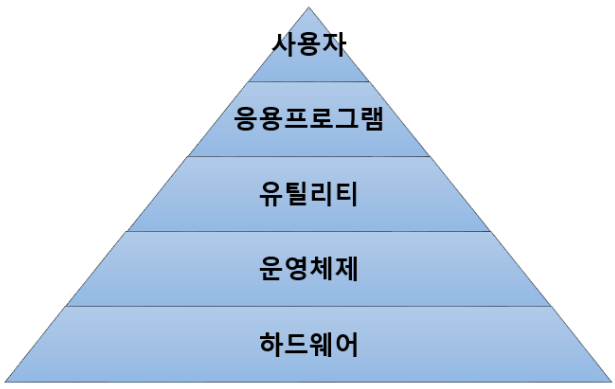

그림1. 시스템 프로그램에서 운영체제(Operating Systems, OS)의 개요

실질적으로 동작하는 응용프로그램은 실질적으로는 운영체제(이하 OS)에 의존하여 자원을 할당 받고 결과물을 출력해 타 프로그램에 접근하는 형태로 구성되어 있다.

따라서 이 문서(프로젝트)에서는 제조사에서 제공하는 CA 관련 아이디어를 조합하고 실질적인 UL Korea 업무 절차 프로세스를 파악하고 재구성한 뒤 프로그램화 시켜 업무의 자동화 + 공정의 자동화를 이룩하여 안정화된 업무 환경을 구축하고 신뢰성 있는 결과를 산출하고 기획되었다.

2. What is Carrier Aggregation (CA)?

운반파 묶음이라고 불리는 Carrier Aggregation (이하 CA) 기술은 LTE- Advanced(이하 LTE-A)의 핵심 기술 중 하나이다. 다중 주파수 블록 (이른바 Component Carrier, 컴포넌트 캐리어)가 동일 사용자에게 할당된 상태에서 사용자당 데이터 속도를 증가시키기 위해 무선통신에 사용되는 기술이다.

사용자별 잠재적인 최대 데이터 속도는 더 많은 주파수 블록이 사용자에게 할당될수록 증가하게 된다. 셀의 총체적인 데이터 속도 또한 더 나은 자원활용 덕분에 증가하게 된다.

CA의 유형

컴포넌트 캐리어(이하 CC)의 위치에 따라 3 가지 케이스의 운반파 묶음(CA)으로 구별된다.

Intra-band contiguous	컴포넌트 캐리어가 동일 주파수대에 인접해 있는 경우

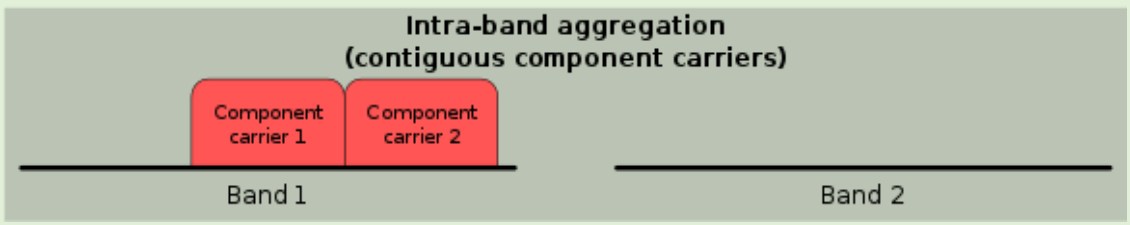

그림2. Intra-band contiguous

Intra-band non-contiguous	**컴포넌트 캐리어가 동일 주파수대이나, 연속적이지 않은 경우**

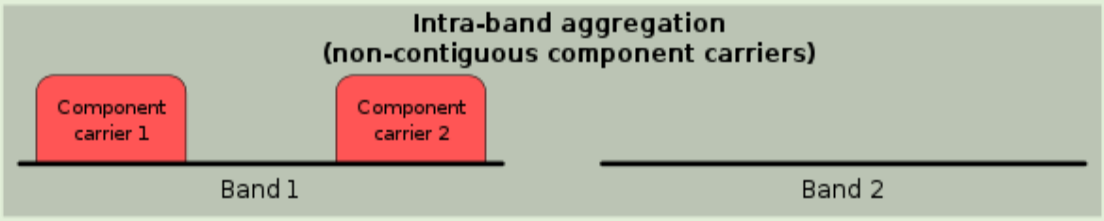

그림3. Intra-band non-contiguous

Inter-band	*** 가장 복잡한 케이스** **컴포넌트 캐리어가 각기 다른 주파수대에 위치하고 있는 경우**

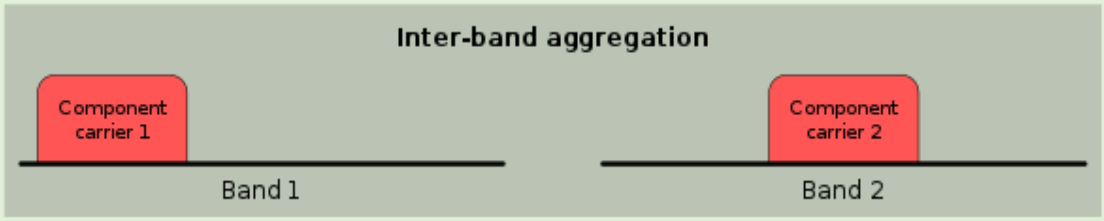

그림4. Inter-band

Baseband 관점에서 3 가지 케이스 간의 차이점은 없다. 허나 RF 관점에서 대역간 CA 케이스의 복잡도는 증가한다.

따라서 이러한 몇가지의 CA 조합끼리의 재조합으로 2cc 3cc 4cc 5cc 까지 증대 될 수 있다. 다음, 그 결과를 논리적으로 분석해 실질적인 CA 측정 절차에 해당하는 *REL. 10LTE SAR TEST GUIDANCE AND KDB INQUIRIES 941225*에 근거하여 측정절차를 설립하고 그 결과를 분석해 재구성하는 방식으로 진행된다.

아래표는 참고한 주파수 표이다.

FDD LTE Frequency Bands

LTE Band Number	Uplink Band (MHz)	Downlink Band (MHz)	Band Width (MHz)	Duplex Spacing (MHz)	Band Gap (MHz)
LTE Band 1	1920 - 1980	2110 - 2170	60 MHz	190 MHz	130 MHz
LTE Band 2	1850 - 1910	1930 - 1990	60 MHz	80 MHz	20 MHz
LTE Band 3	1710 - 1785	1805 - 1880	75 MHz	95 MHz	20 MHz
LTE Band 4	1710 - 1755	2110 - 2155	45 MHz	400 MHz	355 MHz
LTE Band 5	824 - 849	869 - 894	25 MHz	45 MHz	20 MHz
LTE Band 6	830 - 840	875 - 885	10 MHz	35 MHz	25 MHz
LTE Band 7	2500 - 2570	2620 - 2690	70 MHz	120 MHz	50 MHz
LTE Band 8	880 - 915	925 - 960	35 MHz	45 MHz	10 MHz
LTE Band 9	1749.9 - 1784.9	1844.9 - 1879.9	35 MHz	95 MHz	60 MHz
LTE Band 10	1710 - 1770	2110 - 2170	60 MHz	400 MHz	340 MHz
LTE Band 11	1427.9 - 1452.9	1475.9 - 1500.9	20 MHz	48 MHz	28 MHz
LTE Band 12	698 - 716	728 - 746	18 MHz	30 MHz	12 MHz
LTE Band 13	777 - 787	746 - 756	10 MHz	-31 MHz	41 MHz
LTE Band 14	788 - 798	758 - 768	10 MHz	-30 MHz	40 MHz

그림5. FDD LTE Frequency Bands

LTE Band 15	1900 - 1920	2600 - 2620	20 MHz	700 MHz	680 MHz
LTE Band 16	2010 - 2025	2585 - 2600	15 MHz	575 MHz	560 MHz
LTE Band 17	704 - 716	734 - 746	12 MHz	30 MHz	18 MHz
LTE Band 18	815 - 830	860 - 875	15 MHz	45 MHz	30 MHz
LTE Band 19	830 - 845	875 - 890	15 MHz	45 MHz	30 MHz
LTE Band 20	832 - 862	791 - 821	30 MHz	-41 MHz	71 MHz
LTE Band 21	1447.9 - 1462.9	1495.5 - 1510.9	15 MHz	48 MHz	33 MHz
LTE Band 22	3410 - 3500	3510 - 3600	90 MHz	100 MHz	10 MHz MHz
LTE Band 23	2000 - 2020	2180 - 2200	20 MHz	180 MHz	160 MHz
LTE Band 24	1625.5 - 1660.5	1525 - 1559	34 MHz	-101.5 MHz	135.5 MHz
LTE Band 25	1850 - 1915	1930 - 1995	65 MHz	80 MHz	15 MHz
LTE Band 26	814 - 849	859 - 894	30 / 40 MHz		10 MHz
LTE Band 27	807 - 824	852 - 869	17 MHz	45 MHz	28 MHz
LTE Band 28	703 - 748	758 - 803	45 MHz	55 MHz	10 MHz
LTE Band 29	n/a	717 -728	11 MHz		
LTE Band 30	2305 - 2315	2350 - 2360	10 MHz	45 MHz	35 MHz
LTE Band 31	452.5 - 457.5	462.5 - 467.5	5 MHz	10 MHz	5 MHz

그림6. FDD LTE Frequency Bands

TDD LTE Frequency Bands

LTE Band Number	Frequency (MHz)	Bandwidth (MHz)
33	1900 - 1920	20
34	2010 - 2025	15
35	1850 - 1910	60
36	1930 - 1990	60
37	1910 - 1930	20
38	2570 - 2620	50
39	1880 - 1920	40
40	2300 - 2400	100
41	2496 - 2690	194
42	3400 - 3600	200
43	3600 - 3800	200
44	703 - 803	100

그림7. TDD LTE Frequency Bands

(a) Why this work needs to be?

이러한 일련의 절차의 필요성이 대두됨은 실질적인 제조사에서 제공하는 조합 시트의 내용에는 불필요한 데이터가 포함되어있고 해당 내용을 KDB INQUIRIES 에 맞추어 재생산하고 다시 재구성하여 사내 규격문서로 탈바꿈하는 과정이 결과적으로는 상당한 시간을 소요하고 그 과정에서 몇가지의 실수가 일어날 가능성을 포함하는 바 이하의 구성을 단순화시켜 자동화한다.

조합구성 절차

1	제조사 제공 테이블을 분석해 프로그램의 입력데이터가 될 Form 에 입력 -중복 여부에 상관없이 작성한다.
2	같은 CC 간 중복 데이터 제거 - Reverse 예외 처리 과정
3	다른 CC 간 중복데이터 제거 -쌍방데이터의 Reverse 여부 검증
4	다른 CC 간 Coverage 의 경우에는 높은 CC 가 우선 순위 갖도록 코드 구성
5	중복데이터는 블락 처리 및 후에 재 검증 가능하도록 분류 작업 -블락 처리된 중복데이터의 경우 다른 색깔의 cell 로 표현
6	실질적으로 측정해야 할 데이터 문서화 및 새로운 엑셀 파일 생성
7	실질적으로 측정할 조합 값을 Reverse 포함여부 혹은 Restriction 포함여부등을 분석하여 새로운 form 에 재구성 및 나열 -Uplink PCC 의 측정한 power 값 중 maximum power 입력
8	Inter band CA mid Channel 구성형태로 함수 작성, 적용
9	Intra band CA non-contiguous 구성 및 함수 작성, 적용
10	Intra band CA contiguous 구성 및 함수 작성, 적용
11	Inter + intra band 함수 구성 및 충돌여부 판정
12	전체 함수 코드 구성 및 적용

위 구성 절차는 SAR 팀 사원분들과 과장님께 여쭙고 나름대로 도식화하고 정리한 후 소프트웨어 개발방법론적인 측면을 고려해 일괄적으로 재구성한 내용이다.

3. Software Development Methodology

1. **소프트웨어 개발방법론이란?**

   * 소프트웨어 개발에 필요한 반복적인 과정 절차 방법 산출물 기법 도구 등을 정리해 항목화 시키고 구현해 나가는 과정을 도식화한 것
   * 소프트웨어 개발에 대한 분석 설계 구축에 대한 정형화된 방법과 기법을 의미
   * 프로젝트 진행방법과 수행노하우를 체계화하고 표준화한 것을 의미

2. 개발 방법론의 종류

1. 구조적 방법론

구조, 흐름, 간결, 간단 == 구조적 개발방법의 특징을 의미한다. (프로세스 중심)

요구사항 분석 : (첨부된 SRS 참고)

구조적 분석 : 원하는 기능을 구현하기 위한 시스템환경,데이터를 종합하여 데이터 흐름도를 구성

구조적 설계 : 모듈 중심으로 설계를 하는 당계 모듈의 목적은 재활용 결합도를 낮추어 독립성을 높이는 작업

구조적 프로그래밍 : 프로그램 복잡성을 최소화 하기 위해 순차 ,조건,반복 3개의 논리구조를 구성

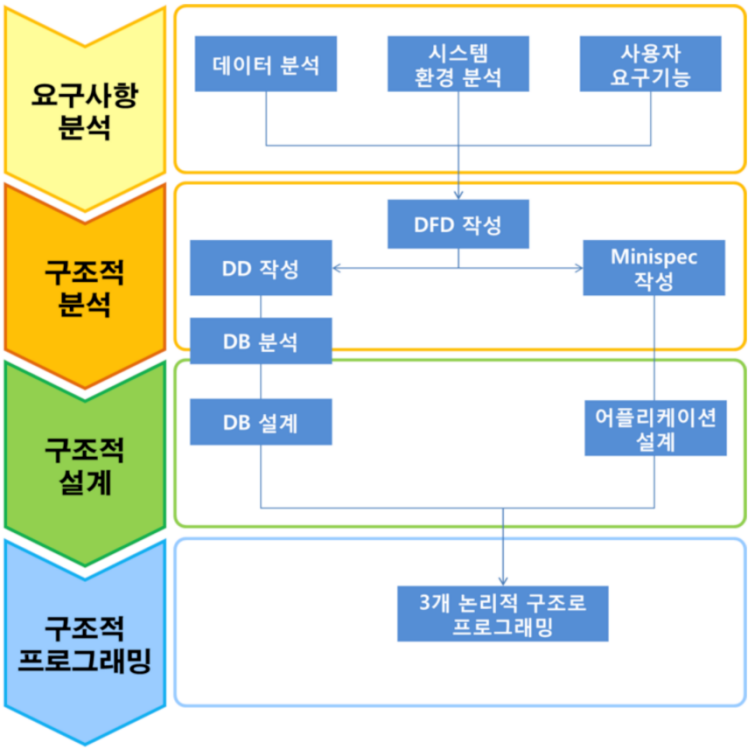

그림8. 소프트웨어 공학의 구조적 개발 방법론

- 구성 요소

데이터 흐름도 (Data Flow Diagram, DFD)

: 각기능을 분할하여 표현한 구조도

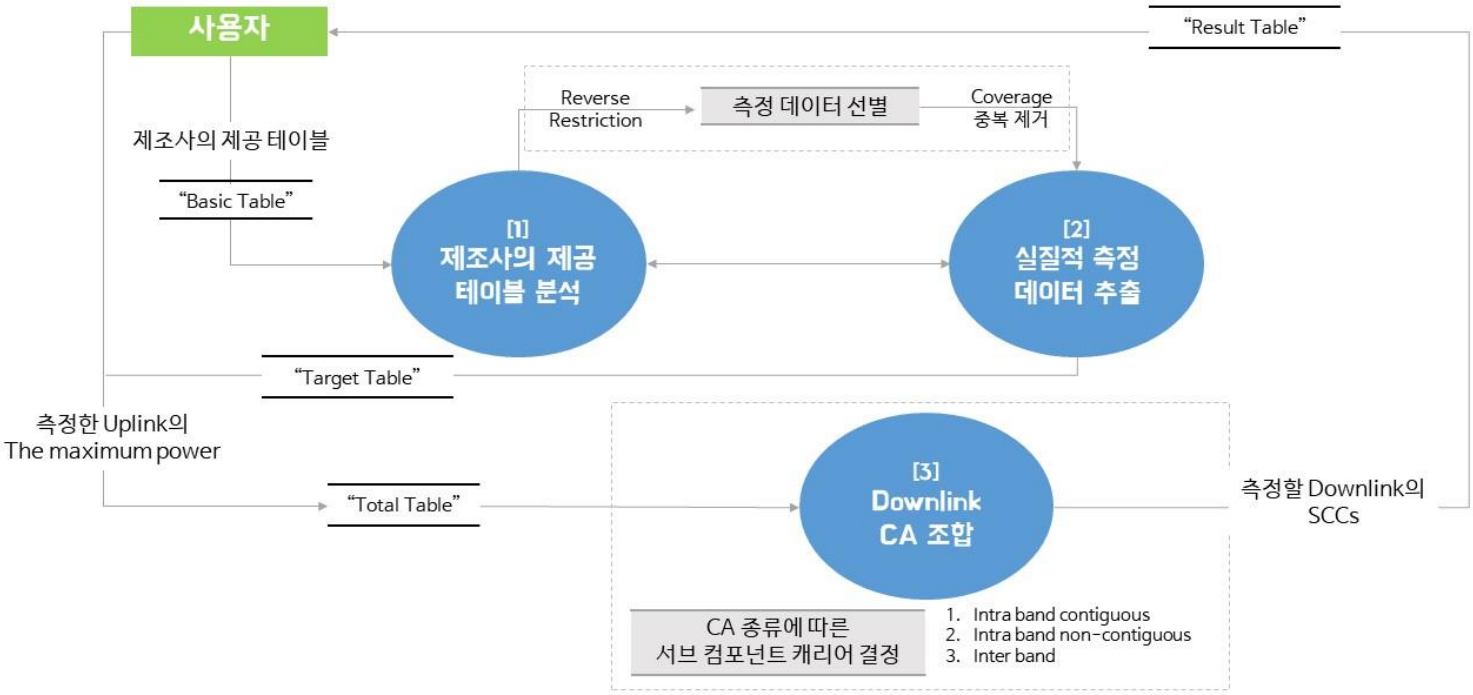

그림9. 프로젝트의 DFD

표기법	의미
[진행 순서 #] 프로세스 이름	프로세스 처리
자료 이름 →	자료 흐름
" 저장소 이름 "	자료 저장소, 데이터 베이스
데이터 처리 과정	데이터 흐름에 따른 작성 코드 내에서 처리되는 과정
단말 이름	자료의 출발과 도착지

그림10. 프로젝트의 DFD 구성요소

상태전이도 (State Transition Diagram, STD)

: 어떤 상태에서 다른 상태로 변경되는 과정과 그 프로세스를 명시하는 것

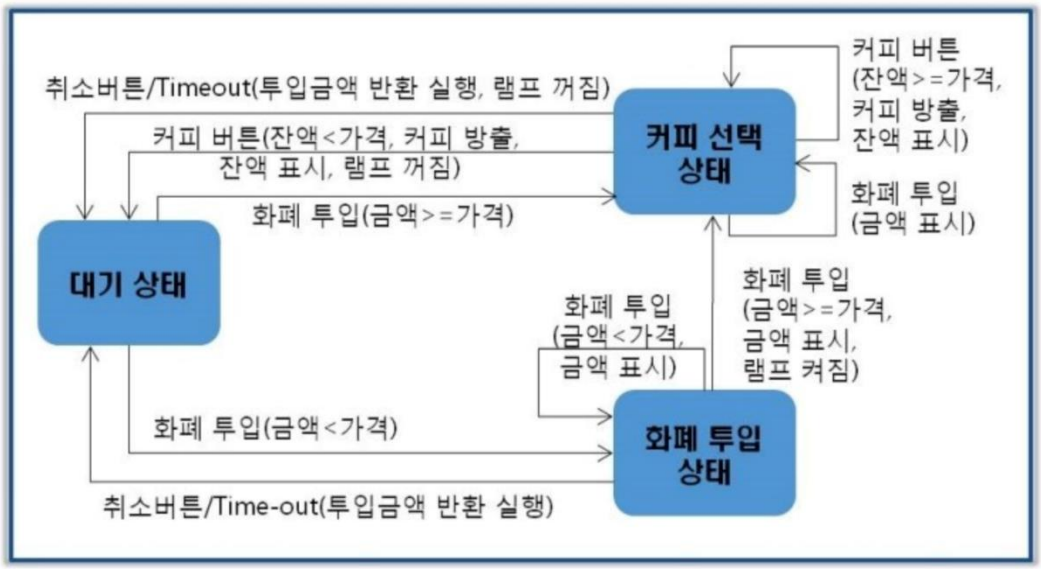

그림11. 상태전이도(STD)의 예시

방법론

: 폭포수 모델 활용

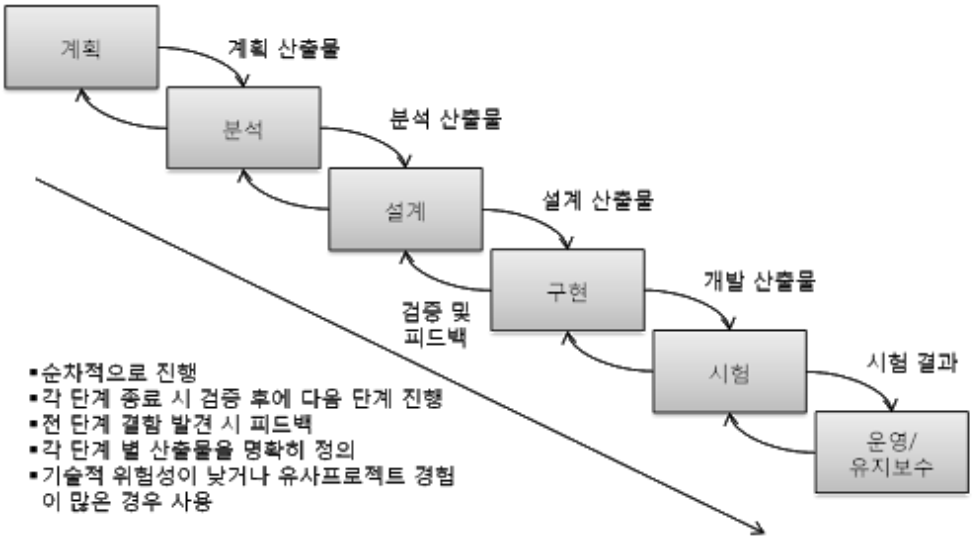

그림12. 소프트웨어 개발 프로세스 (폭포수 모델, Waterfall model)

소프트웨어 구성 단계에서 각 단계를 진입하거나 회피 혹은 후퇴할 수 있으며 결과의 구조나 이후 구성할 데이터의 구조 혹은 프로세스의 절차적인 흐름에 따라 재구성되고 반복될 수 있다.

2. 정보공학적 방법론

- 비즈니스 시스템 규모 성장과 소프트웨어 공학 발전에 따라 1980 년대 중반에 등장한 방법론 (데이터 중심)

- 정보시스템을 구축하기 위한 방법으로서 계획 분석 설계 등 전 과정을 정형화시킨 절차 및 방법론 단순 SW 개발이 아닌 실질적인 업무의 효율성을 증대

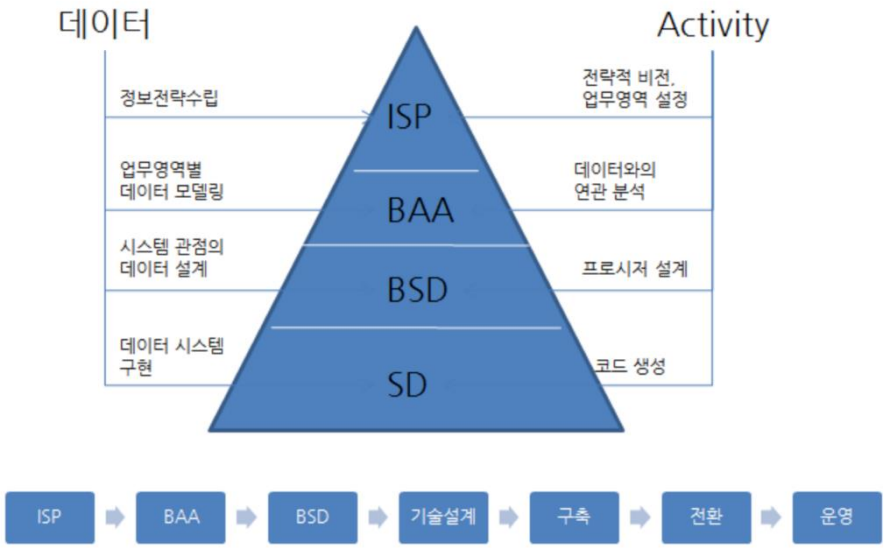

그림14. 정보공학방법론

개발 프로세스

(1) ISP (Information Strategy Planning)

: 기업의 중장기 정보화 전략 수립

경영전략 분석 -> 현행 업무 프로세스 분석 -> 현행 시스템 분석/평가 -> 아키텍처 개발 -> 전략 계획 (우선 순위 도출)

(2) BAA (Business Area Analysis)

: ISP 에서 수집된 자료를 기반으로 세부적으로 확장하는 단계
(업무 영역별 데이터와 프로세스 모델링, 연관 분석)

데이터 모델 다이어그램, 프로세스 분할 다이어그램, 프로세스/데이터 매트릭스 (CRUD 매트릭스)

(3) BSD (Business System Design)

: 비지니스 시스템 설계 (즉, 업무 시스템 설계)

논리적 ERD (Entity Relationship Diagram) 및 DFD 등: ***데이터 설계***
분할 다이어그램, 액션 다이어그램, 의존 다이어그램: ***프로세스 설계***

(4) SD (Structural Design)

: 시스템 설계를 기반으로 정보시스템을 구축하는 단계

연관분석, 물리적 데이터베이스 설계, 코딩 등의 작업 수행

기업의 정보시스템 구축을 목적으로 진행되기 때문에, 일관성 있고 통일된 정보시스템 구축이 가능하다. 따라서 데이터 중심으로 진행되기 때문에 업무 절차 및 환경 변화에 유연하다.

3. 객체지향방법론

- 프로그램을 객체와 객체 간의 인터페이스 형태로 구성하기 위하여 문제영역에서 객체, 클래스 간의 관계를 식별하여 설계모델로 변환하는 방법론
- 추상화, 캡슐화, 정보은폐, 상속, 다형성의 특징을 가짐
- **계획 > 분석 > 설계 > 구현> 테스트** 단계를 거쳐 개발

단계	작업항목	설명
객체지향분석	객체모델링 - 객체다이어그램	시스템 정적 구조 포착 추상화,분류화,일반화,집단화
	동적모델링 - 상태다이어그램	시간흐름에 따라 객체 사이의 변화조사 상태, 사건, 동작
	기능모델링 - 자료흐름도	입력에 대한 처리결과에 대한 확인
객체지향설계	시스템 설계	시스템구조를 설계 성능최적화 방안, 자원분배방안
	객체 설계	구체적 자료구조와 알고리즘 구현
객체지향구현	객체지향언어(객체,클래스)로 프로그램	객체지향언어(C++, JAVA), 객체지향 DBMS

그림15. 객체지향방법론

4. CBD (Component Base Development) 분석방법론

재사용가능한 컴포넌트의 개발 또는 상용 컴포넌트를 조합해 어플리케이션 개발생산성과 품질을 높이고 시스템 유지 보수 비용을 최소화할 수 있는 개발 방법 프로세스

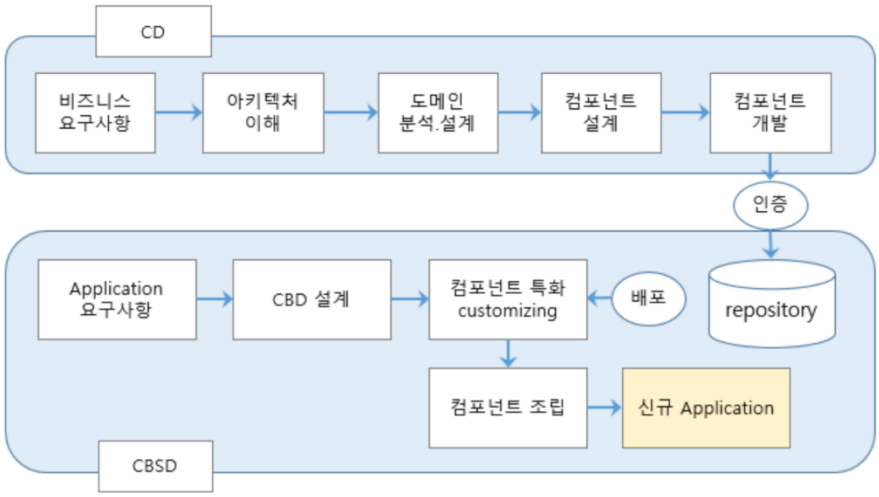

그림16. CBD 분석방법론

그림17. 개발 방법론 표준 프로세스 구성

따라서 이 문서에서 다루는 CA combination and Reconfiguration Auto process에서는 [그림17] 절차 중의 일부를 차용하고 현 UL코리아의 업무 진행 절차 방식을 감안하여 모듈화 시킨 개념을 적용하였다.

<참고> 특허청 sw개발방법론 정보화 업무규정 (훈령 715조) 행안부 정보시스템 구축운영지침 45조, 행안부 제 2011-36호 참조

<정리>

개발 방법론 표준 프로세스 구성

<참조> Python Software 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 고민해야할 필요가 있다.

일반적으로 '구성'이란 로직과 의존성이 깔끔한 코드를 만드는 것을 의미할 뿐 아니라, 파일시스템에 구성된 파일과 폴더에 쉽게 접근할 수 있고 유연성이 존재하며 빠른 결과물 생성에도 초점을 둔다.

어느 모듈에 어느 기능이 들어가야 하는가? 는 가장 중요한 문제이다. 프로젝트에서 데이터는 어떻게 흘러가야 하는가? 각각통합되어야 하거나 분리되어야 하는 기능은무엇인가? 방금한 그 작업이 프로그램의 성능에 어떤 영향을 주는가? 등 여러가지 측면에 대해서도 고려해야 한다.

이하 알파벳 인덱스는 해당 프로그램을 구성하면서 고려한 요소들이며, 원활하고 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 확실하고 가시적인 결과물을 만들어 내는데 도움을 줄 수 있다.

즉, 파일의 종류를 분류하고 집적하되 체계적으로 정리하고 논리적으로 구성하는 것에 의의를 뒀다.

B. 코드 구성

파이썬의 임포트(import) 방식과 잘 만들어진 모듈은 프로그램 개발이 원활하게 돕는 역할을 하여 더욱 원활한 프로젝트 진행을 할 수 있게 한다. 그러나 여기서 말하는 '쉽다' 라는 말은 쉽게 엉망이 되기도 한다는 것을 의미한다. 코드를 가독성 있게 작성하는데 피해야할 몇 가지 요소가 존재한다.

i. 복잡한 순환 참조
ii. 숨겨진 링크연결
iii.전역 구문의 과다한 사용
iiii.스파게티 코드
v. 라비올리 코드

따라서 코드가 갖는 복잡성을 단순화하고 의존성을 줄이며, 정확한 목표를 구현하는 것에 초점을 두었다.

C. 모듈 (Module)

Python의 모듈은 사용가능한 추상 레이어 중 하나로 동작한다. 이는 코드를 기능파트와 저장파트로 나눌 수 있도록 도와준다.

예를 들어 프로젝트의 레이어 중 하나는 사용자 인터페이스를 담당하고 다른 하나는 저 수준(Low level)의 데이터 처리를 담당한다. 이렇게 레이어를 분리시키는 방법은 인터페이스기능과 저 수준의 데이터 처리를 가능하게 하는 파일을 세분화하여 분리하는 것부터 시작한다.

이를 import ...from ....import....형태로 구분하며 서로 간의 의존성을 상승시킨다. 이는 import 구문에 즉시 사용 가능한 import sys와 import os를 포함한 서드파티 모듈을 의미한다.

이러한 모듈은 타 코드와는 다른 특징을 보이는데 타 언어에서 의미하는 include file 형태의 지시문은 전 처리 장치 즉 preprocessor가 해당 파일의 모든 코드를 가져와 호출자에서 복사해 사용하는 방식으로 진행된다면 python에서는 모듈에 포함된 코드를 namespace에서 독립적으로 실행하는 것으로 구분된다.

이는 전체 코드가 오작동하는 경우를 미연에 방지하고 원활한 데이터 처리를 가능하게 함을 의미한다.

D. 패키지 (Package)

파이썬은 패키징 하는 것에 있어서 간단하고 명료하게 작동한다. 이는 파이썬의 모듈 구조를 단순하게 디렉토리 형태로 상승화 시킨 것으로 인식할 수 있다.

__init__.py가 있는 모든 디렉토리는 파이썬 패키지로 분류되며 여러 다른 모듈과 비슷하게 불러와 지며 구성되고 재생산된다. 이 파일들은 패키지 전체의 모든 정의를 모으는 용도로 사용된다.

E. 객체지향 프로그래밍 (Object Oriented Programming)

python에서 모든 것은 객체(object)로 분류된다. 그리고 객체처럼 다룰 수가 있다. 클래스(class), 문자열(string) 심지어 타입(data type)까지도 파이썬에선 객체로 취급된다. 다른 객체처럼 타입이 있고 함수의 인자 값을 받을 수 있다. 게다가 메소드와 속성을 포함할 수도 있다. 이러한 점을 고려하여 파이썬을 객체 지향 프로그래밍이라고 한다.

하지만 자바와 달리 파이썬은 객체 지향 프로그래밍을 프로그램 패러다임으로 두지 않는다. 따라서 클래스를 정의하지 않아도 실행이 가능한 수준에 이른다. 게다가 모듈에서도 살펴보았듯, 파이썬이 모듈과 namespace를 분리하여 다루는 방식은 개발자로 하여금 자연스럽게 캡슐화와 추상 레이어의 분리를 가능하게 해준다. 따라서 클래스의 생성에 집착하지 않아도 되는 편안한 환경을 조성한다.

어떤 아키텍처,어플리케이션에서는 파이썬 프로세스에 동시에 발생하는 외부 요청에 응답하기 위해 복수의 인스턴스(instance)가 발생하거나 구문의 실행이 멈춘 상태로 정지되어 있는 경우도 있다. 이처럼 고정된 정보를 계속 붙잡고 있을 필요성이 있을 때 사용된다.

이러한 문제를 피하기 위해서는 모호한 문맥과 부작용을 최소화하는 함수를 사용하는것이 좋다. 또한 함수 내부에 있는 persistant 레이어 항목을 남용하면 함수의 의미가 모호해져 맥락의 변화를 일으키게 되므로 저장된 값을 일정하게 유지하는 것이 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 쓰이면 메모리의 한 조각이 아니기 때문에 함수가 될 수도 있고string이 될 수도 있으며 int가 될 수도 있다.

이런 동적 타이핑은 약점으로 여겨 지기도 하며 디버깅이 어려워지는 문제를 초래하기도 한다.

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

[참조]

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

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

4.Development Environment and Use Environment

1. Visual Studio Code

: 2015년 4월 19일에 소개된 후 2016년 4월 15일 정식 출시된 텍스트 에디터로 Microsoft사에서 개발되었다.

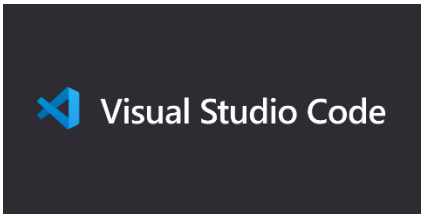

그림18. Visual Studio Code 로고

: Electron 프레임워크를 기반으로 구조되어 있고 마이크로소프트사 개발 플랫폼 최초로 크로스 플랫폼을 지원하는 에디터이며 윈도우, 맥, 리눅스를 모두 지원한다.

CPU		1.6 GHz 이상	
RAM		1GB 이상	
용량		200MB 이상	
운영 체제	Windows	Windows 7 이상	
	Linux	데비안 계열	우분투 14.04, 데비안 7 이상
		레드햇 계열	RHEL 7, CentOS 7, 페도라 23 이상
	macOS	10.10 Yosemite 이상	
추가 요구 사항	Windows	.NET Framework 4.5.2 이상	
	Linux	glibcxx 3.4.14, glibc 2.15 이상	

IDE 가 아니므로 Builder 가 내장되어 있지 않아 VScode 내에서 별도의 컴파일 환경을 구축해야 하며 코드 실행 플러그인으로는 code runner 가 있다.

터미널을 자체 내장하고있어서 별도로 터미널 창을 열 필요가 없으며 비주얼 스튜디오와 인텔리전스를 공유하여 자동으로 지원된다. 그 기원인 git 연동도 가능하며 TFS 연동은 확장을 하면 가능하다. 작업 플러그인에 git init 이 있으면 별도의 커밋(commit) 없이 자동으로 연동되는 작업을 지원한다.

웹과 클라우드 기반으로 개발 디버깅 툴로 기획되었으나 마이크로 소프트 제품가운데 유래가 없을 정도로 호평을 받은 제품군에 속한다.

[참고] https://code.visualstudio.com/docs

2. 리눅스

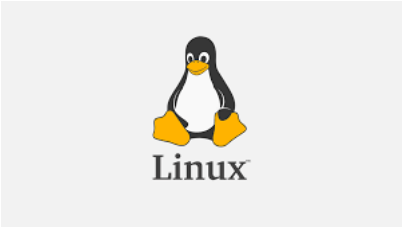

그림19. Linux 로고

커널의 일종인 리눅스 커널, 또는 리눅스 커널을 사용하는 운영 체제를 가리키는 말이기도 하다. GNU 쪽 사람들은 리눅스는 커널일 뿐이고, 이 커널을 이용해 GNU 프로그램들을 올려 만든 운영 체제는 GNU/Linux 라고 하기도 한다 소스 코드가 공개되어 있는 대표적인 오픈 소스 소프트웨어다. 컴퓨터 역사상 가장 많은 사람이 들어간 오픈 소스 프로젝트이며, 모바일 운영 체제로 유명한 안드로이드가 이것을 기반으로 한다.

소스 코드가 공개되어 있기 때문에 해당 CPU 및 보드에 맞도록 설정만 주고 컴파일하면 원칙적으로 CPU 가 딸린 어떤 기계에서도 돌릴 수 있다. 일반적으로 생각하는 PC 용 운영체제 말고도 임베디드 시스템에 리눅스 커널을 얹어서 돌리는 경우도 많으며, 인터넷 공유기나 PDA 나 휴대폰에도 리눅스를 OS 로 사용하는 경우가 있다. 2011 년 들어서 각광을 받고 있는 안드로이드도 자주 쓰이는 x-window 가 서피스 플링거로 대체되고 Dalvik 가상머신 등의 안드로이드 프레임워크 계층이 추가된 리눅스이다.

[참고] linux.org/forums/

3. Python

그림20. Python 로고

1991 년 프로그램 귀도 반 로섬이 발표한 고급 프로그래밍언어로 플랫폼에 독립적이며 인터프리터식 객체지향적 동적 타이핑을 지원하는 대화형언어이다. 파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형,공동체 기반개발모델을 가지고 있으며 C 언어로 구현된 C 파이썬이 표준으로 매김한다.

[참고] https://docs.python.org/3/

4. Openpyxl, pyqt5

Excel2010 파일을 읽고 쓰는 python 라이브러리이다.

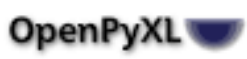

그림21. OpenPyXL 로고

OpenPyXL로서 PHP Excel 팀에 대한 모든 구도는 처음에 PHP Excel에 기반을 두고 있다. 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 웹 브라우저 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 어플리케이션에 내장되어 해당 어플의 사용자가 기능을 구성하고 향상시킬 수 있게 설정된다.

그림22. PyQT5 로고

[참조] https://pypi.org/project/PyQt5/

5. Codework and Explanation

```
import openpyxl as op
from openpyxl.styles import  PatternFill,Color
from collections import Counter
import sys
import copy
```

그림23. 외부라이브러리 import 선언

```
#original = re_test
send =  op.load_workbook('sample1.xlsx')
target = op.load_workbook('DL_CA_output_cpy.xlsx')
sheet = send['Sheet1']
tar_table = target['Sheet1']

index_len=len(sheet['A'])

index_len=len(sheet['A'])

blank_list=[]

index_list=['A','F','K','P']
cc_list=['B','G','L','Q'] #2cc,3cc,4cc,5cc
rev_list=['E','J','O','T']
cover_list=['D','I','N','S']
res_list=['C','H','M','R']

st = 3
ok_list=[]
no_list=[]
ch_cnt=0
CA_cnt=3
```

그림24. 엑셀파일에 접근하기 위한 변수 선언

외부라이브러리를 import하여 구성이 완비될 수 있도록 선언한다.

엑셀파일을 읽어오고 그 결과를 정리하는데 그치지 않고, 중간 연산을 가능하게 하며 추가적인 시스템 접근까지 고려해 코드를 구성한다.

엑셀 파일을 열고 저장하며 구성된 데이터를 올바르게 불러와 변수에 저장해 파라미터 삼아 여러가지 데이터에 접근 가능하게 만들 변수들을 선언한다.

```python
def Band_cal(a):
    data = []
    if int(a) == 2:
        data.append(20)
        data.append(900)
        data.append(1960)
    elif int(a) == 4:
        data.append(20)
        data.append(2175)
        data.append(2132.5)

    elif int(a) == 5:
        data.append(10)
        data.append(2525)
        data.append(881.5)
    elif int(a) == 7:
        data.append(20)
        data.append(3100)
        data.append(2655)
    elif int(a) == 12:
        data.append(10)
        data.append(5095)
        data.append(737.5)
```

그림25. 각 주파수 대역대의 최대 Bandwidth, Mid channel 등에 대한 수치 입력

전체 함수 코드 중 일부이다.

미드 채널cc에 관해서는 이미 결정되어 있는 바가 있어 해당 함수는 1:1 매핑 형태로 구성했다.

```python
for x in range(len(cc_list)):
    cc_len = len(sheet[cc_list[x]])
    for i in range(3,cc_len):
        if sheet[cc_list[x]][i].value == None or sheet[cc_list[x]][i].value == 'Same':
            continue
        if sheet[res_list[x]][i].value != None:
            continue
        split_x = sheet[cc_list[x]][i].value.split('-')
        for j in range(i+1,cc_len):
            if sheet[cc_list[x]][j].value == None or sheet[res_list[x]][j].value != None or sheet[cc_list[x]][j].value
                continue
            split_y = sheet[cc_list[x]][j].value.split('-')
            cnt=0

            p= Counter(split_x)
            q= Counter(split_y)
            for key in p:
                for B_key in q:
                    if key == B_key:
                        if p[key]==q[B_key]:
                            cnt+=1

            if cnt == len(p):
                sheet[cc_list[x]][j].value = 'Same'
                sheet[cc_list[x]][j].fill = PatternFill(patternType='solid', fgColor=Color('ffff00'))
                sheet[cover_list[x]][j].value = sheet[index_list[x]][i].value
                sheet[cover_list[x]][j].fill = PatternFill(patternType='solid',fgColor=Color('ffff00'))
```

그림26. 동일한 CC 내 중복되는 CA 조합 제거하는 코드 중 일부

중복되는 결과치들을 산출하고 정리하는 작업을 통해서 cc간 중복 절차를 간소화하고 시험 케이스를 줄이는 작업을 했다.

```
for se_n in range(len(cc_list)-1):#2cc부터
    for ta_n in range(se_n+1,len(cc_list)):#끝cc까지
        se = cc_list[se_n]
        ta = cc_list[ta_n]
        r_se = rev_list[se_n]
        r_ta = rev_list[ta_n]

        for i in range(st,index_len):#첫 cc의 출발 범위
            if sheet[se][i].value == None:
                continue
            if sheet[se][i].value == 'Same':
                continue

            split_x = sheet[se][i].value.split('-')#있으면 하나씩

            for j in range(st,index_len):
                if sheet[ta][j].value == None:
                    continue
                if sheet[ta][j].value == 'Same':
                    continue

                if sheet[r_ta][j].value != 'No':
                    split_y = sheet[ta][j].value.split('-')
                    cnt=0

                    p= Counter(split_x)
                    q= Counter(split_y)
                    for key in p:
                        for B_key in q:
                            if key == B_key:
                                if p[key]<=q[B_key]:
                                    cnt+=1
```

그림27. 서로 다른 CC간 비교를 통해 Coverage 여부를 판단하는 코드 중 일부

2cc부터 끝cc까지 돌면서 중복되는 과정을 skip하고 올바르게 측정되어야 하는지를 판단하기위한 코드이다.

몇 가지 dependency가 발견되어 수정하고 정리했다.

```
                elif sheet[r_ta][j].value =='No' and sheet[r_se][i].value=='No':
                    split_y = sheet[ta][j].value.split('-')
                    if split_x[0]==split_y[0]:
                        split_x.pop(0)
                        split_y.pop(0)
                        p=len(split_x)
                        cnt = 0
                        for x in split_x:
                            if x in split_y:
                                cnt+=1
                        if cnt==p:
                            if sheet[se][i].value not in ok_list:
                                ok_list.append(sheet[se][i].value)
                                sheet[cover_list[se_n]][i].value=sheet[index_list[ta_n]][j].value
                                sheet[cover_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff000
                                sheet[cc_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff0000')
                                ch_cnt+=1
                                #print('중복 발견 색상 변경, ',ch_cnt)
                            elif sheet[se][i].value in ok_list:
                                sheet[cover_list[se_n]][i].value=sheet[index_list[ta_n]][j].value
                                sheet[cover_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff000
                                sheet[cc_list[se_n]][i].fill = PatternFill(patternType='solid', fgColor=Color('ff0000')
                                ch_cnt+=1
                                #print('중복 발견 색상 변경, ',ch_cnt)



        for i in range(st,index_len):
            if sheet[se][i].value == None:
                continue

            if sheet[se][i].value not in ok_list:
                if sheet[se][i].value not in no_list and sheet[se][i].value !='Same':
                    no_list.append(sheet[se][i].value)
        ok_list.append('//')
        no_list.append('//')
print("Process complete")
```

그림28. 해당 코드를 통해서 중복된 값을 모두 빼고 제조사에서 제공한 데이터 중에서 실질적으로 측정해야 할 값들만 남길수 있게 된다.

```
for x in no_list:
    if x =='//':
        continue
    split_text = x.split('-')
    set_text=list(set(split_text))

    for p in set_text:
        min_data = list(p)
        if 'B' in min_data or 'C' in min_data:
            split_text.append(p)
        elif 'D' in min_data:
            split_text.append(p)
            split_text.append(p)
        elif 'E' in min_data:
            split_text.append(p)
            split_text.append(p)
            split_text.append(p)

    while(len(split_text)!=4):
        split_text.append(' ')

    for p in range(len(set_text)):
        tar_table['B'][CA_cnt].value = x
        tar_table['C'][CA_cnt].value = set_text[p]
        num=0
        split_text_cpy = copy.deepcopy(split_text)
        for de in range(len(split_text_cpy)):
            if set_text[p]==split_text_cpy[de]:
                del split_text_cpy[de]
                break

        for q in range(len(split_text_cpy)):
            if  num==0:
                tar_table['D'][CA_cnt].value = split_text_cpy[q]
            elif num==1:
                tar_table['E'][CA_cnt].value = split_text_cpy[q]
            elif num==2:
                tar_table['F'][CA_cnt].value = split_text_cpy[q]
            num+=1
```

그림29. CA 종류에 따라 Downlink SCCs를 결정하는 코드 중 일부

전체 코드 중 일부이다. 새로운 폼을 구성하기 위해서 이미 만들어진 결과 값 들 중에서 의미 있는 애들만 골라서 새롭게 배열한다. 이 과정에서 Reverse와 Restriction을 고려해 코드를 작성하고 같은 CC를 가지고 있다면 그 결과가 겹치지 않게 구성하는 알고리즘 구성에 상당한 애를 먹었다.

6. Result Overview

실제 데이터를 프로그램에 넣으면 어떤 결과가 나오는지를 확인해 보았다.

Index	2CC	Restriction	Cover	Reverse	Index	3CC	Restriction	Cover	Reverse	Index	4CC	Restriction	Cover	Reverse	Index	5CC	Restriction	Cover	Reverse
2CC #1	2A-2A			Yes	3CC #1	2A-2A-4A			Yes	4CC #1	2A-2A-5A-30A				5CC #1	2A-2A-46D			
2CC #2	2A-4A			Yes	3CC #2	2A-2A-5A			Yes	4CC #2	2A-2A-5A-66A				5CC #2	2A-5B-30A-66A			
2CC #3	2A-5A			Yes	3CC #3	2A-2A-12A			Yes	4CC #3	2A-2A-12A-66A				5CC #3	2A-5B-66A-66A			
2CC #4	2A-7A			Yes	3CC #4	2A-2A-13A			Yes	4CC #4	2A-2A-13A-66A				5CC #4	2A-46D-66A			
2CC #5	2A-12A			Yes	3CC #5	2A-2A-14A			Yes	4CC #5	2A-2A-29A-30A				5CC #5	5B-30A-66A-66A			
2CC #6	2A-13A			Yes	3CC #6	2A-2A-29A			Yes	4CC #6	2A-2A-66A-66A				5CC #6	41C-41D			
2CC #7	2A-14A			Yes	3CC #7	2A-2A-30A			Yes	4CC #7	2A-5A-30A-66A				5CC #7	66A-66A-46D			
2CC #8	2A-29A			Yes	3CC #8	2A-2A-66A			Yes	4CC #8	2A-5A-66A-66A				5CC #8				
2CC #9	2A-30A			Yes	3CC #9	2A-2A-71A			Yes	4CC #9	2A-5A-66B				5CC #9				
2CC #10	2A-66A			Yes	3CC #10	2A-4A-4A			Yes	4CC #10	2A-5A-66C				5CC #10				
2CC #11	2A-71A			Yes	3CC #11	2A-4A-5A			Yes	4CC #11	2A-5B-30A				5CC #11				
2CC #12	2C			Yes	3CC #12	2A-4A-7A			Yes	4CC #12	2A-5B-66A				5CC #12				
2CC #13	4A-2A			Yes	3CC #13	2A-4A-12A			Yes	4CC #13	2A-12A-30A-66A				5CC #13				
2CC #14	4A-4A			Yes	3CC #14	2A-4A-13A			Yes	4CC #14	2A-12A-66A-66A				5CC #14				
2CC #15	4A-5A			Yes	3CC #15	2A-4A-29A			Yes	4CC #15	2A-13A-66A-66A				5CC #15				
2CC #16	4A-7A			Yes	3CC #16	2A-4A-30A			Yes	4CC #16	2A-13A-66B				5CC #16				
2CC #17	4A-12A			Yes	3CC #17	2A-4A-71A			Yes	4CC #17	2A-13A-66C				5CC #17				
2CC #18	4A-13A			Yes	3CC #18	2A-5A-30A			Yes	4CC #18	2A-29A-30A-66A				5CC #18				
2CC #19	4A-29A			Yes	3CC #19	2A-5A-66A			Yes	4CC #19	2A-46A-46C				5CC #19				
2CC #20	4A-30A			Yes	3CC #20	2A-7A-7A			Yes	4CC #20	2A-46C-66A				5CC #20				
2CC #21	4A-71A			Yes	3CC #21	2A-7A-12A			Yes	4CC #21	2A-46D				5CC #21				
2CC #22	5A-2A			Yes	3CC #22	2A-5B			Yes	4CC #22	4A-46A-46C				5CC #22				
2CC #23	5A-4A			Yes	3CC #23	2A-7C			Yes	4CC #23	4A-46D				5CC #23				
2CC #24	5A-5A			Yes	3CC #24	2A-12A-30A			Yes	4CC #24	5A-30A-66A-66A				5CC #24				
2CC #25	5A-7A			Yes	3CC #25	2A-12A-66A			Yes	4CC #25	5B-30A-66A				5CC #25				
2CC #26	5A-25A			Yes	3CC #26	2A-12B			Yes	4CC #26	5B-66A-66A				5CC #26				
2CC #27	5A-30A			Yes	3CC #27	2A-13A-66A			Yes	4CC #27	5C-66A-66A				5CC #27				
2CC #28	5A-66A			Yes	3CC #28	2A-14A-30A			Yes	4CC #28	12A-30A-66A-66A				5CC #28				
2CC #29	5B			Yes	3CC #29	2A-14A-66A			Yes	4CC #29	13A-46D				5CC #29				
2CC #30	7A-2A			Yes	3CC #30	2A-29A-30A			Yes	4CC #30	25A-41D				5CC #30				
2CC #31	7A-4A			Yes	3CC #31	2A-29A-66A			Yes	4CC #31	41A-41D				5CC #31				
2CC #32	7A-5A			Yes	3CC #32	2A-30A-66A			Yes	4CC #32	41C-41C				5CC #32				
2CC #33	7A-7A			Yes	3CC #33	2A-66A-66A			Yes	4CC #33	41E				5CC #33				
2CC #34	7A-12A			Yes	3CC #34	2A-66A-71A			Yes	4CC #34	66A-46A-46C				5CC #34				
2CC #35	7A-66A			Yes	3CC #35	2A-66B			Yes	4CC #35	66A-46D				5CC #35				
2CC #36	12A-2A			Yes	3CC #36	2A-66C			Yes	4CC #36	2A-2A-30A-29A				5CC #36				
2CC #37	12A-4A			Yes	3CC #37	2C-66A			Yes	4CC #37	46D-13A				5CC #37				
2CC #38	12A-7A			Yes	3CC #38	4A-2A-2A			Yes	4CC #38	13A-2A-66C				5CC #38				
2CC #39	12A-25A			Yes	3CC #39	4A-2A-5A			Yes	4CC #39					5CC #39				
2CC #40	12A-30A			Yes	3CC #40	4A-2A-7A			Yes	4CC #40					5CC #40				
2CC #41	12A-66A			Yes	3CC #41	4A-2A-12A			Yes	4CC #41					5CC #41				
2CC #42	12B			Yes	3CC #42	4A-2A-13A			Yes	4CC #42					5CC #42				
2CC #43	13A-2A			Yes	3CC #43	4A-2A-29A			Yes	4CC #43					5CC #43				
2CC #44	13A-4A			Yes	3CC #44	4A-2A-30A			Yes	4CC #44					5CC #44				
2CC #45	13A-66A			Yes	3CC #45	4A-2A-71A			Yes	4CC #45					5CC #45				

그림30. Basic Table의 예시 (제조사의 제공 테이블)

그림30과 같은 **Basic Table**에 해당하는 엑셀 파일의 data가 작성된 코드를 수행함으로써 아래 그림31과 같이 중복 데이터를 제거한 형태가 된다.

2CC #67	Same		2CC #10	Yes	3CC #67	Same		3CC #55
2CC #68	Same		2CC #28	Yes	3CC #68	5A-5A-66A		
2CC #69	Same		2CC #35	Yes	3CC #69	5A-30A-66A		
2CC #70	Same		2CC #41	Yes	3CC #70	5A-66A-66A		
2CC #71	Same		2CC #45	Yes	3CC #71	5A-66B		
2CC #72	Same		2CC #48	Yes	3CC #72	5A-66C		
2CC #73	66A-29A			Yes	3CC #73	Same		3CC #22
2CC #74	Same		2CC #62	Yes	3CC #74	Same		3CC #54
2CC #75	66A-66A			Yes	3CC #75	5B-30A		
2CC #76	66A-71A			Yes	3CC #76	5B-66A		
2CC #77	66B			Yes	3CC #77	Same		3CC #12
2CC #78	66C			Yes	3CC #78	Same		3CC #21
2CC #79	Same		2CC #11	Yes	3CC #79	7A-2A-66A		
2CC #80	Same		2CC #21	Yes	3CC #80	Same		3CC #48

그림31. 동일한 CC내 중복된 CA 조합 제거

Index	CC	Restriction	Cover	Reverse	Index.1	CC.1	estriction.	Cover.1	Reverse.1	Index.2	CC.2	estriction.	Cover.2	Reverse.2	Index.3	CC.3	estriction.	Cover.3	Reverse.3
Index	2CC	Restriction	Cover	Reverse	Index	3CC	Restriction	Cover	Reverse	Index	4CC	Restriction	Cover	Reverse	Index	5CC	Restriction	Cover	Reverse
2CC #1	2A-2A		5CC #1	Yes	3CC #1	2A-2A-4A			Yes	4CC #1	2A-2A-5A-30A			Yes	5CC #1	2A-2A-46D			Yes
2CC #2	2A-4A		3CC #17	Yes	3CC #2	2A-2A-5A		4CC #2	Yes	4CC #2	2A-2A-5A-66A			Yes	5CC #2	2A-5B-30A-66A			Yes
2CC #3	2A-5A		4CC #10	Yes	3CC #3	2A-2A-12A		4CC #3	Yes	4CC #3	2A-2A-12A-66A			Yes	5CC #3	2A-5B-66A-66A			Yes
2CC #4	2A-7A		3CC #79	Yes	3CC #4	2A-2A-13A		4CC #4	Yes	4CC #4	2A-2A-13A-66A			Yes	5CC #4	2A-46D-66A			Yes
2CC #5	2A-12A		4CC #14	Yes	3CC #5	2A-2A-14A			Yes	4CC #5	2A-2A-29A-30A			Yes	5CC #5	5B-30A-66A-66A			Yes
2CC #6	2A-13A		4CC #17	Yes	3CC #6	2A-2A-29A		4CC #5	Yes	4CC #6	2A-2A-66A-66A			Yes	5CC #6	41C-41D			Yes
2CC #7	2A-14A		3CC #29	Yes	3CC #7	2A-2A-30A		4CC #5	Yes	4CC #7	2A-5A-30A-66A			Yes	5CC #7	66A-66A-46D			Yes
2CC #8	2A-29A		4CC #18	Yes	3CC #8	2A-2A-66A		4CC #6	Yes	4CC #8	2A-5A-66A-66A			Yes	5CC #8				
2CC #9	2A-30A		5CC #2	Yes	3CC #9	2A-2A-71A			Yes	4CC #9	2A-5A-66B			Yes	5CC #9				
2CC #10	2A-66A		5CC #4	Yes	3CC #10	2A-4A-4A			Yes	4CC #10	2A-5A-66C			Yes	5CC #10				
2CC #11	2A-71A		3CC #34	Yes	3CC #11	2A-4A-5A			Yes	4CC #11	2A-5B-30A		5CC #2	Yes	5CC #11				
2CC #12	2C		3CC #37	Yes	3CC #12	2A-4A-7A			Yes	4CC #12	2A-5B-66A		5CC #3	Yes	5CC #12				
2CC #13	Same		2CC #2	Yes	3CC #13	2A-4A-12A			Yes	4CC #13	2A-12A-30A-66A			Yes	5CC #13				
2CC #14	4A-4A		3CC #168	Yes	3CC #14	2A-4A-13A			Yes	4CC #14	2A-12A-66A-66A			Yes	5CC #14				
2CC #15	4A-5A		3CC #55	Yes	3CC #15	2A-4A-29A			Yes	4CC #15	2A-13A-66A-66A			Yes	5CC #15				
2CC #16	4A-7A		3CC #58	Yes	3CC #16	2A-4A-30A			Yes	4CC #16	2A-13A-66B			Yes	5CC #16				
2CC #17	4A-12A		3CC #59	Yes	3CC #17	2A-4A-71A			Yes	4CC #17	2A-13A-66C			Yes	5CC #17				
2CC #18	4A-13A		3CC #50	Yes	3CC #18	2A-5A-30A		4CC #7	Yes	4CC #18	2A-29A-30A-66A			Yes	5CC #18				
2CC #19	4A-29A		3CC #61	Yes	3CC #19	2A-5A-66A		4CC #8	Yes	4CC #19	2A-46A-46C			Yes	5CC #19				
2CC #20	4A-30A		3CC #61	Yes	3CC #20	2A-7A-7A			Yes	4CC #20	2A-46C-66A			Yes	5CC #20				
2CC #21	4A-71A		3CC #52	Yes	3CC #21	2A-7A-12A			Yes	4CC #21	2A-46D		5CC #4	Yes	5CC #21				
2CC #22	Same		2CC #3	Yes	3CC #22	2A-5B		5CC #3	Yes	4CC #22	4A-46A-46C			Yes	5CC #22				
2CC #23	Same		2CC #15	Yes	3CC #23	2A-7C			Yes	4CC #23	4A-46D			Yes	5CC #23				
2CC #24	5A-5A		3CC #68	Yes	3CC #24	2A-12A-30A		4CC #13	Yes	4CC #24	5A-30A-66A-66A			Yes	5CC #24				
2CC #25	5A-7A			Yes	3CC #25	2A-12A-66A		4CC #14	Yes	4CC #25	5B-30A-66A		5CC #5	Yes	5CC #25				
2CC #26	5A-25A			Yes	3CC #26	2A-12B			Yes	4CC #26	5B-66A-66A		5CC #5	Yes	5CC #26				
2CC #27	5A-30A		4CC #24	Yes	3CC #27	2A-13A-66A		4CC #15	Yes	4CC #27	5C-66A-66A			Yes	5CC #27				
2CC #28	5A-66A		4CC #24	Yes	3CC #28	2A-14A-30A			Yes	4CC #28	12A-30A-66A-66A			Yes	5CC #28				
2CC #29	5B		5CC #5	Yes	3CC #29	2A-14A-66A			Yes	4CC #29	13A-46D			Yes	5CC #29				
2CC #30	Same		2CC #4	Yes	3CC #30	2A-29A-30A		4CC #18	Yes	4CC #30	25A-41D			Yes	5CC #30				
2CC #31	Same		2CC #16	Yes	3CC #31	2A-29A-66A		4CC #18	Yes	4CC #31	41A-41D			Yes	5CC #31				
2CC #32	Same		2CC #25	Yes	3CC #32	2A-30A-66A		5CC #2	Yes	4CC #32	41C-41C			Yes	5CC #32				
2CC #33	7A-7A		3CC #56	Yes	3CC #33	2A-66A-66A		5CC #3	Yes	4CC #33	41E			Yes	5CC #33				
2CC #34	7A-12A		3CC #58	Yes	3CC #34	2A-66A-71A			Yes	4CC #34	66A-46A-46C			Yes	5CC #34				
2CC #35	7A-66A		3CC #79	Yes	3CC #35	2A-66B		4CC #16	Yes	4CC #35	66A-46D		5CC #7	Yes	5CC #35				
2CC #36	Same		2CC #5	Yes	3CC #36	2A-66C		4CC #17	Yes	4CC #36					5CC #36				
2CC #37	Same		2CC #17	Yes	3CC #37	2C-66A			Yes	4CC #37					5CC #37				
2CC #38	Same		2CC #34	Yes	3CC #38	Same		3CC #1	Yes	4CC #38					5CC #38				
2CC #39	12A-25A			Yes	3CC #39	Same		3CC #11	Yes	4CC #39					5CC #39				
2CC #40	12A-30A		4CC #28	Yes	3CC #40	Same		3CC #12	Yes	4CC #40					5CC #40				
2CC #41	12A-66A		4CC #28	Yes	3CC #41	Same		3CC #13	Yes	4CC #41					5CC #41				
2CC #42	12B		3CC #84	Yes	3CC #42	Same		3CC #14	Yes	4CC #42					5CC #42				
2CC #43	Same		2CC #6	Yes	3CC #43	Same		3CC #15	Yes	4CC #43					5CC #43				
2CC #44	Same		2CC #18	Yes	3CC #44	Same		3CC #16	Yes	4CC #44					5CC #44				
2CC #45	13A-66A		4CC #15	Yes	3CC #45	Same		3CC #17	Yes	4CC #45					5CC #45				
2CC #46	Same		2CC #7	Yes	3CC #46	Same		3CC #10	Yes	4CC #46					5CC #46				
2CC #47	14A-30A		3CC #111	Yes	3CC #47	4A-4A-5A			Yes	4CC #47					5CC #47				
2CC #48	14A-66A		3CC #112	Yes	3CC #48	4A-4A-7A			Yes	4CC #48					5CC #48				
2CC #49	Same		2CC #26	Yes	3CC #49	4A-4A-12A			Yes	4CC #49					5CC #49				
2CC #50	Same		2CC #39	Yes	3CC #50	4A-4A-13A			Yes	4CC #50					5CC #50				

그림32. 서로 다른 CC간의 가능한 Coverage CA 조합까지 표시한 **Target Table**

다음과 같이 실질적으로 측정해야 할 CA 조합을 제외하고 cell의 색깔이 다르게 나온다. 즉, 블록 처리를 색깔로 표현하였다. 실질적으로 SAR팀에서는 하얀 마킹이 되어 있는 부분을 검증하면 된다.

no	E-UTRA CA	Bands				UL					DL								
		PCC	SCC1	SCC2	SCC3	pcc					PCC			SCC1			SCC2		
		1st	2nd	3rd	4th	mode	BW	channel	Freq	RB/offset	BW	Channel	Freq	BW	Channel	Freq	BW	Channel	Freq
no.1	5A-7A	5A	7A								10	2525	881.5						
no.2	5A-7A	7A	5A								20	2850	2630						
no.3	5A-25A	5A	25A								10	2525	881.5						
no.4	5A-25A	25A	5A								20	8365	1962.5						
no.5	12A-25A	12A	25A								10	5095	737.5						
no.6	12A-25A	25A	12A								20	8365	1962.5						
no.7	25A-41A	41A	25A								20	40620	2593						
no.8	25A-41A	25A	41A								20	8365	1962.5						
no.9	26A-41A	26A	41A								15	8865	876.5						
no.10	26A-41A	41A	26A								20	40620	2593						
no.11	41A-41A	41A	41A								20	39750	2506						
no.12	2A-2A-4A	4A	2A	2A							20	2175	2132.5						
no.13	2A-2A-4A	2A	2A	4A							20	900	1960						
no.14	2A-2A-14A	14A	2A	2A							10	5330	763						
no.15	2A-2A-14A	2A	2A	14A							20	900	1960						
no.16	2A-2A-71A	71A	2A	2A							20	68761	634.5						
no.17	2A-2A-71A	2A	2A	71A							20	900	1960						
no.18	2A-4A-4A	4A	2A	4A							20	68761	634.5						
no.19	2A-4A-4A	2A	4A	4A							20	900	1960						
no.20	2A-4A-5A	5A	2A	4A							10	2525	881.5						
no.21	2A-4A-5A	4A	2A	5A							20	2175	2132.5						
no.22	2A-4A-5A	2A	4A	5A							20	900	1960						
no.23	2A-4A-7A	4A	2A	7A							20	2175	2132.5						
no.24	2A-4A-7A	2A	4A	7A							20	900	1960						
no.25	2A-4A-7A	7A	2A	4A							20	2850	2630						
no.26	2A-4A-12A	4A	2A	12A							20	2175	2132.5						
no.27	2A-4A-12A	2A	4A	12A							20	900	1960						
no.28	2A-4A-12A	12A	2A	4A							10	5095	737.5						

그림33. [그림32]의 데이터를 기반으로 다음과 같은 새로운 **Total Table** 생성

새로운 엑셀 Form 형태의 **Total Table** 에 실질적으로 측정할 CA 조합의 Reverse 형태까지 조합해서 한번에 보여줄 수 있게 하였다.

그 결과를 토대로 사원분들이 Uplink PCC 를 측정한 Maximum 파워를 입력하고 작성된 코드의 프로그램을 수행하면, 아래 [그림 34]와 같이 자동적으로 CA 조합에 따른 밴드 채널 주파수(Downlink SCCs)가 일괄적으로 출력되게 된다.

PCC			SCC1			SCC2		
BW	Channel	Freq	BW	Channel	Freq	BW	Channel	Freq
10	2525	881.5	20	3100	2655			
20	2850	2630	10	2525	881.5			
10	2525	881.5	20	8365	1962.5			
20	8365	1962.5	10	2525	881.5			
10	2525	880.5	10	9820	2355			
10	9820	2355	10	2525	881.5			
10	5095	737.5	20	8365	1962.5			
20	8365	1962.5	10	5095	737.5			
20	8365	1962.5	20	40620	2593			
20	39750	2506	20	8365	1962.5			
15	8865	876.5	20	40620	2593			
20	900	1960	20	1100	1980	20	2175	2132.5
20	2175	2132.5	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	10	2525	881.5
10	2525	881.5	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	10	5095	737.5
10	5095	737.5	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	10	5230	751
10	5230	751	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	10	5330	763
10	5330	763	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	10	9820	2355
10	9820	2355	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	20	66786	2145
20	67036	2170	20	900	1960	20	1100	1980
20	900	1960	20	1100	1980	20	68761	634.5
20	68761	634.5	20	900	1960	20	1100	1980
20	900	1960	20	2175	2132.5	20	2300	2145
20	2300	2145	20	2050	2120	20	900	1960
20	900	1960	20	2175	2132.5	10	2525	881.5
20	2175	2132.5	10	2525	881.5	20	900	1960
10	2525	881.5	20	900	1960	20	2175	2132.5
20	900	1960	20	2175	2132.5	20	3100	2655

그림34. **Result Table** 의 모습