객체지향 프로그래밍을 이용한 인버터 SW 구조 설계

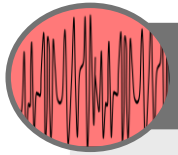
목 차

2023년 7월 4일

LS ELECTRIC

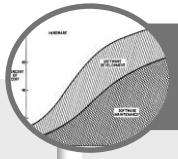
개발 배경

소프트웨어 위기 – “Sea of Software”



환경

- **IT 기술의 진화**
  - 반도체 기술의 획기적 발전
  - 인터넷/IOT/Sensor Network
- **Business 체계의 근본적 변화**
  - 비즈니스 생태계의 등장
  - 비즈니스 Speed의 증가
  - 시장의 급격한 변화



SW영향

- **HW기능의 감소/SW기능의 증가**
  - Software Size의 증가
  - Software Complexity의 증가
  - Software의 개인화/다양화
  - SW Lifecycle의 단축

결과

- **프로젝트의 예산 초과**
- **프로젝트의 지연**
- **SW 품질 저하**
- **SW 요구사항 미충족**
- **프로젝트 관리 복잡도 증가**
- **유지보수 비용의 증가**

출처

The term "software crisis" was coined by some attendees at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany.[1][2]
Edsger Dijkstra's 1972 ACM Turing Award Lecture makes reference to this same problem:[3]

1. Report about the NATO Software Engineering Conference dealing with the software crisis
2. Report on a conference sponsored by the NATO SCIENCE COMMITTEE Garmisch, Germany, 7th to 11th October 1968
3. E. W. Dijkstra Archive

개발 배경

현재 상황

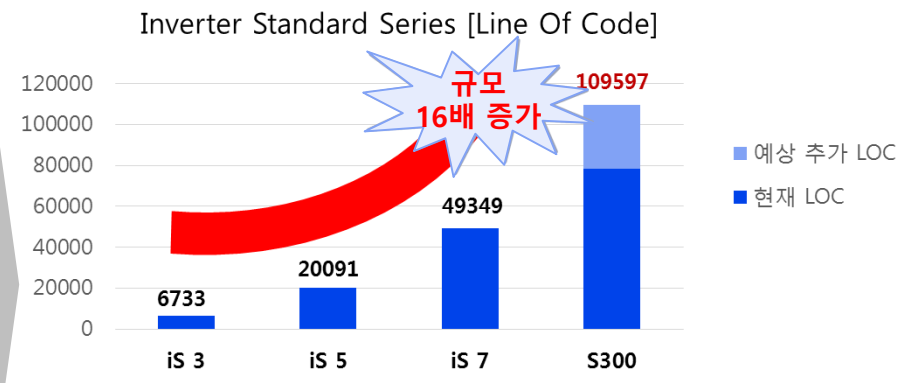
Inverter SW 위기의 도래

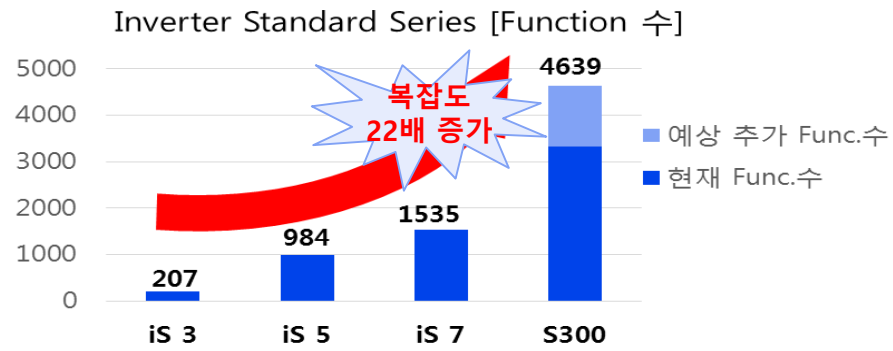
SW를 내장한 제품은 HW 제품 대비 매출 성장률이 낮고 품질 Claim 비중은 매우 높아 제품 경쟁력이 상대적으로 낮은 것으로 판단됨

기기 사업의 성장 및 경쟁력 강화의 걸림돌이 되고 있는 3가지 SW 이슈

- "기능/성능 GAP catch up 필요하며 SW 이슈가 대부분"
  - ⇨ SW 내장 제품의 경우 대다수의 기능이 SW적으로 구현되며, 이 중 다수가 경쟁사 대비 열세임
- "SW 관련 품질/신뢰성 이슈로 고객 불만 증가"
  - ⇨ 또한, HW만으로 구성된 제품 대비 품질/신뢰성 이슈가 다발하고 있으며 이로 인한 고객 불만이 누적되고 있는 상황임
- "성장 위한 line up 보강 필요하나 적기 개발 안되며 SW가 주요 Bottleneck"
  - ⇨ 사업 성장 위해 Line-up 보강 필요하나 개발지연으로 적기 대응이 안 되고 있으며 SW 부분이 주요 Bottleneck으로 판단됨

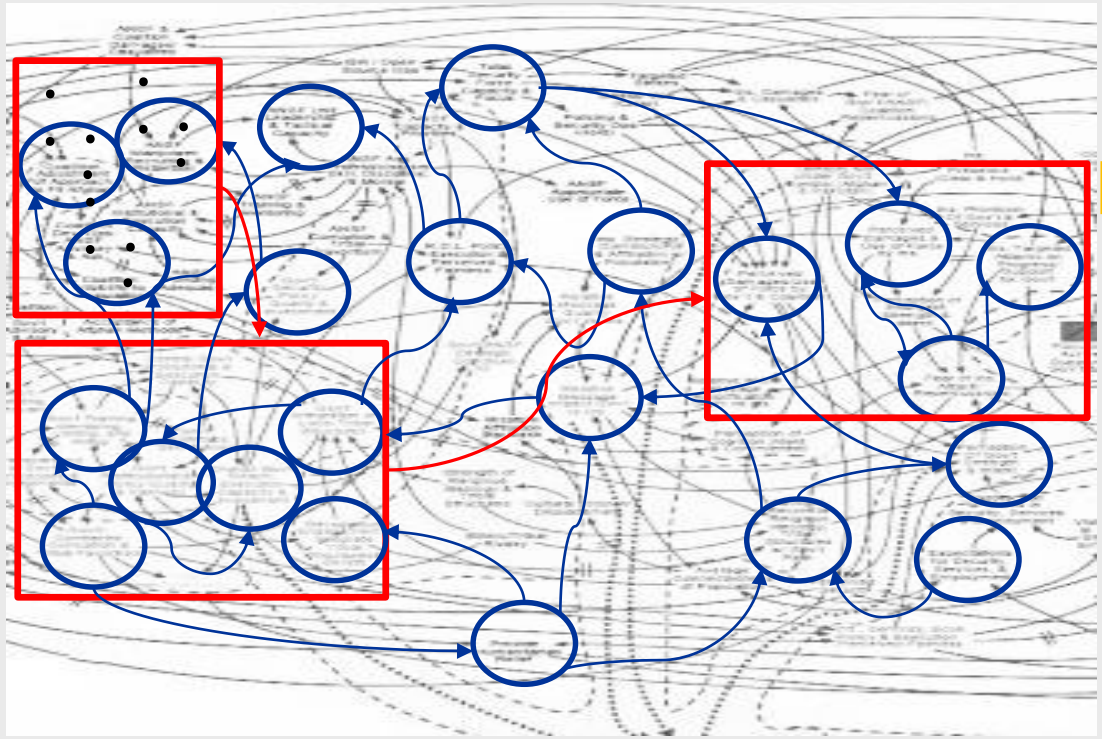
SW 제품의 크기와 복잡도가 한계치(약 8만 라인)를 넘어가면서
SW 제품 개발 및 유지보수의 생산성 및 품질의 급격한 하락

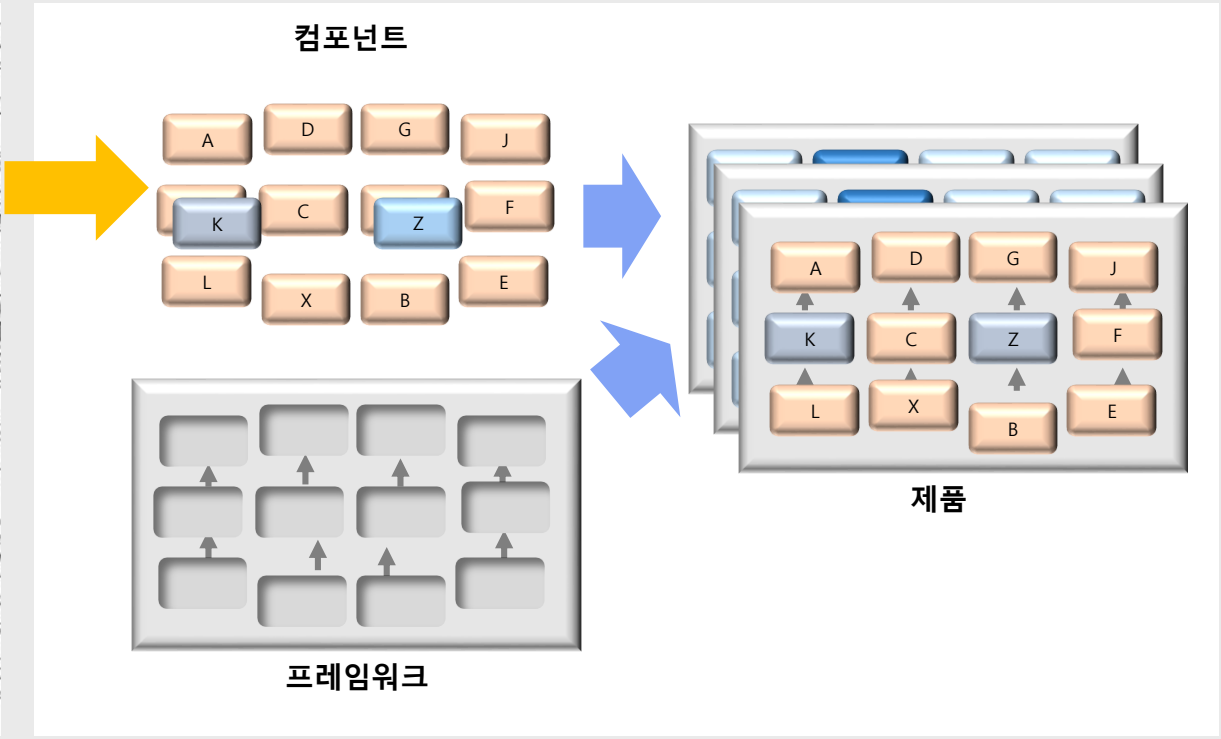




개발 배경

SW 객체화





함수 : SW의 가장 작은 수행 단위
모듈 : 1개의 기능을 수행하기 위해 함수가 모여 있는 집합체
컴포넌트 : 모듈 간의 Interface를 정의 내려주고 비슷한 기능을 하는 모듈들의 집합체.
이벤트를 처리하는 기본 단위
프레임 워크 : 제품의 형식을 정의하는 틀.
컴포넌트 간의 Interface 및 Event를 연결시켜주는 큰 SW틀

개발 배경

SW 생산 Line 구성

SW 재사용 체계

- 프로그래머 중심
  - ⇨ 단순 함수 라이브러리 재사용
- 프로젝트 중심
  - ⇨ 컴포넌트 프레임워크 구성
  - ⇨ 프로젝트간 컴포넌트 재사용
  - ⇨ 조립 생산 라인의 구축

플랫폼 기반 조립 생산 체계 구축

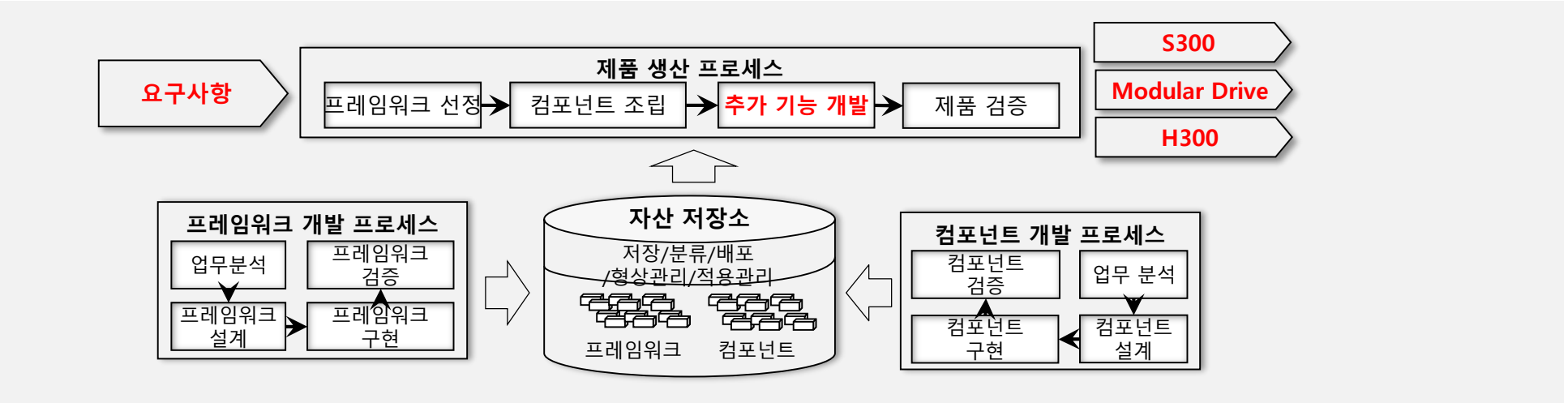
- 표준 SW 컴포넌트의 조립을 가능하게 하는 기반 프레임워크 개발

추가 기능과 가변성 관리 체계 구축

- 공통성과 가변성의 체계적 관리를 통한
- 프레임워크와 컴포넌트 기반 SW 제품의 조립 생산라인 구성

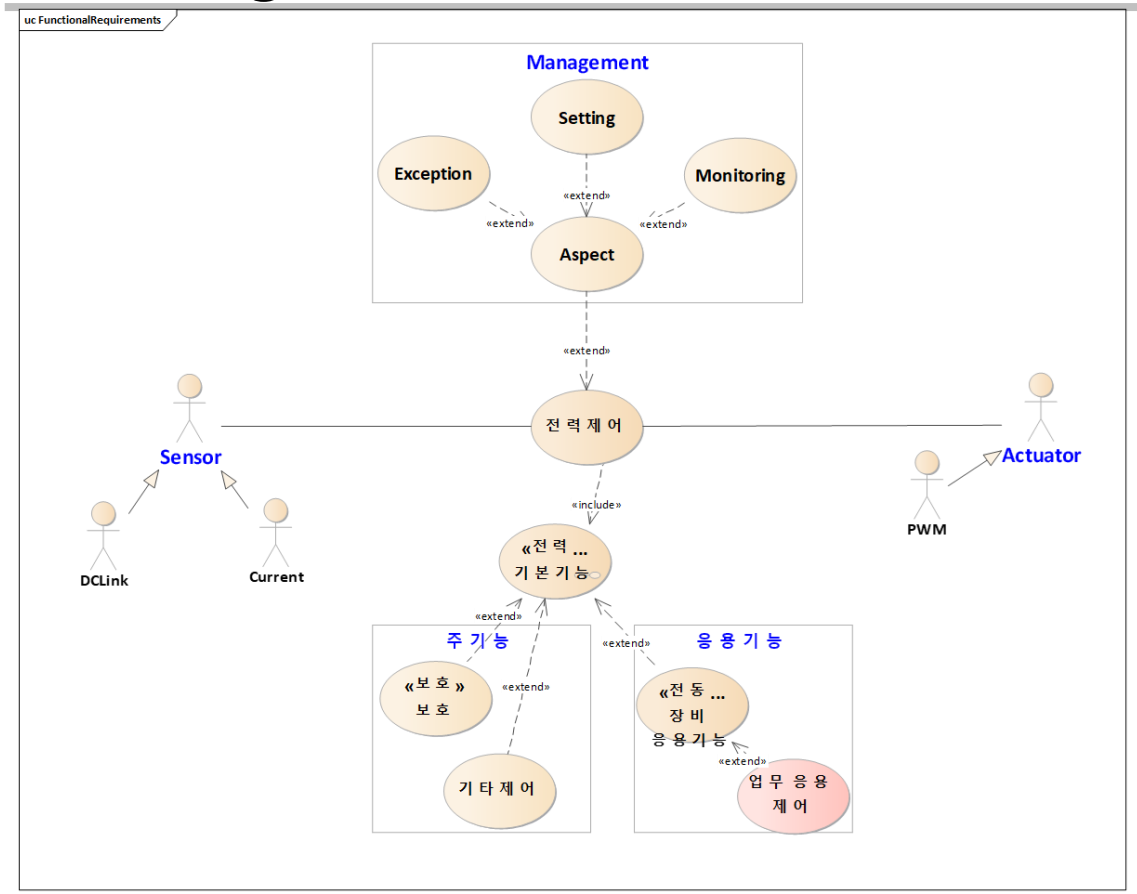
자산저장소 구축

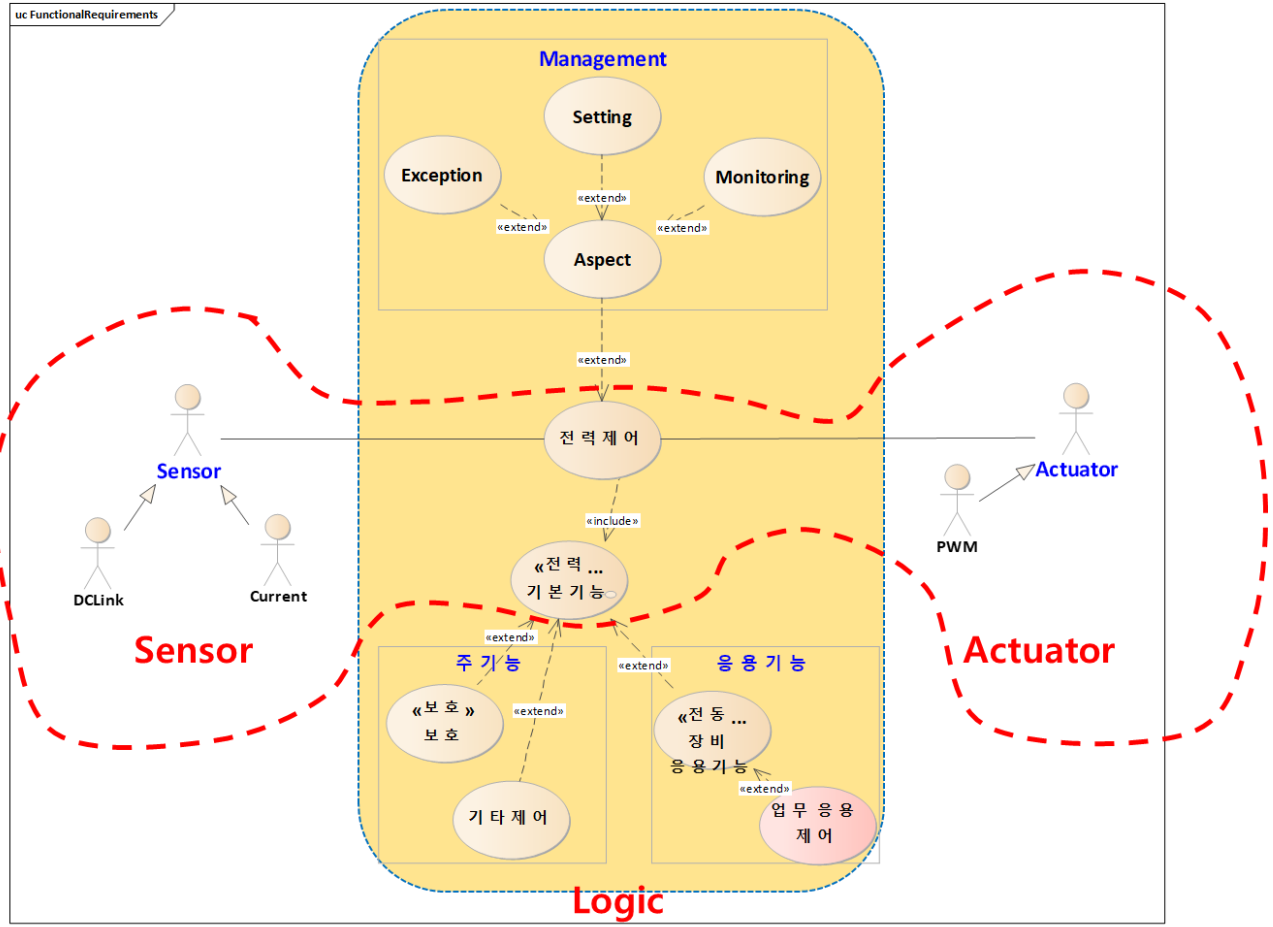
- 각 프로세스 산출물의 저장소
- 저장된 자산들은 다른 프로세스나 다른 버전의 SW 개발 시 재사용 될 수 있도록 체계적 관리



SW Diagram

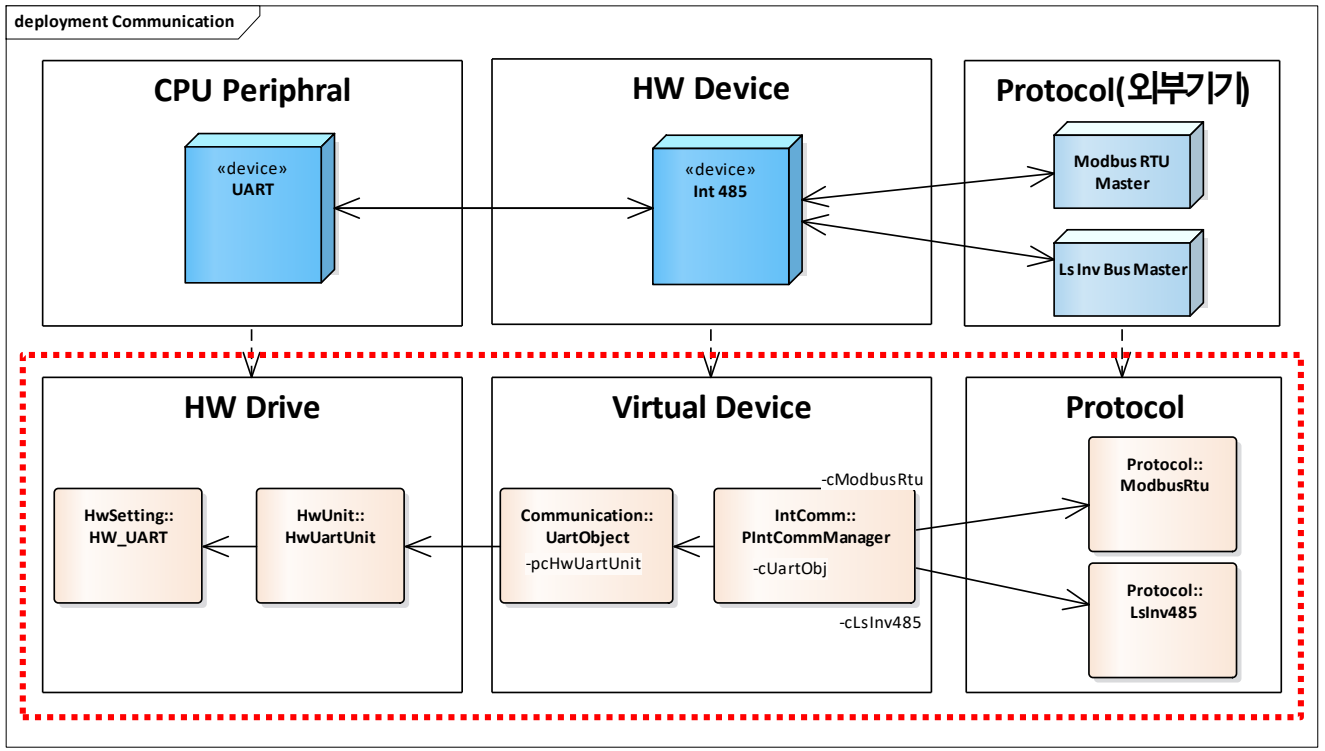
Use case 관점의 업무범위

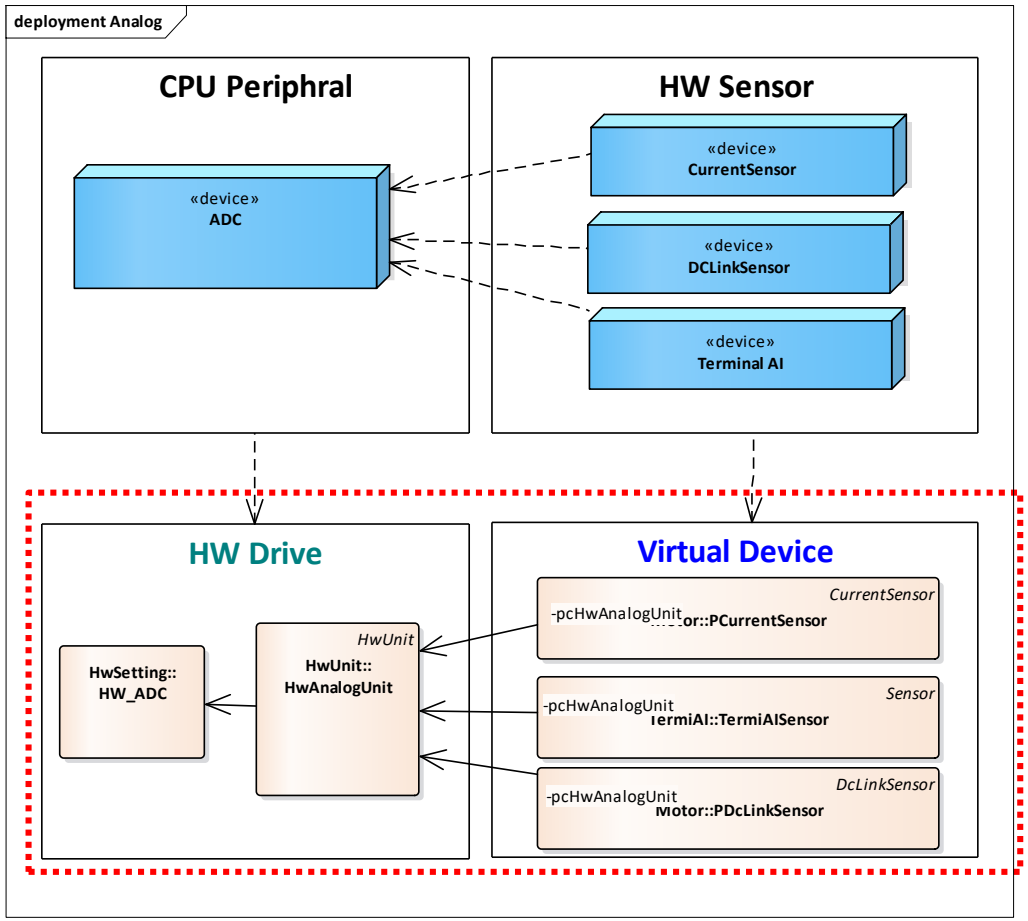




SW 객체화에 의한 재활용성

- **Deployment Diagram을 Base로 Technical Component 개발**
  - H/W의 구성에 따라 대칭 시켜 S/W 모듈 구현
  - H/W 변경에 따른 S/W Coding 최소화
    - CPU 변경 시 H/W Device만 수정
    - H/W Sensor 변경 시 Virtual Device만 수정



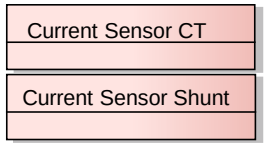


SW 객체화에 의한 재활용성

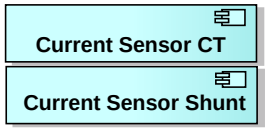
공통 모듈과 가변 모듈의 체계적 분리

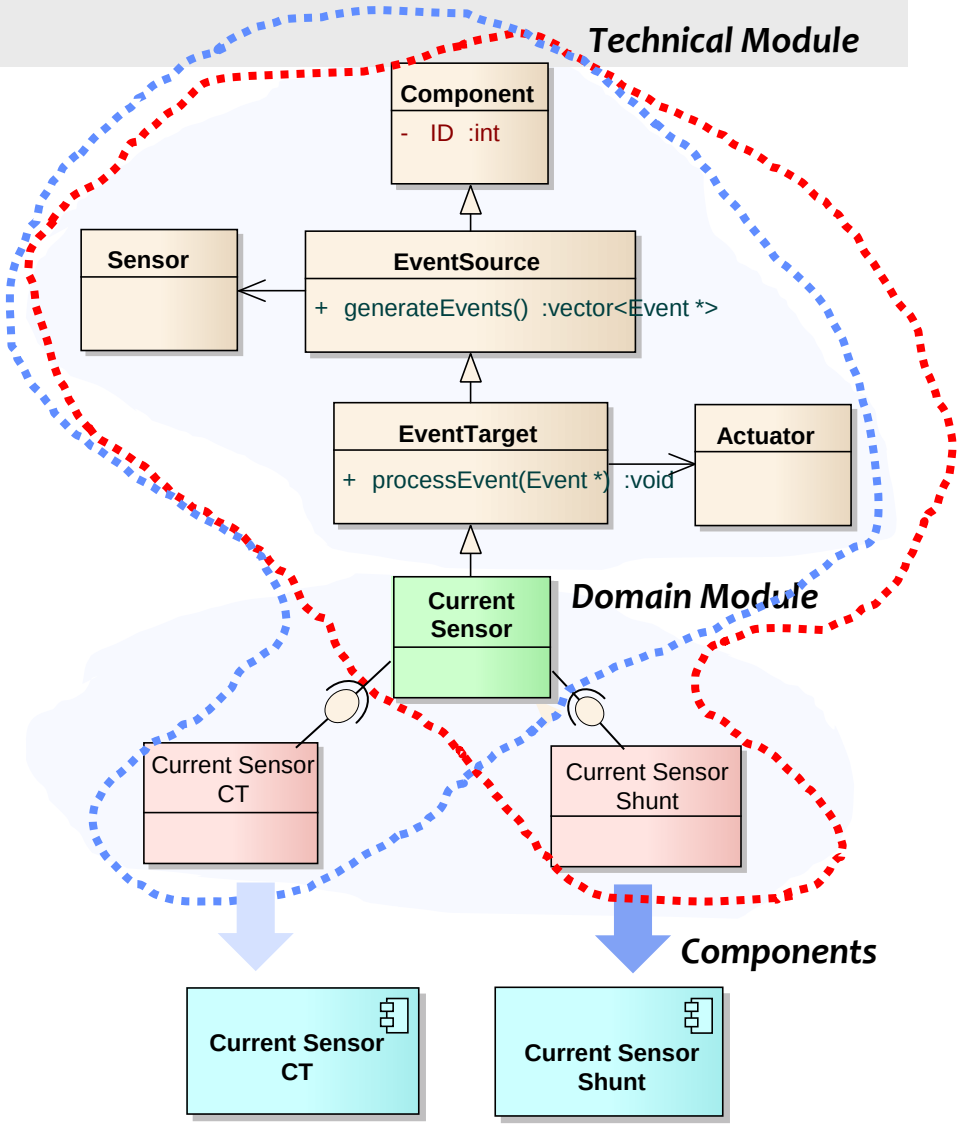
- **공통 Module 재사용/상속**
  - Technical Module
    - 기술적 공통 기능
  - Domain Module
    - 업무적 공통 기능

- **가변성 추가**
  - SOC/HW Device/OS/Data/Network의 변화
  - 업무 로직의 변화
  - HW의 변동에 따른 가변 Module 정의
    - Current Sensor CT Module
    - Current Sensor Shunt Module



- **가변 컴포넌트 생성**
  - Component Current Sensor CT
  - Component Current Sensor Shunt





SW 객체화에 의한 재활용성

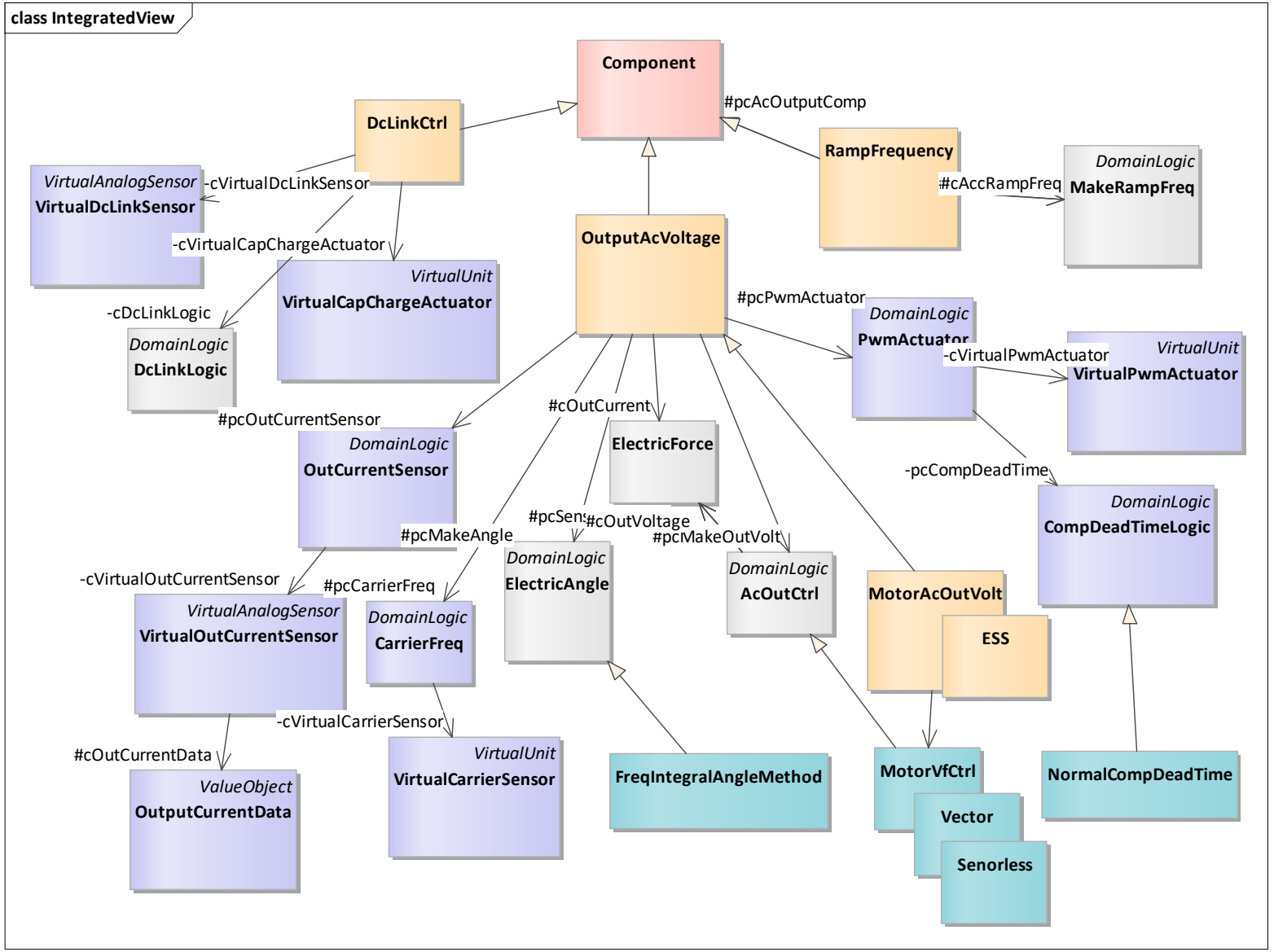
전동기 제어 Class 관계도

계층 구조

- Sensor
- Logic
- Actuator

가변성 관리

- Algorithm
- Component

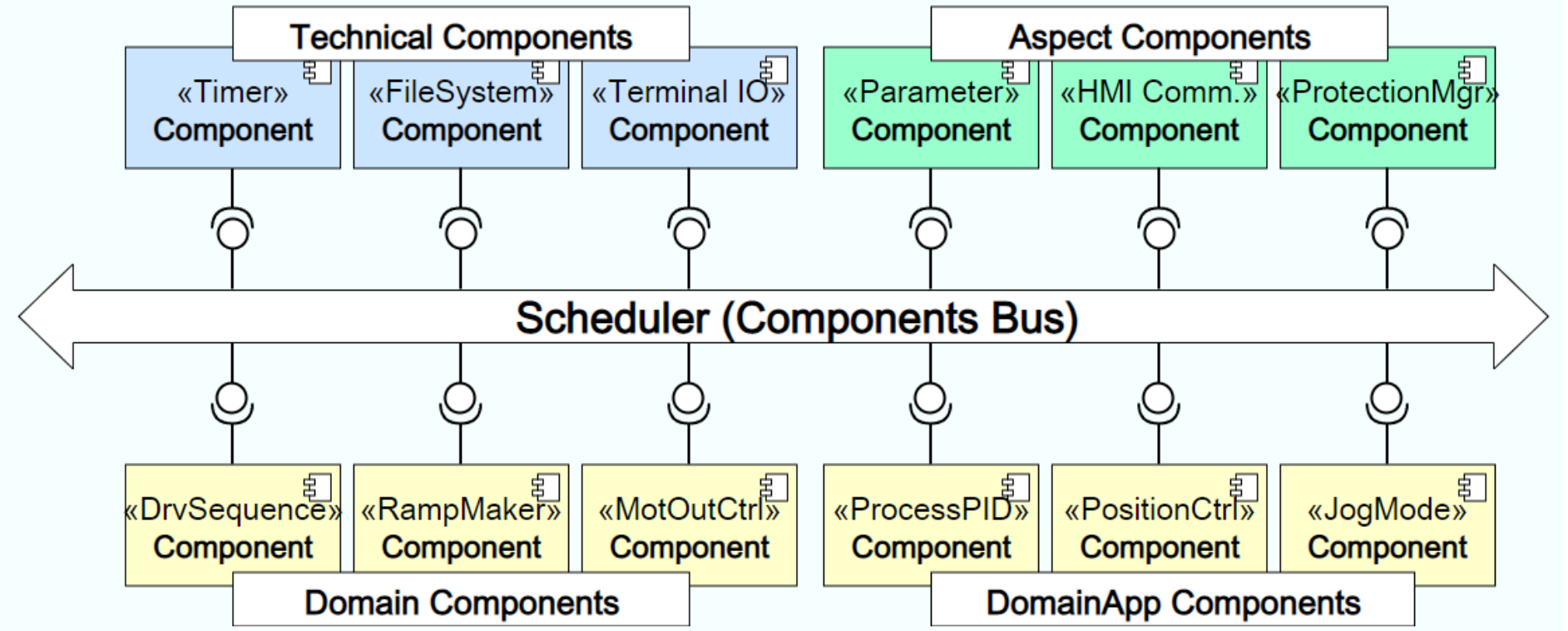


SW플랫폼 아키텍처

Event Driven

- 컴포넌트 기반 시스템 개발
  - 표준 인터페이스 기반 컴포넌트의 추가/삭제
  - 서비스 계열/계층간 상호 연동성 확보

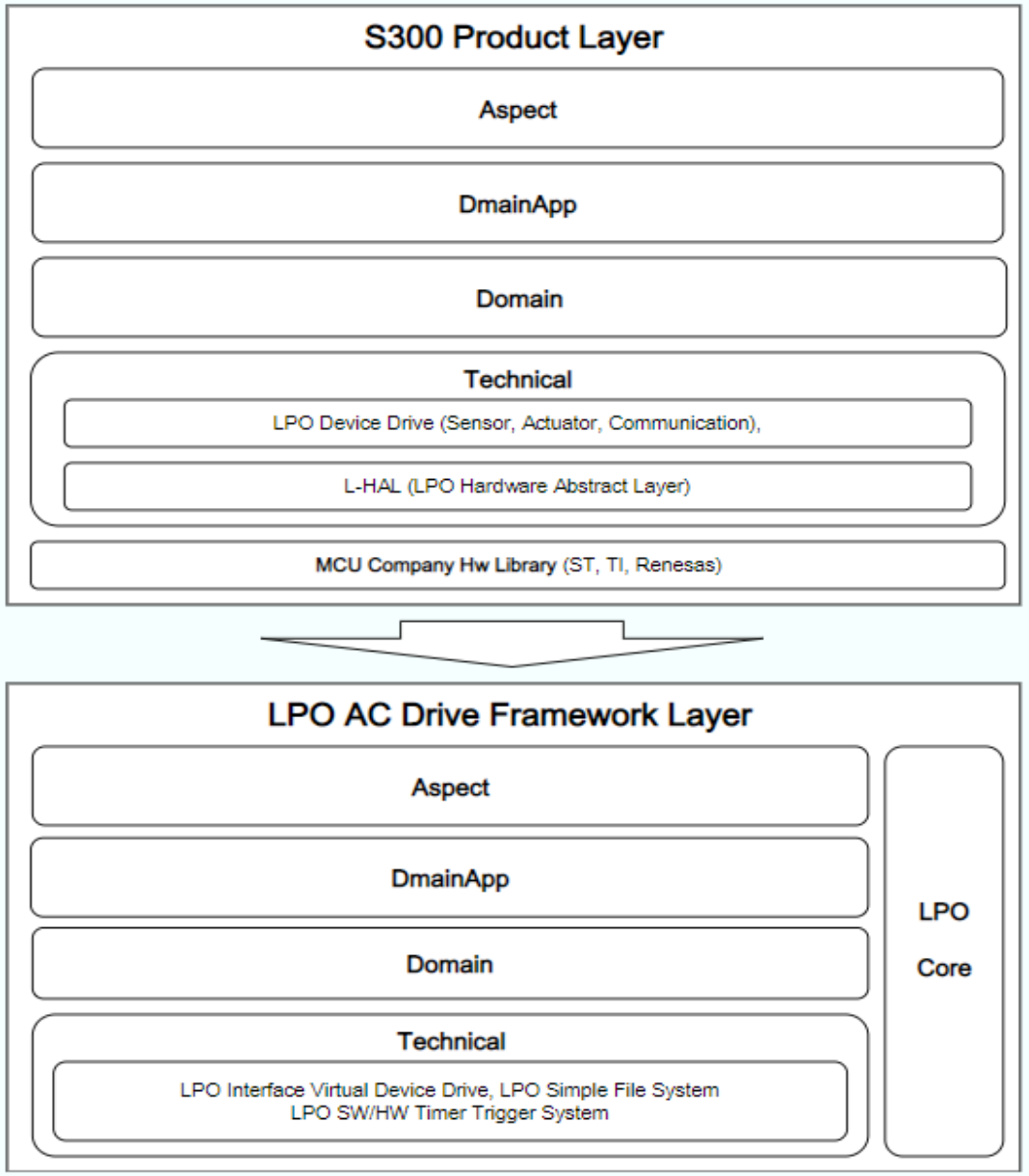
- SW 생산라인
  - 업무 내용의 변경 용이성 확보
  - 다양한 하드웨어의 이식성 확보



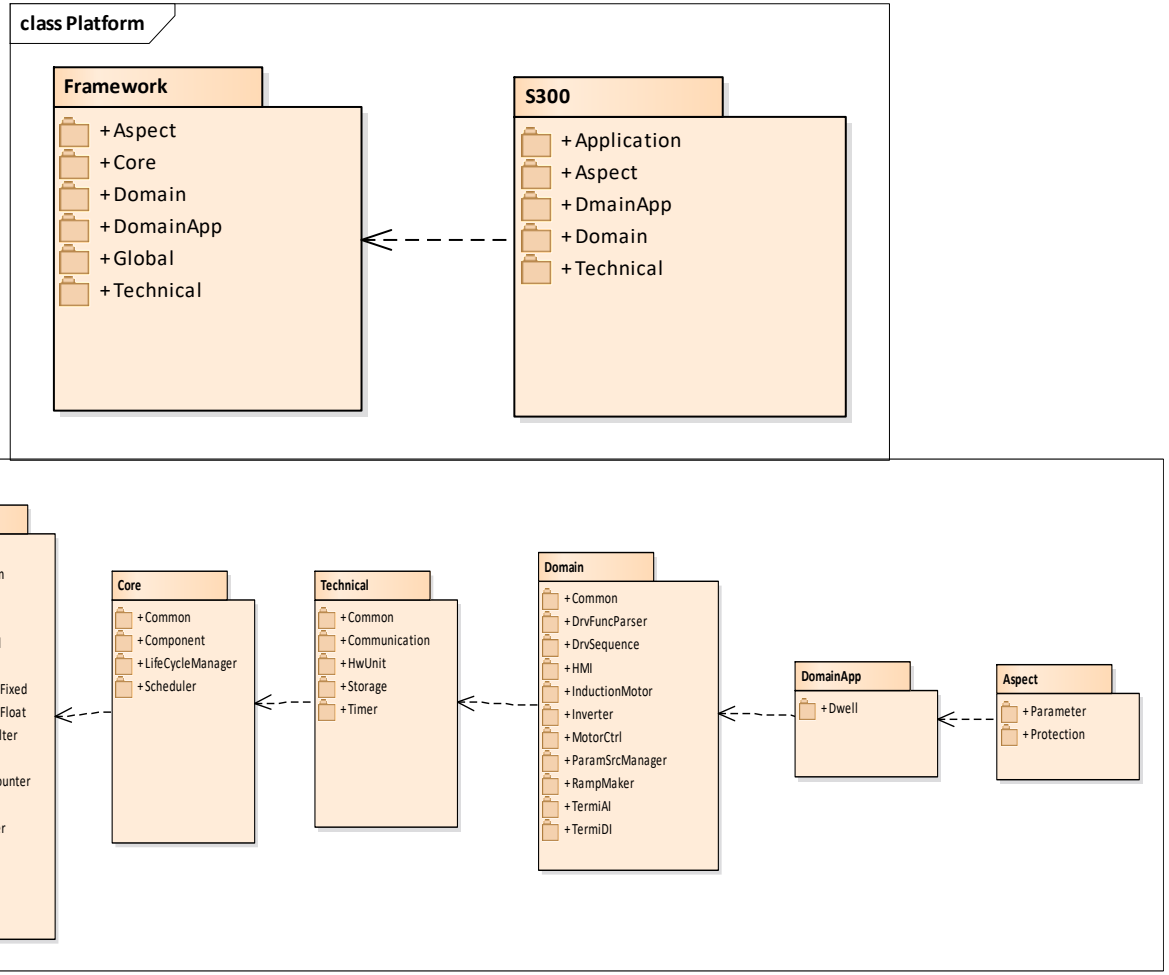
Technical Component : HW Device Drive, File System, Communication
Domain Component : 전동기 제어, Drive Sequence, Dc Link Process, PWM 출력
Domain Application Component: 전동기를 이용한 Application, HAVC, Crane, etc.
Aspect Component : Parameter, HMI, Protector 등의 Domain을 관리하는 Component

SW플랫폼 아키텍처

AC Drive Header File Include



- S300 SW 구조
  - 폴더 간의 종속 관계
  - Tree 구조의 Header File Include



결론

객체지향 프로그래밍의 장점

1. 기능의 객체화를 통해 산업용 인버터의 기능 확장에 의한 SW 복잡도를 해결 였다.
2. 객체화를 통해 SW 개발 시 업무 분담이 용의하여 생산성이 높아 졌다.
3. SW Coding 품질이 아주 높아 진다.
4. 기능 변경에 따른 SW 수정이 빨라진다.
5. Agile하게 업무를 진행하기에는 SW 객체화가 필수 적이다.
6. 스파게티 코드가 아니기 때문에 기능의 연결성이 줄어들어 BUG 발생률이 감소 하였다.
7. Interrupt까지 Event처리를 하여 Data변경에 따른 제어가 틀어지는 오류가 없어 졌다.
8. Code의 재활용성이 높아져서 Code의 안전성이 높아 졌다.

객체지향 프로그래밍의 단점

1. Embedded C++의 학습이 추가로 필요하다.
2. SW 전체를 관장하는 SW Architect가 필수적이다.
3. 객체지향 프로그래밍에 대한 이해도가 높아야 구현 가능하다.