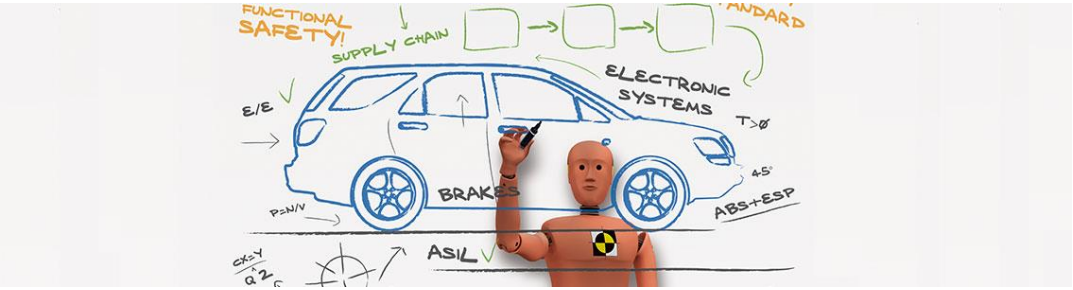
기능안전 확보를 위한 SW 측면 고려사항
- Architecture 개선 측면

DNV GL © 2017

DNV GL - Introduction



CUSTOMERS

≈100,000

ESTABLISHED

1864

OFFICES WORLDWIDE

350

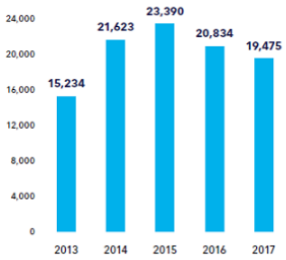
COUNTRIES

100+

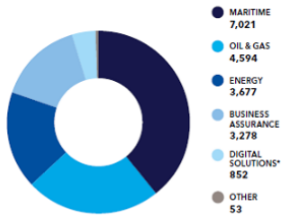
REVENUE (MILLION NOK)

19,475

LAST FIVE YEARS



PER BUSINESS AREA

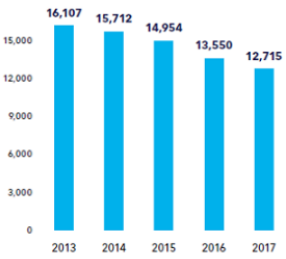


* Figures for software business only. Software became part of the new business area Digital Solutions on 1 Jan 2016.

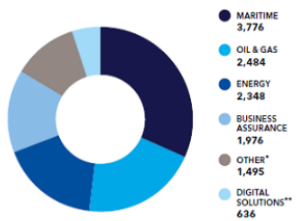
NUMBER OF EMPLOYEES

12,715

LAST FIVE YEARS

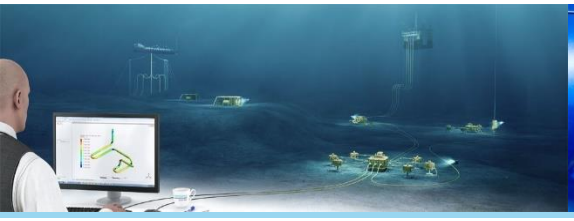


PER BUSINESS AREA



* Global shared services and Group functions.

** Figures for software business only. 763 employees in Digital Solutions when first operational on 1 Jan 2016.



DNV GL Korea - Automotive Functional Safety Business Portfolio

Concept Engineering	Vehicle Test	FuSa Management
- Item Definition - HARA (Hazard Analysis & Risk Assessment) - FSC Engineering	- Safety Validation Specification	- DIA Management - OEM Engineering Process - Supplier Management - Supplier Engineering Review (e.g. TSC)

System Engineering	Hardware Engineering	Software Engineering
- Technical Safety Concept (i.e. Safety structure, safety mechanisms design)	- Schematic Analysis (SPFM, LFM, PMHF) - MCU/ASIC Engineering Support	- SW Architecture Design - AUTOSAR/OSEK Support - SW Safety Analysis, DFA

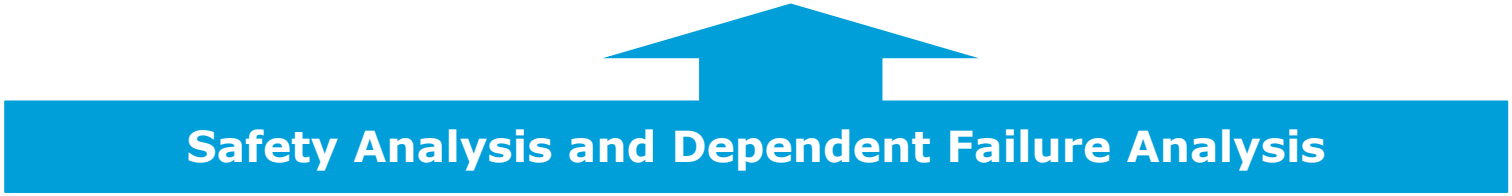
Supplier Engineering Process	3rd Party Evaluation	Training
- A-SPICE Process Consulting - ISO 26262:2018 Process Consulting	- FuSa Audit - FuSa Assessment - A-SPICE Assessment	- ISO 26262: 2018

DNV GL © 2017

Challenges for the SW Architecture

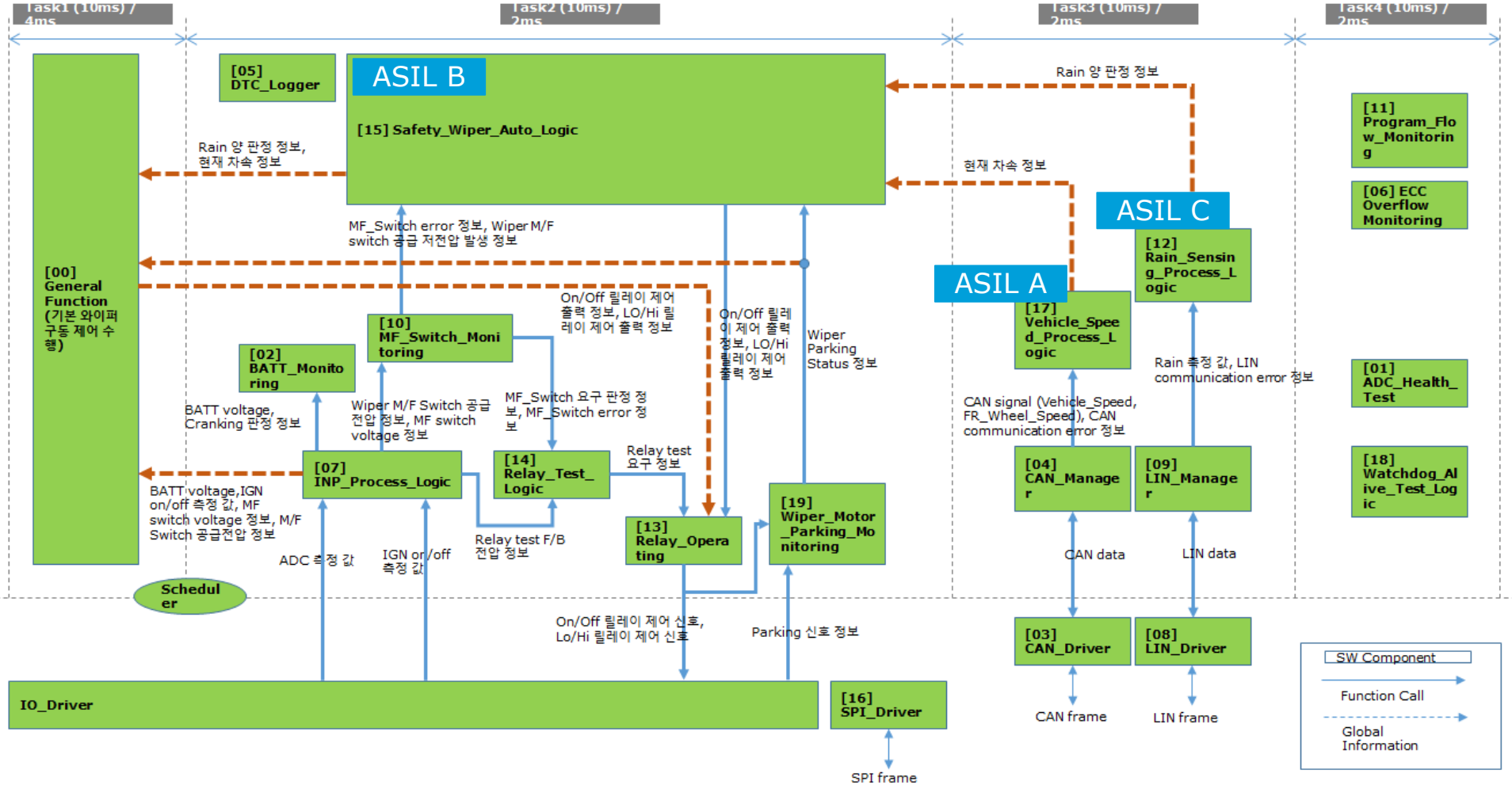
How the architecture can satisfy FuSa requirements?

- **Consider..**
  - **SSRs**
  - **Different ASILs in the same SW architecture**
  - **Sufficient and effective safety mechanisms**
  - **ASIL decomposition**
  - **Evidences**
  - **etc**



Architecture - example

Static Architecture – Data Flow (Nominal Control)



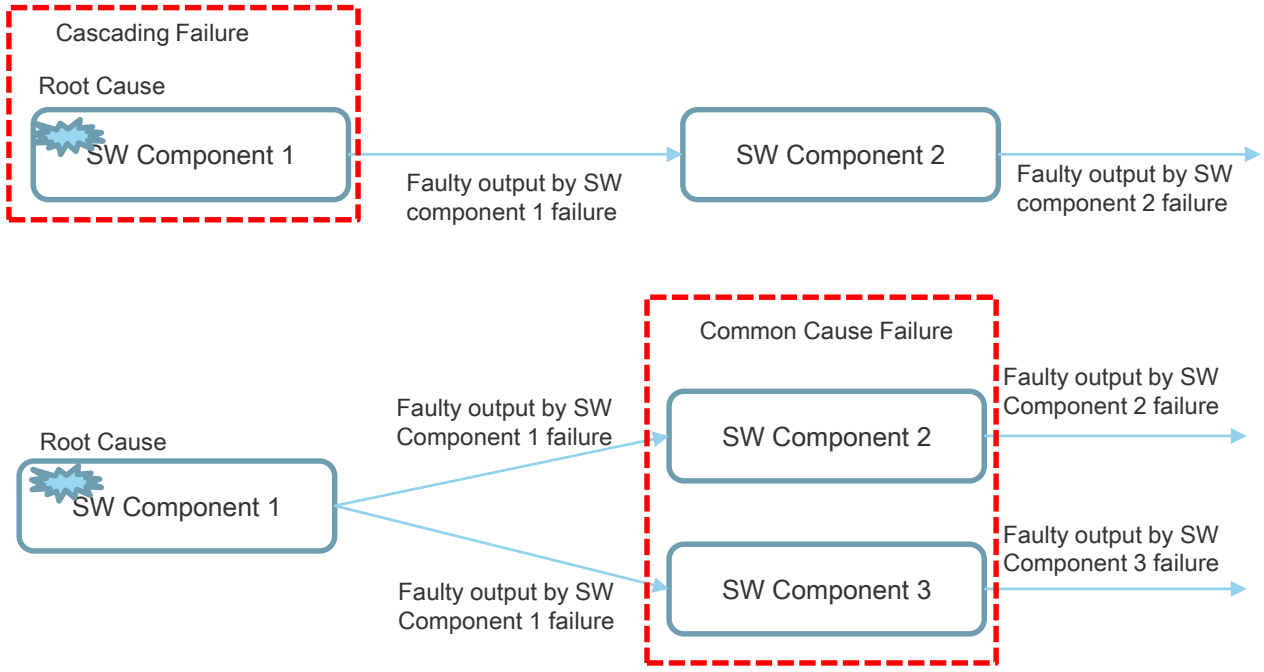
DNV GL © 2017

SW Safety Analysis - Background

Consideration	Description
구현 수준 (즉, code level)의 분석은 수행하지 않음	Variable, C-function 수준의 분석은 수행하지 않음 구현 수준의 systematic fault로 인한 영향은 SW architecture 수준에서 발생하는 각 SW component의 오동작과 대부분 중복적인 성격을 갖음. 구현 수준의 fault는 code review, analysis (e.g. static analysis, run-time analysis), testing을 통해 검출 및 수정 수행
RHF에 대한 고려는 제한적으로 수행	MCU의 processing core, memory 수준의 fault (transient fault 포함) 정도가 부분적으로 고려 될 수 있음
SW의 random fault는 고려하지 않음	SW는 자제척인 random fault를 가지고 있지 않음
Fault classification을 수행하지 않음	SPF, RF, MPF는 HW fault에만 적용되는 개념임
Failure rate estimation, failure mode distribution, diagnostic coverage과 같은 정량적인 분석을 수행하지 않음	정량적인 지표 산출 불가
HW safety mechanism이 고려 될 수 있음	특정 SW fault는 HW safety mechanism에 의해 cover될 수 있음 (e.g. MPU)
Inductive, deductive 방식에 대한 recommendation level이 없음	FTA 방식의 효과성은 널리 알려지지 않음

SW Safety Analysis and Dependent Failure Analysis - Background

Term.	Definition
Cascading Failure	Failure of an element of an item resulting from a root cause (inside or outside of the element) and then causing a failure of another element of elements of the same or different item
Common Cause Failure	Failure of two or more elements of an item resulting directly for a single specific event or root cause which is either internal or external to all of these elements



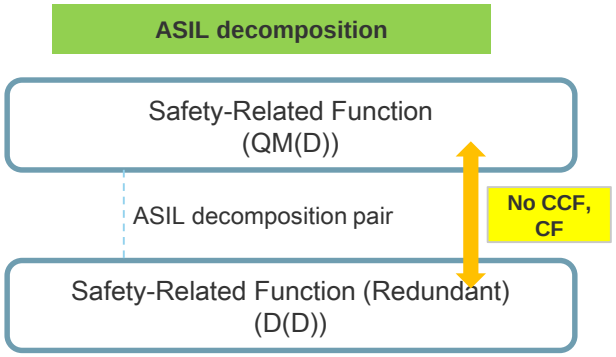
SW Safety Analysis and Dependent Failure Analysis - Background

Achievement of independence or freedom from interference between the software architectural elements can be required because of:

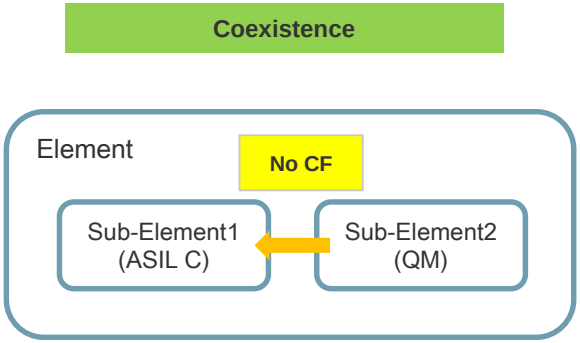
— the application of an ASIL decomposition at the software level (see 6.4.3),

— the implementation of software safety requirements (see 7.4.11), for example to provide evidence for effectiveness of the software safety mechanisms such as independence between the monitored element and the monitor, or

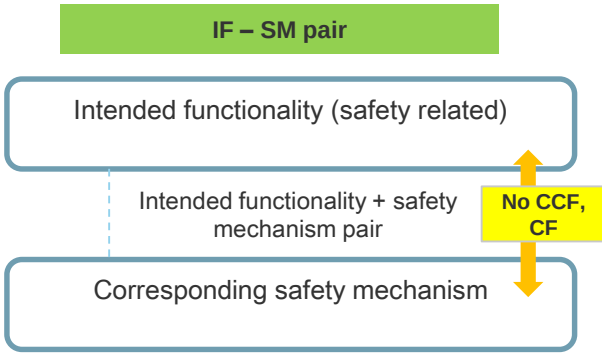
— required coexistence of the software architectural elements (see 7.4.6, 7.4.8 and 7.4.9 and ISO 26262-9:2018, Clause 6).



Case 1

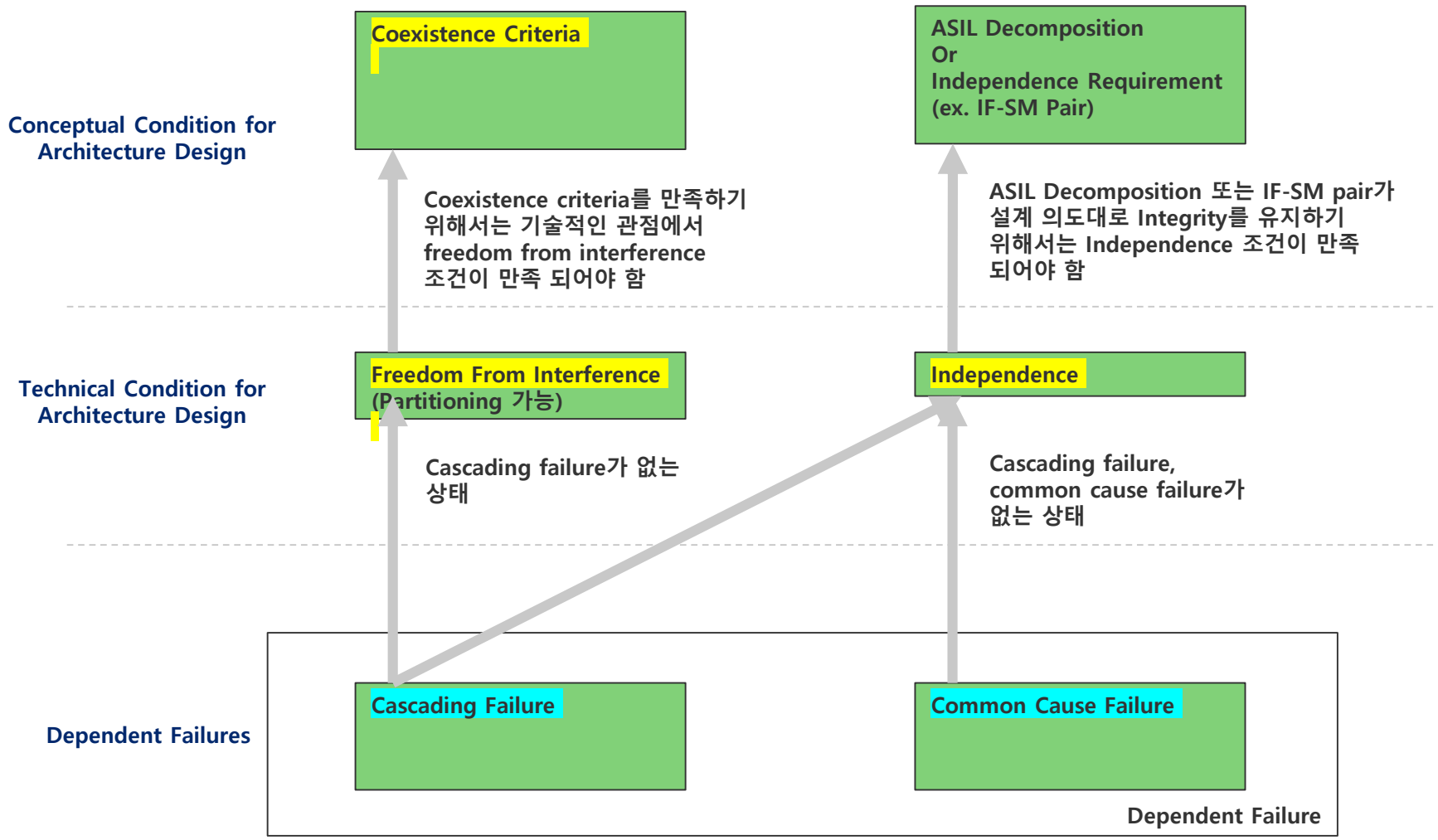


Case 2



Case 3

DFA - General


Conceptual Condition for Architecture Design
Technical Condition for Architecture Design
Dependent Failures
Coexistence Criteria
ASIL Decomposition Or Independence Requirement (ex. IF-SM Pair)
Coexistence criteria를 만족하기 위해서는 기술적인 관점에서 freedom from interference 조건이 만족 되어야 함
ASIL Decomposition 또는 IF-SM pair가 설계 의도대로 Integrity를 유지하기 위해서는 Independence 조건이 만족 되어야 함
Freedom From Interference (Partitioning 가능)
Independence
Cascading failure가 없는 상태
Cascading failure, common cause failure가 없는 상태
Cascading Failure
Common Cause Failure
Dependent Failure

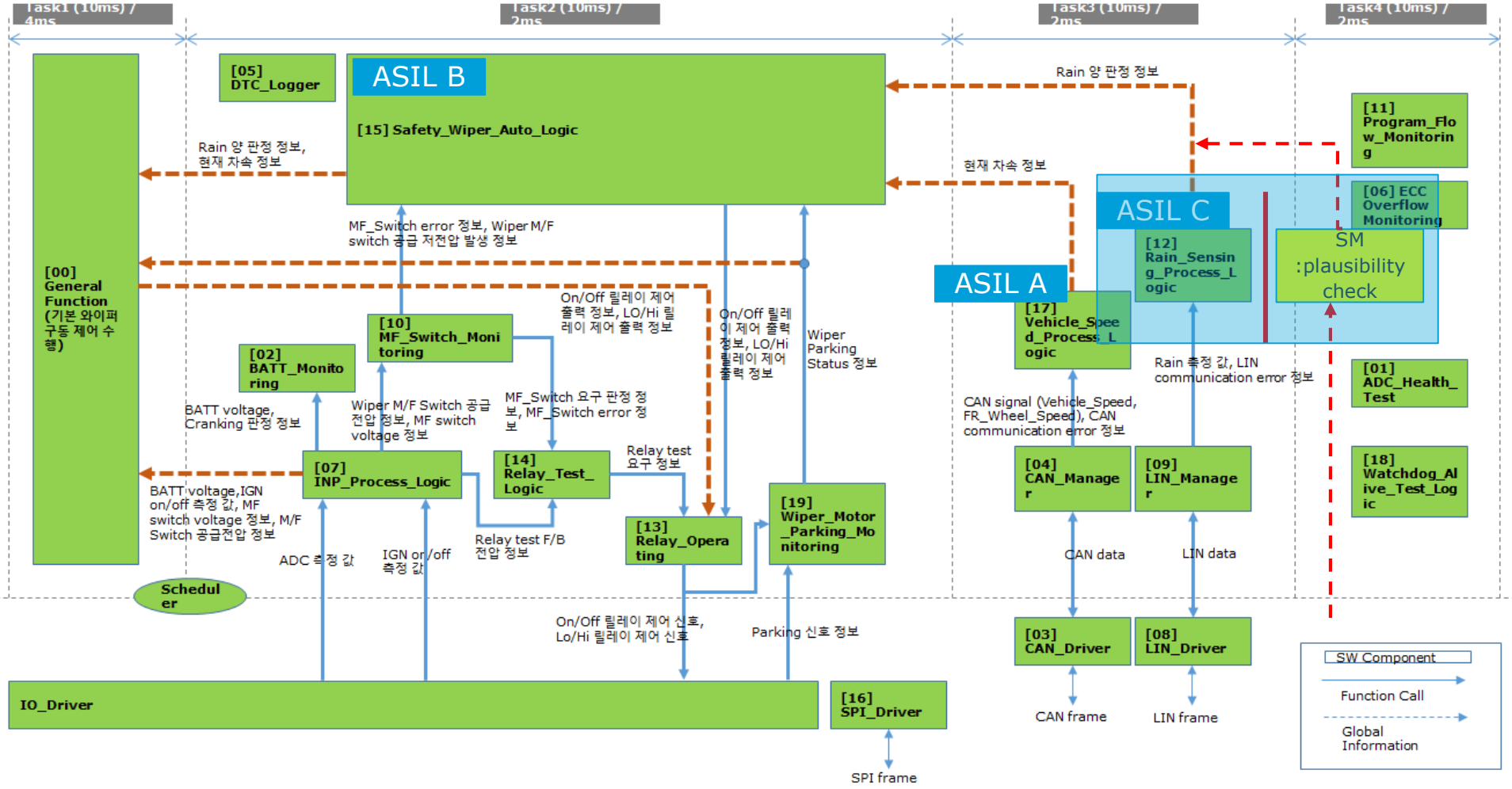
Achievement of freedom from interference – ISO 26262:2018 (Part 6)

	Timing and execution	Memory	Exchange of information
Faults	- Blocking of execution - Deadlocks - Livelocks - Incorrect allocation of execution time - Incorrect synchronization between software elements	- Corruption of content - Inconsistent data (read/write timing) - Stack overflow/underflow - Access violation (read/write)	- Repetition - Information loss, delay, insertion - Masquerade - Incorrect sequence - Information corruption - Asymmetric information sending (sender → multiple receiver) - Information sending to subset of receivers - Communication channel access block
Example Safety measures	- Cyclic execution scheduling - Fixed priority based scheduling - Processor execution time monitoring - Program sequence monitoring	- Memory protection - Error-correcting code (ECC) - Cyclic redundancy check (CRC) - Redundant storage - Restricted memory access - Memory static allocation	- Information repetition - Loop back - Information ACK - Sequence numbers - Alive counter - EDC/ECC (error detection code, error correction code)

DNV GL © 2017

Architecture(first step) - example

Static Architecture – Data Flow (Nominal Control)



Thank you for your attention

고병각 기능안전 실장
Byeong.gak.ko@dnvgl.com
010-5646-0923

www.dnvgl.com

SAFER, SMARTER, GREENER

DNV GL © 2017