SBB CFF FFS

Einführung in die sicherheitskritische SW Entwicklung

Segregation von Anwendungen

Stefanos Moumouris, 10.12.2020

Agenda

1. Einführung in die sicherheitskritische Software Entwicklung nach EN 50128 für Bahnanwendungen

2. Einführung in die Segregation von Anwendungen

Der Vortrag soll einen groben Überblick vermitteln und kann deswegen nicht vollständig und detailliert auf alle Punkte einer sicherheitskritischen Software Entwicklung eingehen.

Vorstellung

- Dipl. Ing. (FH) Elektrotechnik
- 16 Jahre Erfahrung in der Entwicklung und Integration sicherheitskritischer Systeme in der Luftfahrt

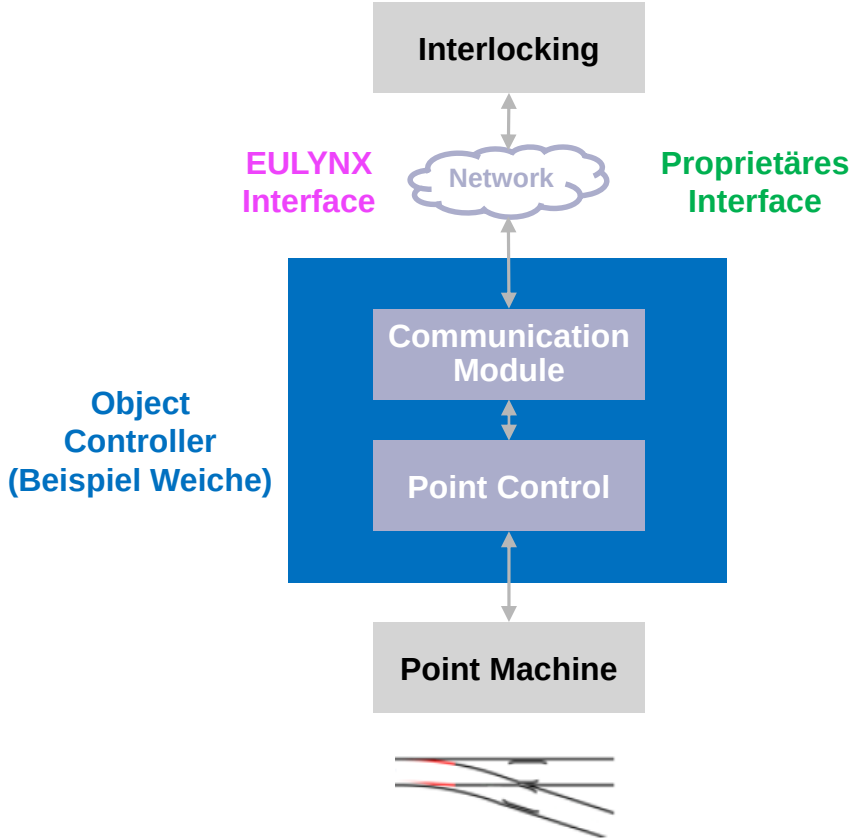
- Seit März 2019 bei SBB in Bern als Systemingenieur im Projekt Object Controller

- Kontakt: stefanos.moumouris@sbb.ch

Einführung Object Controller

(vereinfachte Darstellung)

- Ein Object Controller (OC) steuert und überwacht eine Aussenanlage über ein standardisiertes Interface
- OCs sind schon länger in der Stellwerkstechnikarchitektur vorhanden. Die Sicherungslogik kommuniziert dabei über einen proprietäres Interface mit den entsprechenden Stellteilen (würde in Zukunft einem OC entsprechen)



Einführung Object Controller

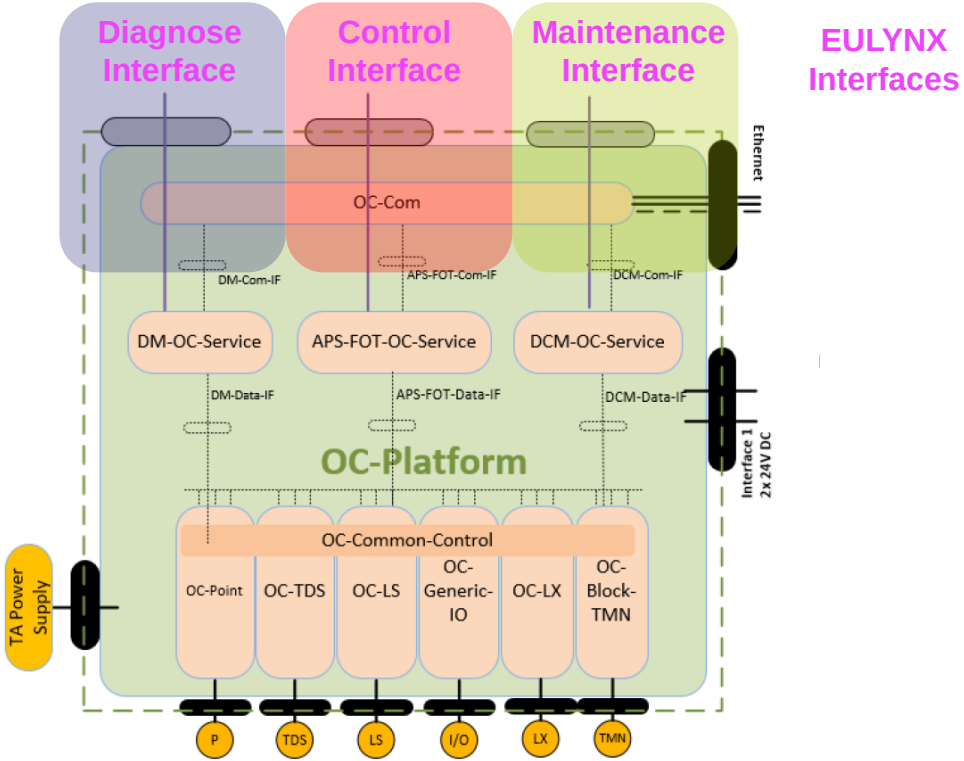
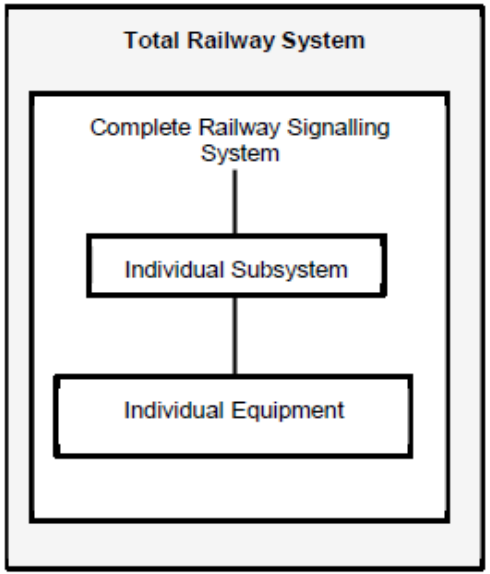


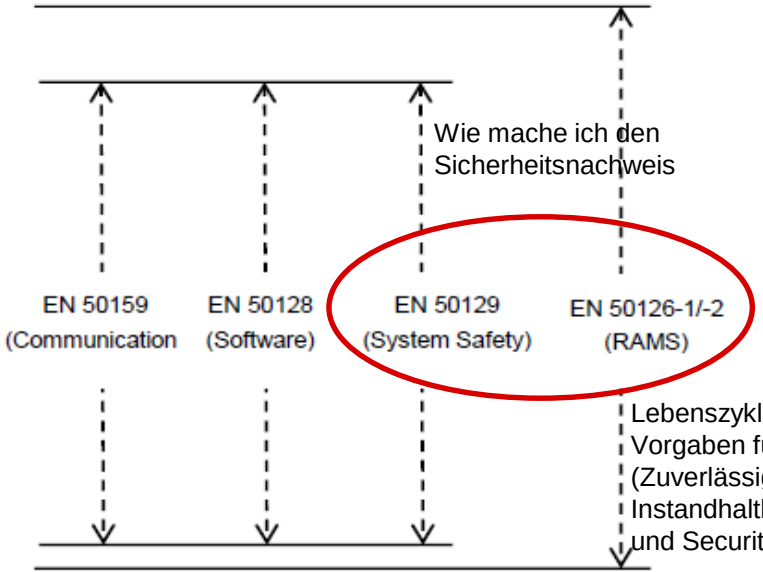
Bild aus Smartrail 4.0 OC Grundkonzept public, Version 0.200

➔ Teil 1: Einführung in die sicherheitskritische SW Entwicklung

Einführung in die sicherheitskritische SW Entwicklung

Welche Normen gibt es?





Basisnorm: EN 61508
Funktionale Sicherheit

Figure 1 – Scope of the main CENELEC railway application standards

Quelle: EN 50129:2016

Einführung in die sicherheitskritische SW Entwicklung

Bevor wir mit der SW Entwicklung anfangen können müssen einige Phasen auf System Ebene durchlaufen werden.
Diese dienen als Input für die Software Entwicklung.

1. Phase 1: System Konzept
2. Phase 2: System Definition und betrieblicher Kontext
3. Phase 3: Risikoanalyse und -beurteilung
4. Phase 4: Festlegen von Systemanforderungen
5. Phase 5: Architektur und Aufteilung von Systemanforderungen

Einführung in die sicherheitskritische SW Entwicklung

- Bei Systemen, welche im sicherheitskritischen Umfeld eingesetzt werden, muss das Risiko eines Schadens (Unfall mit Verletzten oder Toten) minimiert werden.
- Die Gefährdungsanalyse wird auf System Level durchgeführt.
- Die identifizierten Gefährdungen werden bewertet (THR, Tolerable Hazard Rate) und entsprechende Sicherheitsanforderungen werden definiert um diese zu mitigieren.

All diese Schritte werden auf Systemlevel durchgeführt und sind Input für die SW Entwicklung

Einführung in die sicherheitskritische SW Entwicklung

Table 2 — SIL quantitative and qualitative measures

TFFR [h^{-1}]	SIL attribution	SIL qualitative measures
$10^{-9} \leq TFFR < 10^{-8}$	4	Defined in sector-specific standards
$10^{-8} \leq TFFR < 10^{-7}$	3	
$10^{-7} \leq TFFR < 10^{-6}$	2	
$10^{-6} \leq TFFR < 10^{-5}$	1	

THR: Tolerable Hazard Rate (tolerierbare Gefährdungsrate)
TFFR: Tolerable Functional Failure Rate

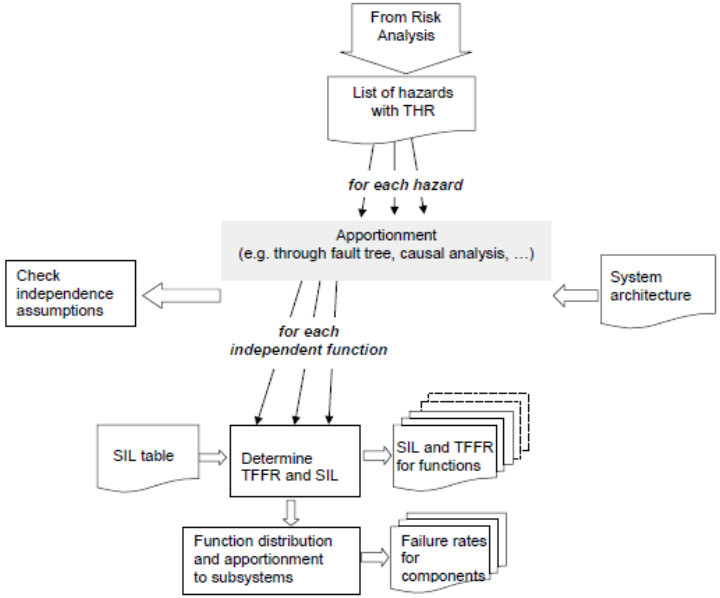
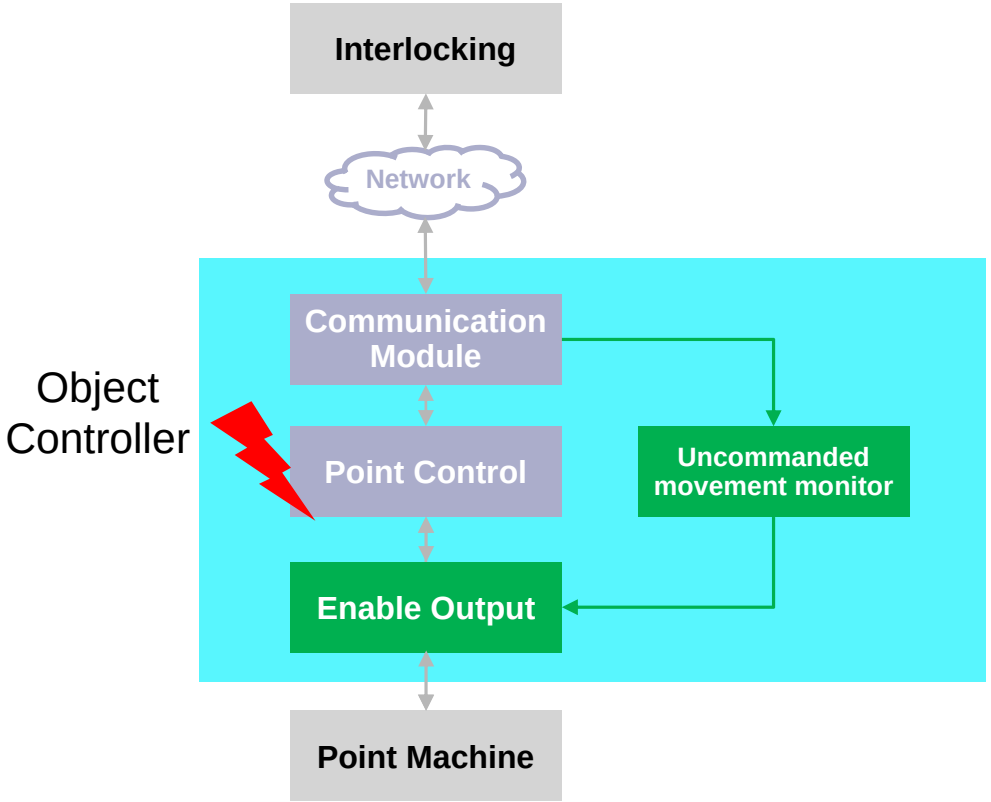


Figure 10 — Apportionment of functional safety requirements

Quelle: EN 50126-2:2017

Einführung in die sicherheitskritische SW Entwicklung

Beispiel: Weiche darf nicht unbeabsichtigt umgestellt werden (identifizierte Gefährdung)



Requirement: Die Weiche soll umgestellt werden wenn ein entsprechendes Kommando vom Stellwerk empfangen wird.

Gefährdung: Die Weiche wird ohne Stellwerks-kommando vom Point Control (bestehend aus HW und SW) umgestellt.

Die Gefährdung wird bewertet und eine tolerierbare Gefährdungsrate (THR) wird festgelegt. Die Gefährdung muss durch geeignete Mitigationsmassnahmen beherrscht werden.

Sicherheitsfunktion: eine unabhängige Funktion überprüft ob ein entsprechendes Stellwerkskommando empfangen wurde und schaltet den Ausgang vom Point Control frei.

Nachweis das die Sicherheitsanforderungen (qualitativ und quantitativ) erfüllt sind

Einführung in die sicherheitskritische SW Entwicklung

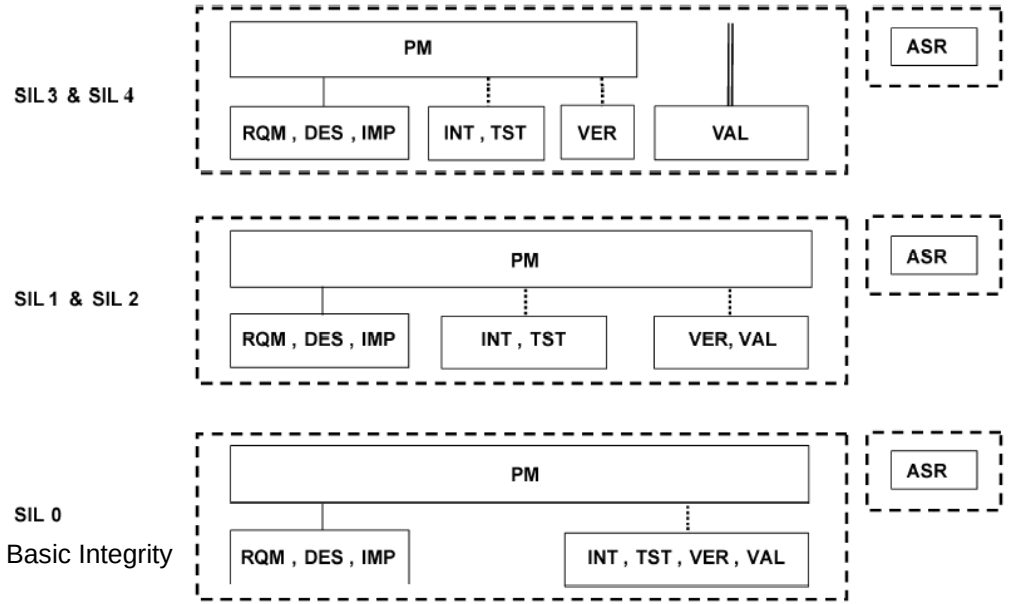
- Wird jetzt in unserem Beispiel ein Teil der SW nach SIL x und ein Teil nach keinem SIL entwickelt?
  - Das muss je nach Situation bewertet werden. Denn unterschiedliche SIL verschiedener Funktionen in einem System müssen segregiert werden (dazu später mehr). Wenn sich dieser Aufwand nicht rechnet, entwickelt man eben die komplette SW nach dem festgelegten höchsten SIL.

Einführung in die sicherheitskritische SW Entwicklung

- Was bedeutet es nach einem bestimmten SIL zu entwickeln?
  - Abhängig vom SIL werden organisatorische Massnahmen als auch Techniken / und weitere Massnahmen gefordert um
    - Systematische Software- und Hardwarefehler in der Entwicklung zu vermeiden (Spezifikations- und Implementierungsfehler)
    - Zufällige Hardwarefehler zu entdecken (Power Up Test / Continuous Monitoring) und sicher darauf zu reagieren (z.B. Fail Safe Zustand)

Einführung in die sicherheitskritische SW Entwicklung

➔ Organisatorische Massnahmen (Auszug)



Quelle: EN 50128:2011

PM:	Projektmanager
RQM:	Requirementsmanager
DES:	Designer
IMP:	Implementierer
INT:	Integrator
TST:	Tester
VER:	Verifizierer
VAL:	Validierer
ASR:	Gutachter

Darf dieselbe Person sein

Darf dieselbe Organisation sein

Muss dem PM unterstellt sein

Darf dem PM unterstellt sein

Darf dem PM **NICHT** unterstellt sein

Einführung in die sicherheitskritische SW Entwicklung

→ Organisatorische Massnahmen (Auszug)

- Mitarbeiter müssen entsprechend geschult sein und bestimmte Qualifikationen besitzen (siehe EN 50128 Anhang B)
- Die Schulungen und die Qualifikationen der Mitarbeiter werden dokumentiert
- Vor Projektbeginn müssen Software Pläne erstellt werden:
  - Qualitätssicherungsplan (wie wird entwickelt, Rollen, Skills, etc)
  - Verifizierungsplan (wie will ich testen, wie will ich überprüfen)
  - Validierungsplan (wann und was möchte ich überprüfen)
  - Konfigmanagementplan

Quelle: EN 50128:2011

Einführung in die sicherheitskritische SW Entwicklung

→ Organisatorische Massnahmen (Auszug)

- Die Entwicklung ist in Phasen unterteilt (siehe V-Diagramm)
- Die Übergangskriterien von einer Phase in die nächste müssen festgelegt werden
- Alle Anforderungen müssen zur höheren Anforderungen verlinkt sein (Traceability) und müssen diese vollständig umsetzen. Das gilt auch für Testspezifikationen und Testprozeduren.
- Derived Requirements (Req. welche nicht zur einem höheren Req verlinkt werden können) müssen bzgl. ihrer Safety Relevanz genau bewertet werden.

  Beispiel: Aufzeichnung zusätzlicher Maintenance Daten in einem OC. Hat das einen Einfluss auf die Sicherheit des Systems?

Quelle: EN 50128:2011

Einführung in die sicherheitskritische SW Entwicklung

→ Organisatorische Massnahmen (Auszug)

- Die Artefakte (z.B. Dokumentation, Source Code, Tests), welche in den einzelnen Phasen entstehen, unterliegen dem Konfig-management und dem Change Management. D.h. es muss genau nachvollzogen werden können was eine Baseline beinhaltet und was sich in einer neuen Version geändert hat.
- Bei Änderung der SW: Impact Analyse durchführen
- Alle Artefakte müssen reviewed und freigegeben werden
- Mit Checklisten arbeiten

Quelle: EN 50128:2011

Einführung in die sicherheitskritische SW Entwicklung

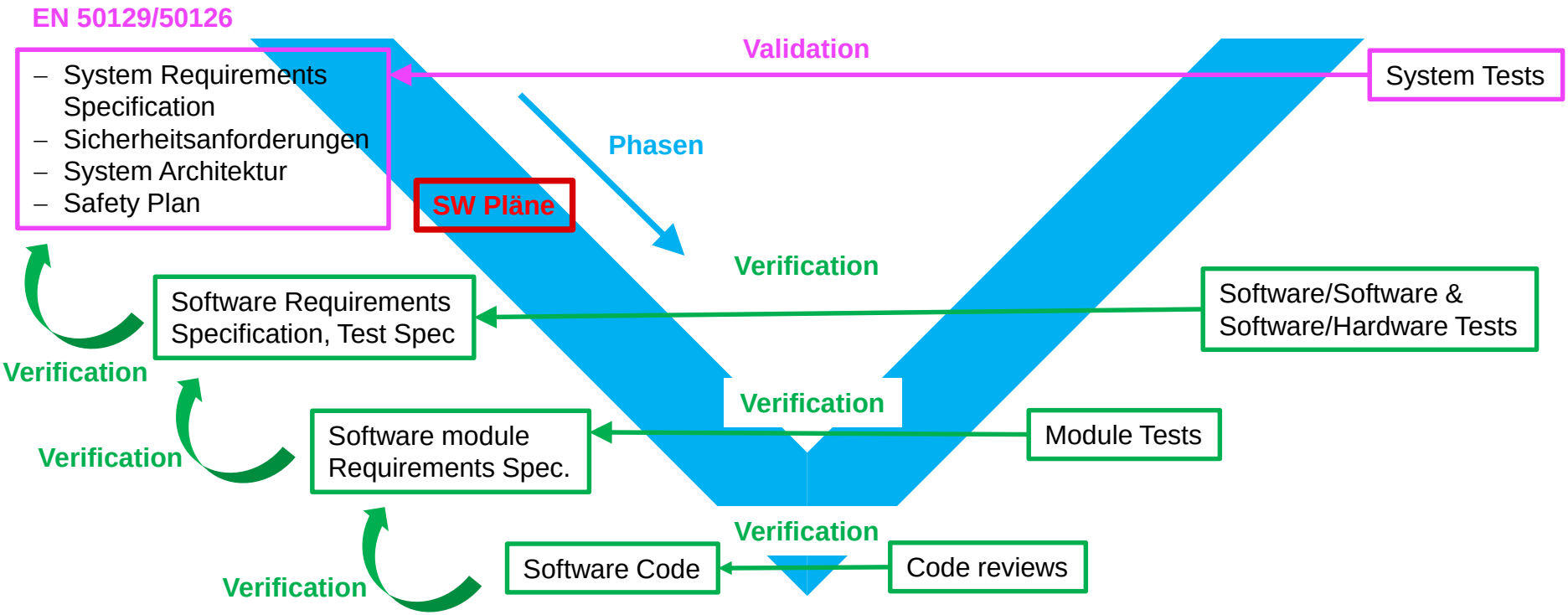
→ Organisatorische Massnahmen (Auszug)

- Verifizierer und Validierer überprüfen ob alle Vorgaben eingehalten wurden und halten dies in einem Bericht fest
- Der Gutachter ist eine unabhängige externe Person welcher überprüft, ob alle Vorgaben eingehalten wurden und hält dies in einem Gutachten fest

Quelle: EN 50128:2011

Einführung in die sicherheitskritische SW Entwicklung

V-Modell



Einführung in die sicherheitskritische SW Entwicklung

- Technik / weitere Massnahmen (Auszug)
  - Requirement Standard (wie sollen Requirements formuliert werden)
  - Design Standard, z.B.
    - Modularer Ansatz (einfache und nachvollziehbare Module)
  - Coding Standard, z.B.
    - Keine dynamische Speicherverwaltung
    - Globale Variablen vermeiden
  - Defensive Programmierung (z.B. Variablen auf ihren Wertebereich überprüfen)
  - Kein Dead Code

Quelle: EN 50128:2011

Einführung in die sicherheitskritische SW Entwicklung

- Technik / weitere Massnahmen (Auszug)
  - Dissimilare Programmierung / HW
    (Beispiel: wie entdecke ich einen Prozessor Fehler)
  - Deterministisches System (fixer scheduler und keine Interrupts verwenden für SIL 4)
  - Statische SW Tests (manuell oder mit Code Checker)
  - Dynamische SW Tests (basierend auf der Module Spezifikation)
  - RAM/Flash Analyse
  - (Worst Case) Runtimeanalysen
  - Stackanalyse

Quelle: EN 50128:2011

Fazit

- SW Entwicklung nach EN 50128 ist kein Hexenwerk aber erfordert bei ersten mal viel Aufwand und Disziplin in der Umsetzung
- Es ist eine geplante, nachweisbare und nachverfolgbare Entwicklung
- Es muss vor Entwicklungsstart genau überlegt werden, wie man vorgehen möchte um die Anforderungen und die entsprechende Normen/Standards zu erfüllen. **Halte Dich an diesen Plan.**
- Den Gutachter sehr früh mit einbinden damit es am Ende keine böse Überraschungen gibt
- **Sicherheitskritische Systeme so einfach wie möglich gestalten**

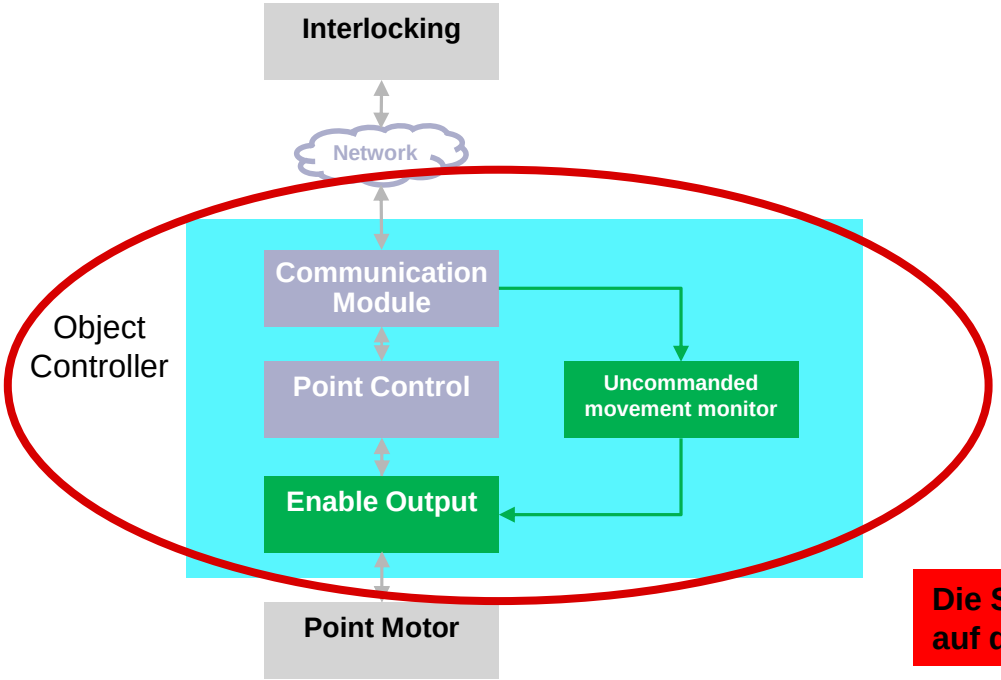
Einführung in die sicherheitskritische SW Entwicklung

→ Haben Sie Fragen zur sicherheitskritischen SW Entwicklung?

➔ Teil 2: Einführung in die Segregation von Anwendungen

Einführung in die Segregation von Anwendungen

➔ Wann ist es erforderlich Anwendungen zu segregieren?



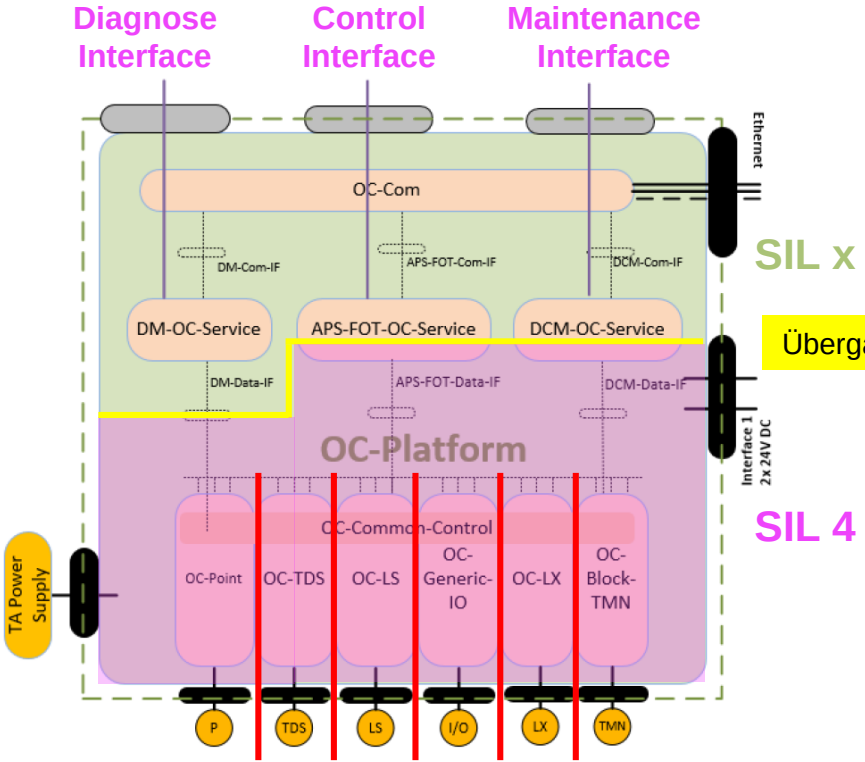
z.B. SIL x

z.B. SIL 4

Die SIL x Anwendung darf keine Auswirkung auf die SIL 4 Anwendung haben.

Einführung in die Segregation von Anwendungen

➔ Wann ist es erforderlich Anwendungen zu segregieren?



Die SIL x Anwendung darf keine Auswirkung auf die SIL 4 Anwendung haben.

Die SIL 4 Anwendungen dürfen untereinander keine Auswirkungen haben.

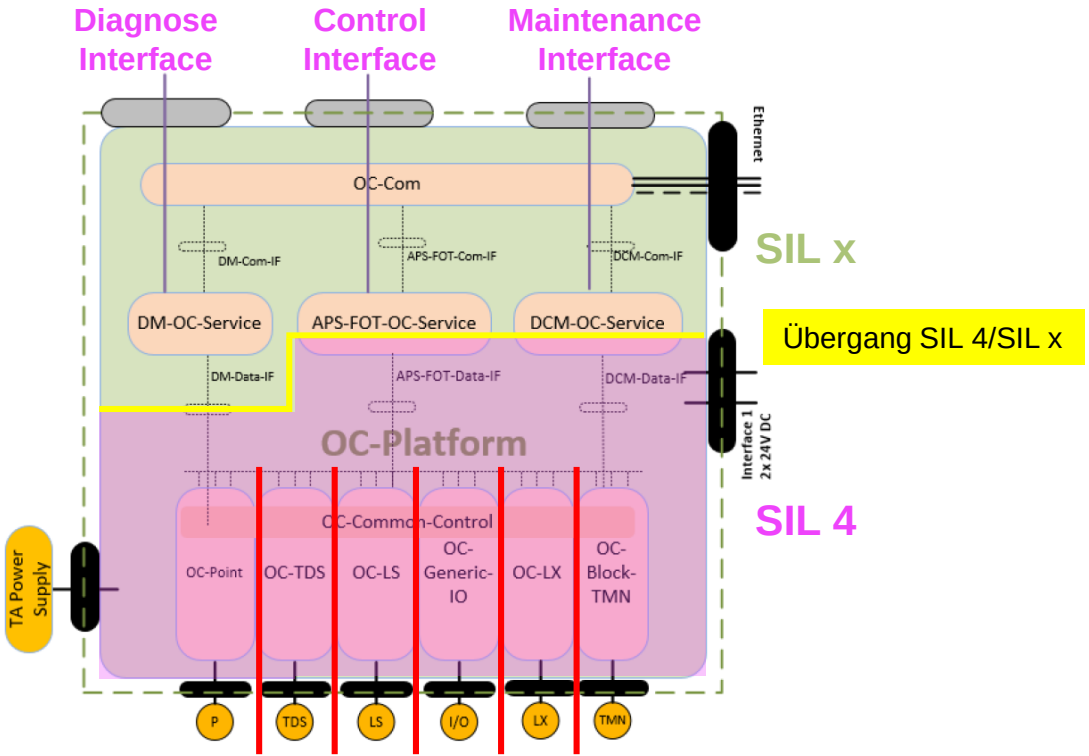
Einführung in die Segregation von Anwendungen

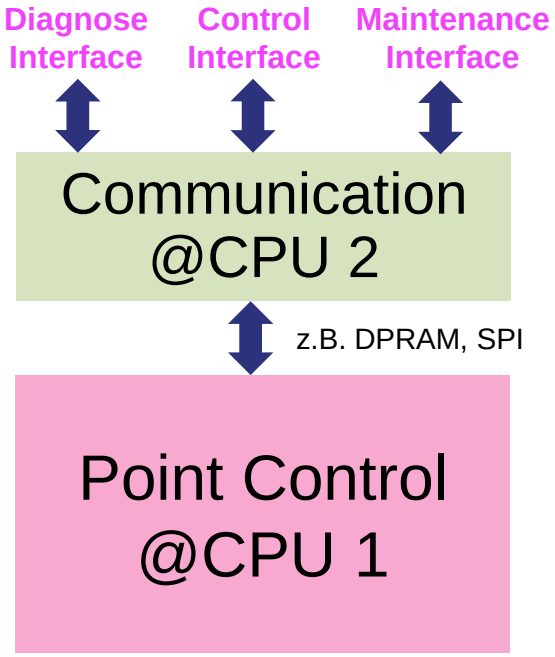
➔ Wie kann ich Anwendungen segregieren?

- Hardware basierte Segregation
- Software basierte Segregation

Einführung in die Segregation von Anwendungen

Hardware basierte Segregation (traditioneller Ansatz)

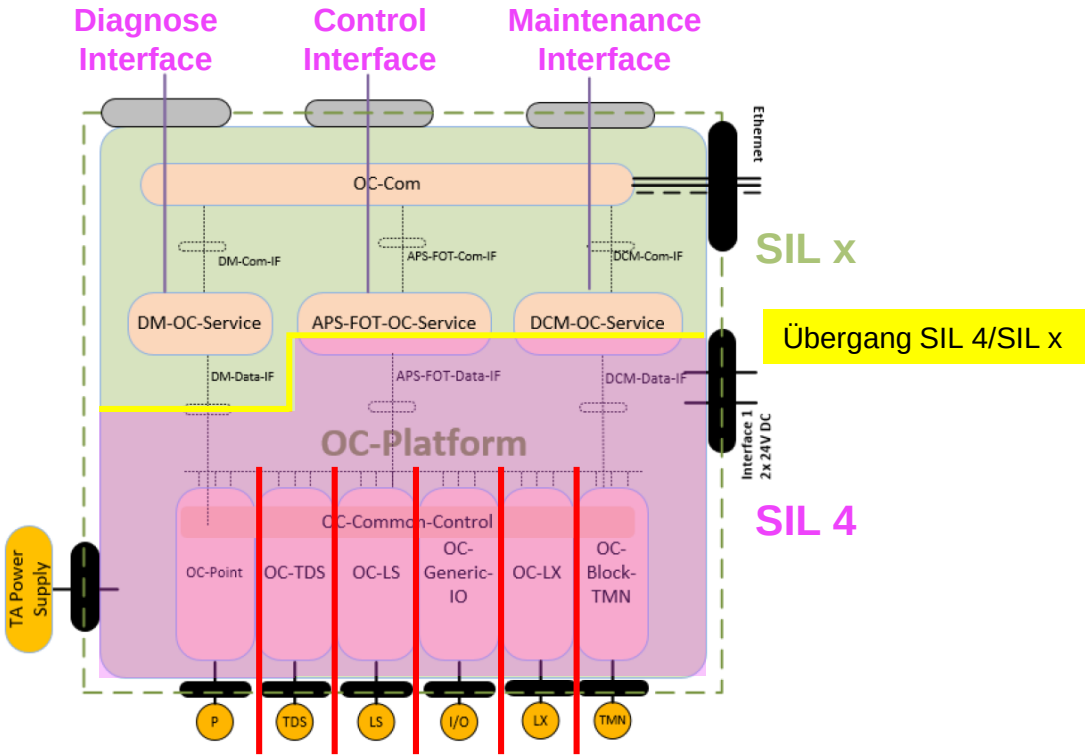




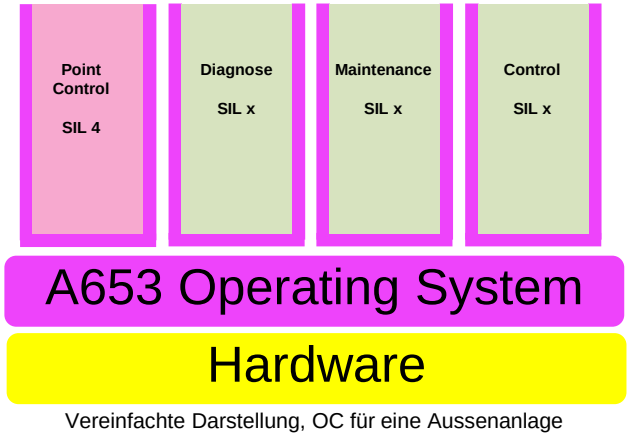
Beispiel: OC für eine Aussenanlage

Einführung in die Segregation von Anwendungen

Software basierte Segregation



- Jede Applikation bekommt eine Rechenzeit und einen Speicherbereich zugewiesen
- Die Applikationen können untereinander Daten über Standard Interfaces austauschen
- Die Applikationen können mit der Peripherie Daten über Standard Interfaces austauschen
- Das System ist deterministisch



Vereinfachte Darstellung, OC für eine Aussenanlage

Einführung in die Segregation von Anwendungen

Software basierte Segregation – Beispiel aus der Luftfahrt

Vergangenheit

Rechner 1
Applikation 1

Rechner 2
Applikation 2

Rechner 3
Applikation 3

Heute

App 1 | App 2 | App 3

Rechner mit A653 OS und Interface Karten

Copyright Diehl Aviation

SBB CFF FFS

Danke für Ihre Aufmerksamkeit.