HYBRIDES SYSTEMS ENGINEERING IM SAFe (SCALED AGILE FRAMEWORK)

ReConf 2022 I München I 27. April 2022

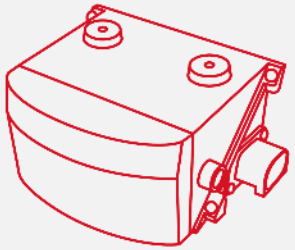
Stefan Rölle – Corporate Systems Engineering Manager

Public

ÜBER IBEO

Weltweiter Technologie-Führer im Feld von Lidar Sensoren, dazugehörige Produkte und Software Tools

- Hamburg, Deutschland
- Eindhoven, Niederlande
- Detroit, USA
- Coming soon: China



Ibeo Scala in Serien-Produktion seit 2017

420+ Mitarbeiter

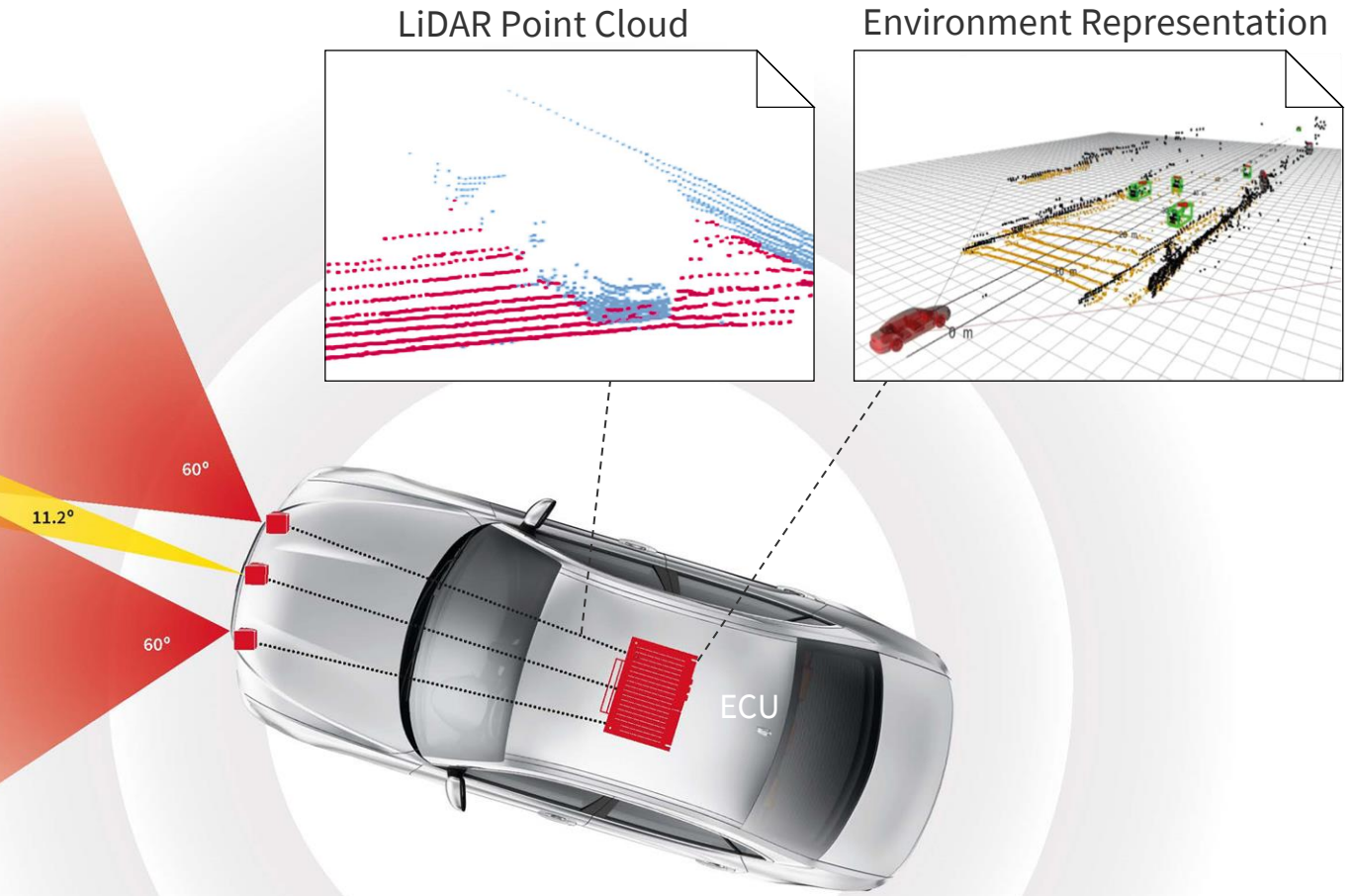
OUR MISSION

We want a future without any road accidents, with the car as the cooperative partner for those who can't protect themselves – especially our children.

PUBLIC

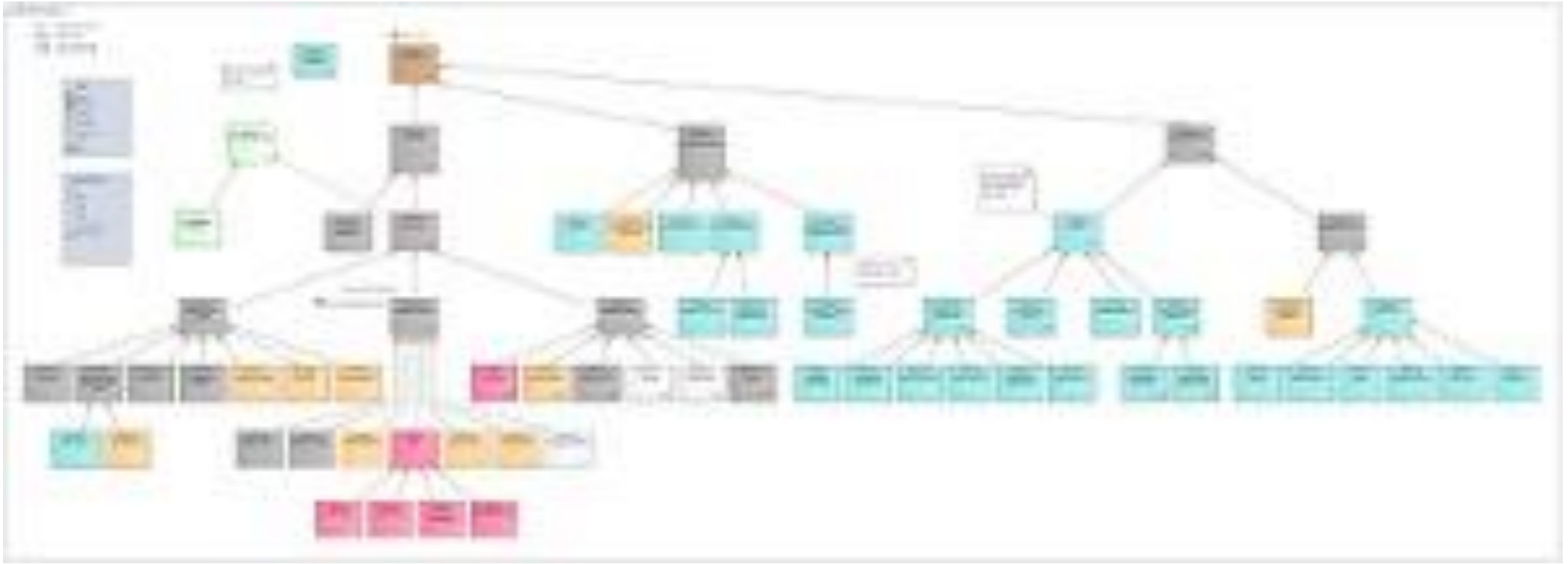
ibeoNEXT SOLID STATE LIDAR – SYSTEM ARCHITECTURE

- Picture shows system architecture for L3 highway system which is currently in industrialization
- Sensors provide raw point cloud to central ECU
- The central ECU with integrated Perception Software fuses point clouds to environment representation



PUBLIC

ibeoNEXT - ARCHITECTURAL DOCUMENTATION LEVELS



HERAUSFORDERUNGEN/CHANCEN

- Kunden fordern agile Produktentwicklung
- Neuartiges Produkt (kein Vorgänger-Produkt mit gleicher/ähnlicher Technologie)
- Cross-Domain Produkt (System „besteht“ aus Hardware, Software, Mechanik, Optik)
- Balance zwischen Core-Entwicklung und kundenspezifischer Entwicklung
- Bedienen von verschiedenen Märkten mit unterschiedlichen Anforderungen
- Autonomous Driving Umfeld (ISO 26262 – Functional Safety, ISO/SAE 21434 – Cybersecurity, A-Spice, etc.)
- Stark wachsendes Unternehmen
- Selbstorganisation

ANALYSE DER DOMÄNEN HINSICHTLICH AGILER ENTWICKLUNG/DOKUMENTATION

Software

Feature-based Dokumentation

Vertikale Sichtweise („Durchstich“) möglich

Requirements und Architekturen (sowie die dazugehörige rechte Seite des V) können in kürzeren Zyklen iteriert werden

„Kurze Wege“ durch Inhouse-Entwicklung

Kurze Implementierungs-Zeit

System / Hardware / Mechanik / Optik

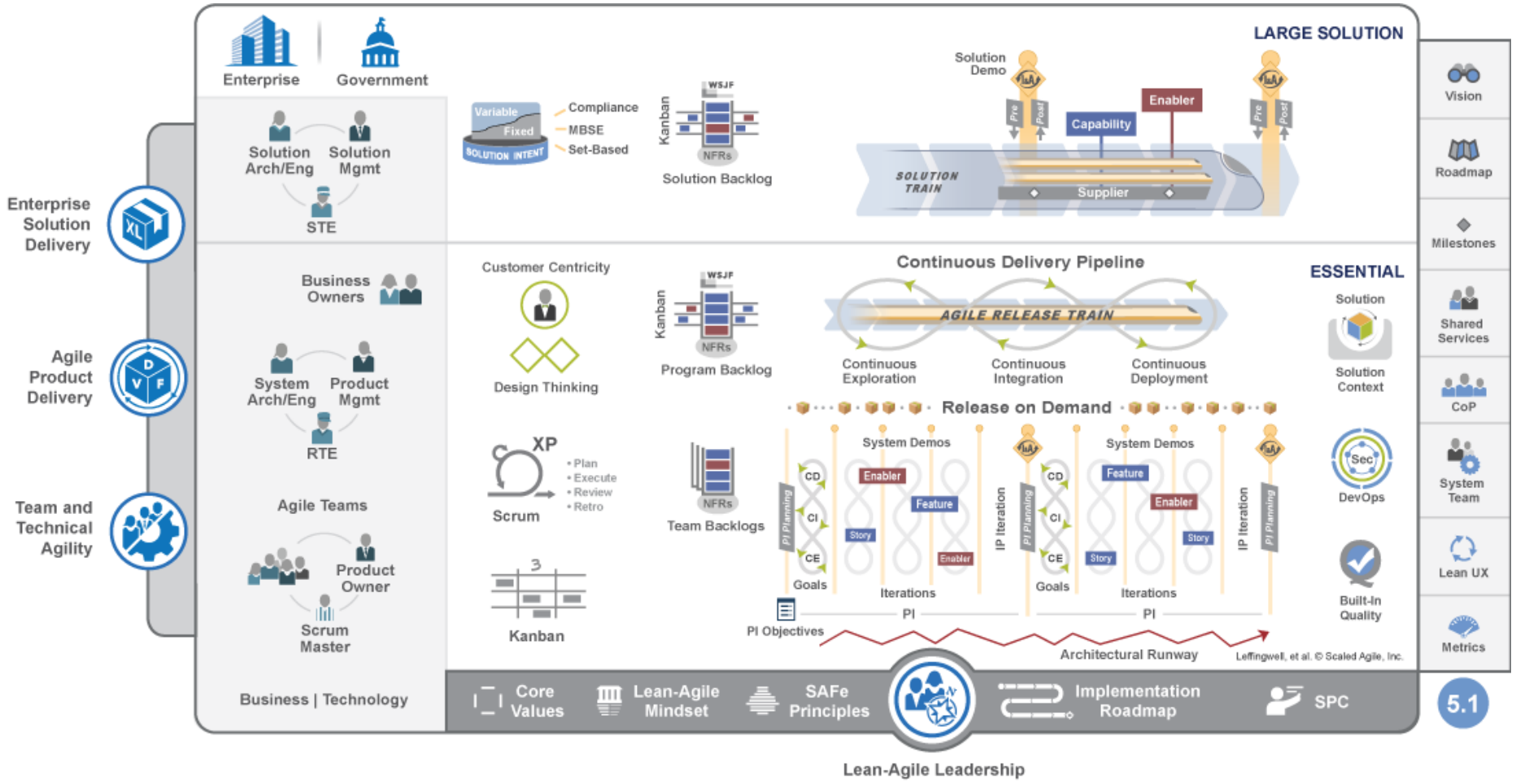
Component-based Dokumentation (mit Feature-Fokus)

Fokus auf nicht-funktionale Randbedingungen auch zu Beginn der Entwicklung extrem wichtig

Höhere Formalität notwendig durch externe Entwicklung (Lastenheft – Pflichtenheft auch für Sample-Phasen)

Definition und Dokumentation der Integrations-Stufen sowie der Verantwortlichkeiten dafür

ÜBERBLICK SCALED AGILE FRAMEWORK



Quelle: https://www.scaledagileframework.com/

DEFINITION UND STRUKTURIERUNG DER ENGINEERING TÄTIGKEITEN

Entscheidung:
Generische Definition von Epic-Kategorien inklusive Definition of Done

Die Schneidung der Epic-Kategorien basierte auf folgenden Kriterien:

- Abarbeitbar in einem PI (Product Increment)
- Weiter zerlegbar in Tasks
- Zuweisung der Epics zu Teams soll (in der Regel) 1:1 möglich sein

Wichtig(st)e Vorbedingung für eine funktionierende agile Produktentwicklung im Rahmen von SAFE:

SAFe follows Architecture

Strukturierung der ARTs, Teams, etc. sollte sich weitestgehend
an der Produktarchitektur orientieren

ENTWICKLUNG UNTERSCHIEDLICHER DEFINITION OF DONES FÜR EPICS

Software

Kategorien pro SW-Feature (d.h. relevant für mehrere Produkt-Architektur-Ebenen) je Release

FEA 1: Requirements und Architekturen (für alle relevanten Ebenen)

FEA 2: Test-Case Dokumentation (für alle relevanten Ebenen)

MVP: Implementierung und Test (für alle relevanten Ebenen)

System / Hardware / Mechanik / Optik

Kategorien pro Betrachtungsgegenstand (System oder HW/MEC/OPT-Komponente) je Release/Sample

REQ 1: Funktionale Anforderungen

REQ 2: Nicht-Funktionale Anforderungen

ARC 1: Funktionale und logische Architektur

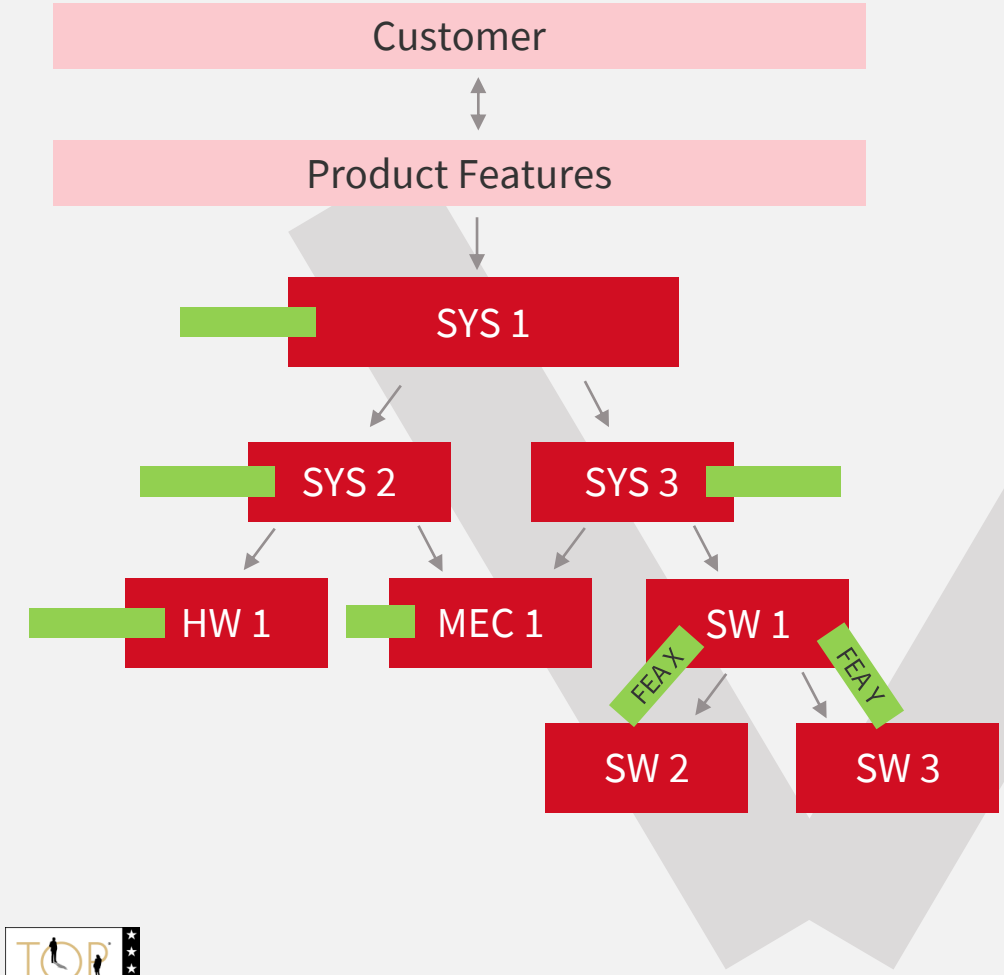
ARC 2: Physikalische Architektur

TES 1: Test-Case Dokumentation

TES 2: Test-Durchführung inkl. Reporting/Metriken/etc.

Keine weiteren Kategorien, weil keine interne Implementierung/Produktion

BEISPIEL NOTWENDIGER EPICS FÜR DIE KERN-ENGINEERING-TÄTIGKEITEN



Beispiel:

Erforderliche Epics (in Bezug auf das Core-V) zur Umsetzung des Gesamtsystems für einen Release-Scope, der SW-Feature 1 und SW-Feature 2 beinhaltet:

SYS 1: REQ1, REQ2, ARC1, ARC2, TES1, TES2

SYS 2: REQ1, REQ2, ARC1, ARC2, TES1, TES2

SYS 3: REQ1, REQ2, ARC1, ARC2, TES1, TES2

HW 1: REQ1, REQ2, ARC1, ARC2, TES1, TES2

MEC 1: REQ1, REQ2, ARC1, ARC2, TES1, TES2

Feature X: FEA1, FEA2, MVP

Feature Y: FEA1, FEA2, MVP

FALLSTRICKE UND WICHTIGE ZU TREFFENDE ENTSCHEIDUNGEN (1/3)

- **Wie geht man in SAFe mit den System-Ebenen und vor allem dem Gesamtsystem um? Verantwortet ein ART dies? Gibt es eine Plattform-Ebene, die das Gesamtsystem inkl. Dokumentation verantwortet?**
  Idee: Es muss zumindest klar definiert sein, wer das Gesamtsystem verantwortet inklusive all der Aufgaben, die daran hängen. Je nach Produkt kann es auch Sinn machen, dass ein SW-naher ART das Gesamtsystem verantwortet. Wichtig ist auf jeden Fall, dass System- und Domänen-Verantwortliche sehr eng zusammenarbeiten.

- **Wie stellt man im Projekt-Management den „Issue Workflow“ über die SAFE-Ebenen sicher? (z.B. aus Customer Storys werden User Storys abgeleitet, daraus werden Epics abgeleitet, daraus werden wieder Epics für Aufgaben zu tieferen Architektur-Ebenen abgeleitet etc.)**
  Idee: Idealerweise beginnt man, einen Prozess zu definieren und diesen innerhalb der PM-Tool-Strukturen (z.B. Jira) umzusetzen. Dazu sollte man zwingend Workflows definieren und implementieren, ansonsten wird man der Komplexität nicht Herr – dabei hilft ein Entwicklungs-naher Dev-Support sehr.

- **Ist es möglich, eine Auslieferung für Ende eines PIs zum Beginn dieses PIs zu planen?**
  Idee: Je nach Release-Scope. Grundsätzlich ist es, gerade bei komplexen, verteilten Systemen, realistisch, dass eine detaillierte Planung (nicht unbedingt Umsetzung) mindestens schon zu Beginn oder spätestens zu Mitte des Vor-PIs beginnt.

PUBLIC

FALLSTRICKE UND WICHTIGE ZU TREFFENDE ENTSCHEIDUNGEN (2/3)

- **Wie geht man mit unterschiedlichen Entwicklungs-Zyklen verschiedener Sub-Systeme um? Ist man wirklich zu Ende jedes PIs auslieferungsfähig? (Stichwort: technisches Konfigurations/Kompatibilitäts-Management)**
  Idee: Agilität hin oder her, es ist in der Systementwicklung (vor allem im Automotive mit den extremen Supplier-Strukturen) extrem wichtig, Meilensteine vom Kunden-SOP kommend rückwarts aufzuplanen und abzustimmen. Die Agilität, die SAFe einem ermöglicht, ist, so spät wie nötig eine Detaillierung der Meilensteine vorzunehmen.
  Auch eine kleine SW-Änderung, die dem Kunden bereitgestellt wird, lässt ein neues System entstehen. Es müssen nicht zu jedem PI möglichst viele Teil des Produkts angerührt und ausgeliefert werden.

- **Wie realisiert man „Built-In Quality“ im sicherheitskritischen/Nachweis-intensiven Umfeld?**
  Idee: Definition und Implementierung von Sondern-Rollen, die SAFe als solches nicht vorgesehen hat (z.B. QA, Requirements Manager, Test Manager, Configuration Manager, Release Manager etc.) sind (zumindest bei Ibeo) sehr wichtig. Diese Rollen sind verantwortlich, die Prozessqualität für das jeweilige Fachgebiet sicherzustellen sowie die Entwicklungsteams zu coachen und zu begleiten.

FALLSTRICKE UND WICHTIGE ZU TREFFENDE ENTSCHEIDUNGEN (3/3)

- **Wie viel Dokumentation benötige ich für Development-Releases?**
  Idee: Je nach Kunden-Use-Case, Release Scope und Maturität sind relevante Normen, Gesetze und Standards anzuwenden.
  Beispiel: Wird abgesicherte Funktionalität (in Bezug auf ISO26262) gefordert und mit Outputs des Produkts Lenk-Aktuatorik angesteuert plus auf öffentlichen Straßen damit autonome Fahrfunktionen verprobt, ist es ziemlich wahrscheinlich, dass man einen vollständigen Safety Case für den Release Scope in Gänze erfüllen muss.
  Generell empfiehlt es sich, sinnige Release Levels zu definieren (inkl. „Welche Doku benötige ich dafür für welches Release Level?“) und diese mit Kunden-Release-Levels zu konsolidieren bzw. zu mappen.

- **Wie bringe ich ISO 26262, ISO/SAE 21434, A-Spice, IATF 15949, etc. in SAFe unter?**
  Idee: Es ist elementar wichtig, einen voll integrierten Entwicklungsprozess zu haben und zu leben. Das bedeutet, dass alle relevanten Normen, Standards und Gesetze als Input für den Prozess dienen und von Prozess- und Methoden-Experten analysiert und konsolidiert werden. Etwaige Tailorings können im Prozess beschrieben werden und in Projekten angewandt werden (z.B. semi-formale Notation von Anforderungen vs. formale Notation von Anforderungen, je nach ASIL).
  -> Nicht jeder Entwickler/Architekt/Tester/etc. muss im Detail alle für ihn relevanten Normen kennen. Die verantwortlichkeit liegt beim Prozess-Owner

VIELEN DANK FÜR DIE AUFMERKSAMKEIT

Stefan Rölle I Corporate Systems Engineering Manager
stefan.roelle@ibeo-as.com

www.ibeo-as.com Ibeo Automotive Systems GmbH | Merkurring 60-62 | 22143 Hamburg, Germany
phone: +49 - 40 298 676 – 0 | fax: +49 - 40 298 676 - 10 | E-Mail: info@ibeo-as.com