Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

# Herausforderungen in der Releaseplanung im Umfeld Automotive

Aretina Iazzolino

| | |
|---|---|
| **Studiengang:** | Softwaretechnik |
| **Prüfer/in:** | Prof. Dr. Stefan Wagner |
| **Betreuer/in:** | M.Eng. Kristina Marner |
| **Beginn am:** | 21. Februar 2019 |
| **Beendet am:** | 21. August 2019 |

## Kurzfassung

Kaum ein Industriezweig steht unter starken Konkurrenzdruck wie die Automobilbranche. Die Fahrzeuge müssen bei möglichst gleichbleibender Qualität und kürzeren Produktzyklen, immer schneller zur Marktreife gebracht werden. Das ist auf den ständigen Zyklus der Neuerung und Verbesserung zurückzuführen. Mehr als 90% der Kundenanforderungen werden prädominant durch Software umgesetzt. So hat sich ein cross-funktionales Umfeld, in der von der Mechanik geprägten Automobilwelt aufgebaut, bestehend aus den weiteren Bereichen Elektrik und Elektronik. Die Vernetzung dieser Bereiche führt zu einer erhöhten technischen Komplexität. Dies muss bei der Releaseplanung von sicherheitskritischen Systemen, wie das Automobil einzustufen ist, berücksichtigt werden. Alle involvierten Disziplinen und ihre technischen Abhängigkeiten müssen in ein funktionierendes Ganzes vereint und validiert werden. So wird bespielsweise die Steuergerätesoftware in Verbindung mit dem zugehörigen Steuergerät getestet und Umgekehrt. Die Schwierigkeit ergibt sich, wenn die Teilsysteme ihre eigenen Releasezyklen und Fertigstellungstermine besitzen. Sie können zum einen starkt an die Meilensteinstruktur angelehnt sein, und somit klassisch abgewickelt werden. Zum anderen führt der Einzug agiler Methoden im automotive Umfeld, vorzüglich in Softwareumgebungen, zu schneller lieferfähige Produktinkremente. Die unterschiedlichen Releasezyklen der Teilsysteme und ihre Abhängigkeiten, erschweren einen reibungslosen Ablauf der Qualifikationsphase im Verbund. Ziel dieser Arbeit ist es diese Herausforderungen in der Releaseplanung im automobilen Umfeld zu identifizieren. Dazu soll eine Studie mit Experten aus dem automotive oder automotive ähnlichen Umfeld durchgeführt werden, um den Stand des Markt wiedergeben zu können.

# Inhaltsverzeichnis


**1. Einleitung** **15**
- 1.1. Motivation . . . 15
- 1.2. Herausforderungen für das Projektmanagement . . . 17
- 1.3. Ziel der Arbeit . . . 18
- 1.4. Aufbau der Arbeit . . . 18

**2. Stand der Technik** **19**
- 2.1. Der klassische Produktentstehungsprozess in der Automobilindustrie . . . 19
- 2.2. Die Produktentwicklungsphase . . . 20
- 2.3. Sicherheitskritische Elektroniksysteme . . . 28
- 2.4. Agile Produktentwicklung . . . 33
- 2.5. Agile Projektorganisation und -Strukturen . . . 33

**3. Verwandte Arbeiten** **39**
- 3.1. Verwandte Studien zur Releaseprozessysthematik . . . 39

**4. Methodisches Vorgehen** **41**
- 4.1. Grounded Theorie Methodologie (GTM) . . . 41
- 4.2. Coding . . . 42
- 4.3. Studiendesign und Methodik . . . 42
- 4.4. Samplingstrategie . . . 43

**5. Auswertung und Vergleich zu den Porscheergebnissen** **45**
- 5.1. Context . . . 45
- 5.2. Prozesssystematik: Verbundrelease . . . 46
- 5.3. Prozesssysthematik: Planung . . . 49
- 5.4. Prozesssystematik: Integration . . . 52
- 5.5. Entwicklungsmethodik: Abstimmung . . . 54
- 5.6. Entwicklungsmethodik: Testen . . . 56

**6. Interpretation** **59**
- 6.1. Studienvergleich . . . 59
- 6.2. Auswertung der Daten mit Hilfe der GTM . . . 67
- 6.3. Anforderungsumsetzung und Featureaufbau . . . 67
- 6.4. Verbundreleasephase . . . 68
- 6.5. Testphase . . . 68
- 6.6. Fehlerabbau- und Featureaufbauphase . . . 69

**7. Fazit und Ausblick** **71**
- 7.1. Fazit . . . 71

7.2. Ausblick . . . . . 72

**Literaturverzeichnis** **73**

**A. Fragebogen** **79**

# Abbildungsverzeichnis

1.1. Die Entwicklung der verbauten Funktionen und elektronischen Steuergeräte im Automobil. [SZ16] . . . . . 15
1.2. Der klassische Produktentstehungsprozess. [BBW+16] . . . . . 16

2.1. Schematische Darstellung eines Produktentstehungsprozesses. [BBW+16] . . . . 19
2.2. Lebenszyklus eines Automobils eigene Darstellung. . . . . 20
2.3. Das Wasserfallmodell, angeleght an [Wol18]. . . . . 21
2.4. Eine abstrakte Darstellung des V-Modells [Sch05]. . . . . 22
2.5. Traditioneller Entwicklungsverlauf . . . . . 23
2.6. Eine eigene Darstellung der Meilensteinstruktur. . . . . 24
2.7. Meilenstein . . . . . 25
2.8. V-Modell für die Entwicklung mechtronischer Systeme nach VDI 2206 [FG13]. . 29
2.9. Die Softwareentwicklung nach V-Modell. [Wol18] . . . . . 30
2.10. Scaled Agile Framework (SAF)-Framework [Agi] . . . . . 37

6.1. Der ideale Prozessverlauf. . . . . 67

# Tabellenverzeichnis

5.1. Rollen der befragten Experten . . . . . 45

# Akronyme

**ASIL** Automotive Safety Integrity Level. 28

**ASPICE** Automotiv Software Process Improvement and Capability Determination. 26

**AUTOSAR** Automotive Open System Architecture. 31

**CMM** Capability Maturity Model. 25

**CMMi** Capability Maturity Model Integration. 26

**GTM** Grounded Theorie Methodologie. 5

**HIL** Hardware in the Loop. 68

**MIL** Model in the Loop. 68

**OEM** Original Equipment Manufacturer. 17

**PEP** Produktentstehungsprozess. 16

**PIL** Prozessor in the Loop. 68

**RE** Requirement Engineering. 23

**SAF** Scaled Agile Framework. 7

**SE** Simultaneous Engineering. 21

**SIL** Software in the Loop. 68

**SOP** Start of Production. 20

**SPICE** Software Process Improvement and Capability Determination. 26

# 1. Einleitung

## 1.1. Motivation

Das Automobil welches wir heute fahren, hat bereits drei industrielle Revolutionen überstanden. Nicht nur die Optik hat sich stark verändert, vor allem aber die Komplexität der verbauten Komponenten. Geschichtlich betrachtet hat die zugehörige Branche mehrere evolutionäre Lösungen im Bereich der unabhängigen Mobilität entwickelt. Dazu zählt die Erfindung passiver und aktiver Sicherheitsmaßnahmen, wie AIR BAGs oder ABS Systeme [KLP03]. Mehr als die Hälfte dieser Entwicklungen hätten aber ohne den Einfluss von Software nur schwer Einzug im Automobil gefunden. Die wichtigsten Meilensteine wurden ab dem Jahr 1980 gesetzt (Abbilgung 1.1). Seitdem wurde die IT, gepaart mit mechatronischen Lösungen, immer mehr zum Game-Changer und etabliert heute noch die größten Veränderungen im Automobil [SZ16].

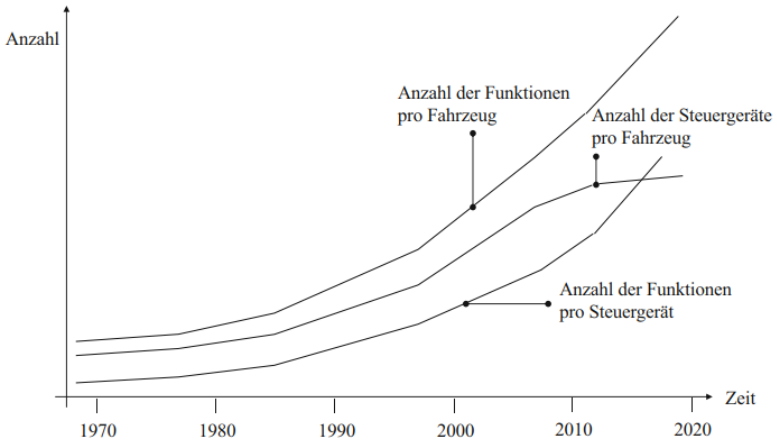


**Abbildung 1.1.:** Die Entwicklung der verbauten Funktionen und elektronischen Steuergeräte im Automobil. [SZ16]

Neben einem exponentiellen Anstieg des Funktionsumfang steigt auch die Anzahl der verbauten Steuergeräte die vermehrt durch den Einsatz von Softwarefunktionen überholt wird. Dies ist dadurch bedingt, dass etwa 90% der Innovationen am Fahrzeug durch mechatronische Systeme und 80% davon durch Software umgesetz werden [BS12]. Infotainment Systeme werden größtenteils von Software dominiert. Viele mechanische Funktionen im Fahrzeug, werden durch den Einsatz mechatronischer Lösungen abgelöst. In der Motorsteuerung bspw. will man durch den Einsatz von elektronischen Steuergeräten den Kraftstoffverbrauch und die Emissionen effizient reduzieren. So können sozialpolitische Ziele verfolgt werden und gleichzeitig umweltfreundlichen Anfoderungen gerecht werden.

Sicherheits- und Umweltaspekte führen maßgeblich zu kontinuierlichen Verbesserungen und gleichzeitig zu einer steigenden Produktkomplexität und Produktvielfalt. Bis dato hat die traditionelle Automobilbranche überlebt, da die Marktbedürfnisse vor allem mit evolutionären Lösungen gestillt werden konnten.

Nicht zuletzt sichert der Wunsch unabhängiger Mobilität diesem Sektor auch in Zukunft einen großen Marktanteil. Jedoch herrscht in der heutigen Automobilbranche eine angespannte Atmosphäre, da revolutionäre Entscheidungen bevorstehen. Innovationen wie Connected Cars, autonomes Fahren und Carsharing sind einige der gesellschaftspolitischen Diskussionsfelder, die den Weg in eine vierte industrielle Revolution ebnen. Neben der Modalität wie wir unsere Autos in Zukunft fahren, ändern sich auch vermehrt die Technologien die wir einsetzen. Eine schnelle Anpassung und Anforderungsumsetzung zählt zu den Erfolgsfaktoren und ist gleichzeitig eine große Herausforderung für die heutigen Automobilbauer. Denn die Entstehungsprozesse die heute adaptiert werden sind nur bedingt in der Lage, die Wünsche des Marktes zeitgemäß umzusetzen. Häufig werden klassische Methoden eingesetzt, die wasserfallartig arbeiten und eine semi-sequentielle Abfolge mit hohem Kommunikationsaufwand während dem Projekt erfordern.

Die größten Optimierungspotentiale herrschen im klassischen Produktentstehungsprozess (Produktentstehungsprozess (PEP)). Der klassische (PEP) bildet die Hauptphasen der Entstehung eines Automobils (Abbildung 1.2).

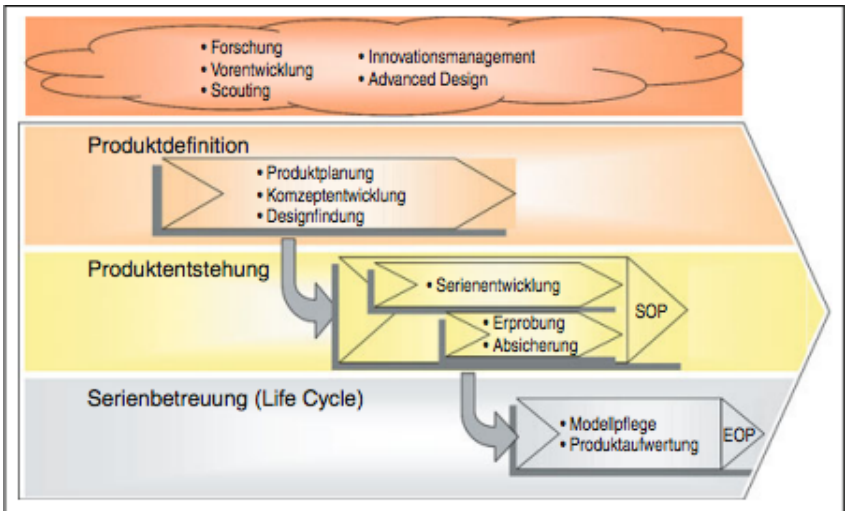


**Abbildung 1.2.:** Der klassische Produktentstehungsprozess. [BBW+16]

Man erkennt eine semi Sequentialität und die kaskadenartige Anordnung der Prozesse. Solche Vorgehesmodelle werden im Automobilsektor besonders vom Managament und den höheren Organisationsstufen bevorzugt, da sie eine strikte Vorausplanung des Produktes von bis zu zwei Jahren im Voraus erfordern. So stehen die relevanten Anforderungen zu Beginn des Projektes fest, die das Management als Basis für eine umfangreiche Projektplanung (Kosten, Termine etc.) verwendet. Eine Detailplanung rund um das Verfügbare Budget und den Ressourcen ist ein großer Erfolgsfaktor. Gleichzeitig kann auf außerplanmäßige Änderungen nur schwer reagiert werden, da sie unvorhersehbar sind und für die nachträgliche Umsetzung sehr hohe Kosten entstehen. Häufig

werden Probleme und Unstimmigkeiten erst spät im PEP erkannt. Die Bereitschaft und Möglichkeit, auch im fortgeschrittenen Entwicklungsprozess auf geänderte Kundenwünsche und Anforderungen einzugehen ist von essentieller Bedeutung [Sch14], [Wol18].

Die Produktionsphase, die als Vorbereitung der Serienentwicklung dient, nimmt eine immer größere strategische Rolle in der Produktentstehung. Hierfür definiert jeder Original Equipment Manufacturer (OEM) eine individuelle Releaseplanungssystematik, die mit Hilfe von Meilensteinen eine termingerechte Entwicklung der Steuergeräte/Komponenten vorantreibt. Das ist ein zeitintensiver Prozess der zwischen 2 bis 5 Jahren dauern kann und mit dem aktuellen Stand der Technik äußerts präzise Vorausgeplant werden muss. Die obersten Ziele der heutigen Automobilbauer sind eine erhebliche Verkürzung des Entwicklungszeitraumes bei gleichzeitiger Verbesserung der Qualität um nicht im Wettbewerbsstrom zu versinken. Hier fehlt heutzutage aber ein konsequenter Einsatz an Methoden die diese Vision unterstützen. Deshalbt sucht man nach Alternativen, die auch im Automobilsektor einsetzbar sind [BS13] [Wol18].

In der Softwareentwicklung werden Methoden wie SCRUM oder Kanban als Standardvorgehen erfolgreich eingesetzt. Die Erfahrung zeigt, dass sich die Projekte flexibler gestalten lassen. Außerdem ist eine frühzeitige Produktoptimierung und Fehlererkennung durch Integrationspipelines und automatisierte Testvorgänge möglich. Die kurzen iterativen Entwicklungsphasen von wenigen Monaten zählen zu den Grundzügen agiler Methoden. Dadurch kann auf neue und geänderte Anforderungen während eines Projekts besser und schneller reagiert werden und permament die richtige funktionsweise des Produktes getestet werden. So soll bei jeder Iteration ein potentiell lieferfähiges Produktinkrement generiert werden. Davon will auch die Automobilbranche profitieren [Bor18], [ADS+10], [Zan09].

Die Steuergerätesoftware kann in hohem Maße nach agilen Prinzipien entwickelt werden. In der Hardwareentwicklung können solche Vorgehensmodelle nur schwer adaptiert werden. Denn vor allem die Hardware ist unmittelbar davon abhängig, dass die Software für die Ansteuerung der Elektronik rechtzeitig für das Testen der Funktionalität bereitgestellt wird. Die korrekte Funktionsweise der Software muss demensprechend auch in Verbindung mit der Hardware überprüft werden. So entstehen Abhängigkeiten, die einen planmäßigen Entwicklungsverlauf hemmen. Bei einer kleinen Anzahl an Steuergeräten wären solche Problematiken möglicherweise auch in kürzeren Releasephasen beherrschbar. Wenn man aber bedenkt, dass im heutigen Automobil durchschnittlich mehr als 40 Steuergeräte verbaut werden, zeigt sich schnell, mit welcher Intensität sie die Releasezyklen belasten. Hinzu kommt, dass die hohe Anzahl an Steuergeräte nach einem Teile und Herrsche Prinzip, sodass die Komplexität auf Bauteilebene heruntergebrochen wird und von mehreren Teams in einem Simultaneous Engineering Prozess quasi-parallel entwickelt werden. Oft arbeiten sie mit unterschiedlichen Projektmanagement und Entwicklungsmethodiken die zusammen mit der übergeordneten Releaseplanungssystematik vereinbart werden müssen. Somit baut sich ein Spannungsfeld zwischen klassischer und agiler Entwicklung auf, dass das Management vor neuen Herausforderungen stellt [Sch05], [Her14], [Wol18].

## 1.2. Herausforderungen für das Projektmanagement

Die steigende Komplexität der im Automobil verbauten Komponenten erfordern mit der Zeit neue Projektmanagementmaßnahmen. Vor allem die Synchronisation hybrider Projektstrukturen, mit unterschiedlichen Entwicklungs- und Projektmanagement Methodiken, stellt die Automobilbranche

und die Releaseplanung vor neuen Herausforderungen. Die immer kürzer werdenden Produktzyklen und der unhaltsame Wachstum des Wettbewerbs fordern neue Maßnahmen. Daher stellt sich die Frage nach der Zukunftsfähigkeit der implementierten Prozesse. Damit dies keine existentielle Bedrohung für den Automobilmarkt wird, sieht sich diese Branche dazu gezwungen den Grad der Agilität und Flexibilität der verwendeten Prozesse zu erhöhen.

## 1.3. Ziel der Arbeit

Mit Hilfe einer Expertenumfrage soll die Meinung der Industrie über die vorherschenden Herausforderungen in der Releaseplanung befragt werden. Die Dr. Ing. h. c. F. Porsche AG gilt als Referenzunternehmen. Eine Analyse, die den aktuellen Stand der Releaseplanungssystematik der Porsche AG wiederspiegelt, wurde a priori durchgeführt. Daraus entstand ein Fragebogen mit 35 Fragen. Darauf basierend wurde anschließend eine Online Umfrage durchgeführt und ausgewertet. Ziel ist es Experten aus der Automobil- oder einer verwandten Branche zu befragen, die einen Releaseprozess adaptieren und in diesen involviert sind.

## 1.4. Aufbau der Arbeit

Im nachfolgenden Kapitel wird der Stand der Technik der Produktentstehung im Automobilsektor und die Entwicklung sicherheitskritischer Komponenten vorgestellt. Es folgen verwandte Arbeiten zur Verbreitung agiler Praktiken im Automobil Umfeld sowie Studien über hybride Strukturen im Release Prozess. Hier ist anzumerken, dass es relativ wenige Vorreiter Studien gibt, die sich mit dieser Problemanalyse befasst haben. Anschließend wird die Durchführung der Interviews vorgestellt, sowie die Analysemethodik der *Grounded Theory*. Die anschließende Auswertung der Ergebnisse sowie Fazit und Ausblick bilden den Abschluss dieser Arbeit.

# 2. Stand der Technik

In diesem Kapitel wird der Stand der Technik für die Fahrzeugentstehung, die Entwicklung sicherheitskritischer Systeme in der Automobilbranche sowie die heutige Rolle agiler und klassicher Methoden im Produktentstehungsprozess (PEP) dargestellt.

## 2.1. Der klassische Produktentstehungsprozess in der Automobilindustrie

Die Entwicklung eines Fahrzeugs ist ein hoch komplexes Vorgehen, das hochgradig simultan abgewickelt wird [Gün07]. Ein detailliertes Vorgehensmodell für die Entstehung eines Automobils bildet der Produktentstehungsprozess (PEP). Dies ist ein Phasenkonstrukt mit einer semi-sequentiellen Struktur, wie sie in Abbildung 2.1 zu sehen ist.

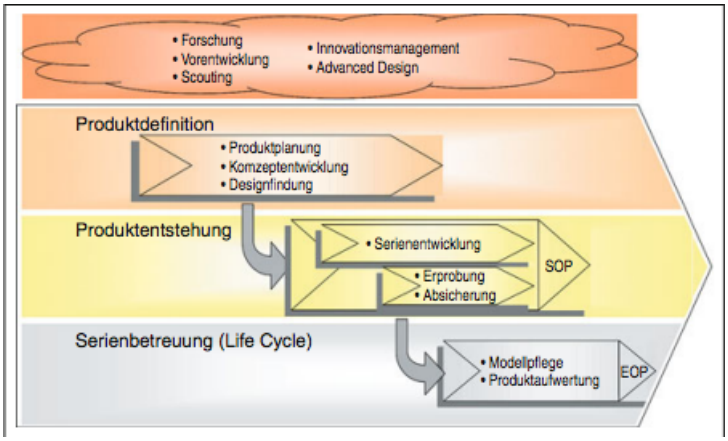


**Abbildung 2.1.:** Schematische Darstellung eines Produktentstehungsprozesses. [BBW+16]

Wechselnde Marktbedingungen und die individuellen Produktstrategien der Automobilhersteller (engl. Original Equipment Manufacturer, OEM) führen zu Variationen im PEP. Der klassische PEP umfasst drei Hauptphasen - die Produktdefinitons- , die Produktentwicklungs- und die Serienbetreuungsphase [BS13]. Während der Produktdefinitonsphase werden alle technisch relevanten Anforderungen an das Produkt definiert. Sie werden aus einem kontinuierlichen Prozess von Forschung, Vorentwicklung, Scouting etc., ins Leben gerufen. Daraus resultiert eine hoch detaillierte Produktdefinition, die in Form von technischen Lastenheften in die Produktentwicklungsphase

einfließt. In der Produktentwicklungsphase findet anschließend die Entwicklung der elektronischen Steuergeräte und mechanischen Komponenten statt. Mit dem Start of Production (SOP) beginnt die Serienbetreuung, die das Automobil in seinem restlichen Lebenszyklus begleitet [BS13].

### 2.1.1. Lebenszyklus eines Automobils

Der Lebenszyklus eines PKWs beginnt mit der Produktentstehungsphase und lässt sich in 3 Abschnitte aufteilen (Abbildung 2.2).

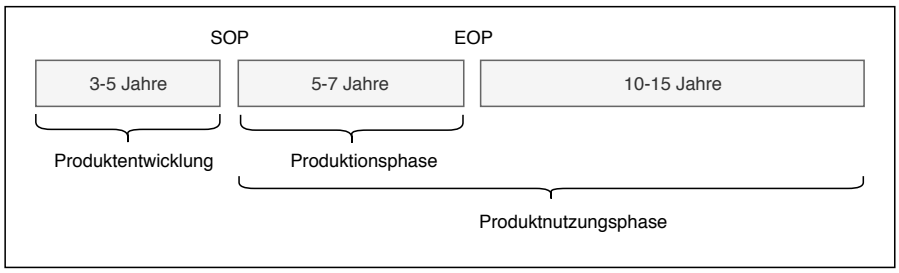


**Abbildung 2.2.:** Lebenszyklus eines Automobils eigene Darstellung.

Die Entwicklungsdauer beträgt heute typischerweise drei bis fünf Jahre. Mit dem Start of Production (SOP) beginnt die Produktionsphase des Fahrzeugs, die zwischen fünf bis sieben Jahre dauert. Zusammen mit einer Phasendauer von 10-15 Jahren ergibt sich der Gesamtlebenszyklus des PKWs [Sch14]. Der Markt ist lebendig und durch sich schnell ändernde Bedingungen und Nachfolgemodelle verkürzt sich die Nutzungsdauer des Automobils tendentiell. Heute beträgt sie durchschnittlich fünf bis sechs Jahre [MSG07].

### 2.1.2. Strategische Verkürzung der Produktentwicklungsdauer

Die Automobilhersteller versuchen ihre Prozesse ständig zu optimieren und an die aktuellen Gegebenheiten anzupassen. Die zukünftig eingesetzen Prozesse sollen möglichst flexibel auf Änderungen reagieren und Anforderungen schnell umsetzen. Gerade am PEP wird versucht die nötigen Maßnahmen einzuleiten, um kosteneffizienter, qualitativ hochwertig und konkurrenzfähiger zu werden. Die größten Optimierungspotentiale herrschen in der Produktentwicklungsphase, die immer strategischer angegangen wird. Um mit dem Puls des Marktes mithalten zu können wird deshalb eine Verkürzung dieser Phase angestrebt, um die Marktreife schneller zu erreichen [Tie03].

## 2.2. Die Produktentwicklungsphase

Nachfolgend wird der Stand der Technik für die Produktentwicklung vorgestellt. In der Automobilbranche gilt es auch heute die perfekte Relation zwischen Zeit, Kosten und Qualität zu finden. Deshalb werden ständige Prozessoptimierungen durchgeführt, um diese Faktoren möglichst auszubalancieren. Insbesondere die Serienanfertigung, nimmt im Rahmen der Produktentwicklung

eine zentrale Rolle ein und muss genauestens koordiniert und geplant werden [Tie03] [Wes03]. Optimierungsmaßnahmen in laufenden Geschäfsprozessen kann eine Organisation auf 3 Ebenen durchführen - Strukturverbessernde Maßnahmen (Organisationsebene), Prozessverbessernde Maßnahmen (Prozessebene) und die Verbesserung der Arbeitssteuerung (Operative Ebene). Diese sind i.d.R. auch die Entitäten die in einem Projekt vereint sind [Wei06]. Um die *Prozessebene* und die *Operative Ebene*, mitsamt aller Testphasen zu vereinen, werden gemeinsame Ziele auf *Organisationsebene* definiert, die in Form von Meilensteinen auf Prozessebene festgehalten werden. So werden Fertigstellungstermine für Anforderungen definiert, die je Meilenstein einen bestimmten Reifegrad sowie Qualitäts- und Sicherheitslevel erreichen müssen [BS13]. Auf operativer Ebene orientieren sich alle Fachbereiche und Projektteams danach, die in einem Simultaneous Engineering (SE)-Prozess semi-parallel an der Anforderungsumsetzung arbeiten [BS13] [Wol18]. Die Meilensteinstruktur wird in einer ausführlichen Releaseplanung (siehe 2.2.3) über die gesamte Dauer des Projektes detailliert. Im weiteren Verlauf werden Vorgehensmodelle vorgestellt, die sich mit der Zeit für die Entwicklung (sicherheitskritischer) Systeme herauskristallisiert haben.

### 2.2.1. Vorgehensmodelle

Hochkomplexe und sicherheitskritische Systeme, wie das Automobil einzustufen ist, bedarfen einer strukturierten Prozesssystematik [Her14]. Dafür werden Vorgehensmodelle eingesetzt, die durch die Definition genauer Prozessschritte die Komplexität bei der Projektplanung beherrschbar machen. So wird für eine effiziente Projektdurchführung gesorgt. Die ersten Vorgehensmodelle im Projektmanagement waren durch eine strenge Sequentialität karakterisiert mit eingeschränkter Möglichkeit, Änderungen nach Projektstart zu veranlassen. Mit der Erfindung des Wasserfallmodells von Royce im Jahre 1970 ([Roy87]) wurden erste wichtige Meilensteine gesetzt, zur Strukturierung nicht linearer Entstehungsprozesse. Eine Erweiterung des Modells um Rekursionspheile wurde von Boehm veranlasst [Böh05] (Abbildung 2.3).

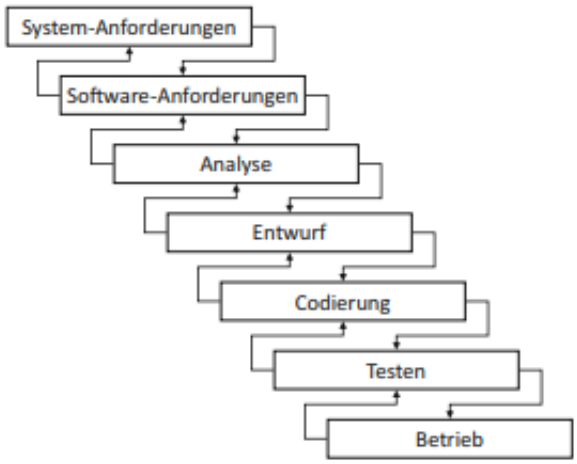


**Abbildung 2.3.:** Das Wasserfallmodell, angeleght an [Wol18].

Das Wasserfallmodell zeichnet sich durch seine kaskadenartige Hintereinanderreihung der Prozesse aus und einen rekursiven Feedbackprozess mit der unmittelbar vorherigen Phase aus [Böh05]. Weitere Modelle sind beispielsweise das Spiralenmodell, dass ebenfalls von Beohm erfunden wurde, als verbesserte Version des Wasserfallmodells [Sto14]. Die Modelle die bis dato existierten hatten ein großes Manko im Bereich des Testings. Gerade bei sicherheitskritischen Systemen wie das Automobil ist es von enormer Wichtigkeit jeden Prozessschritt testen und validieren zu können [Her14]. Mit der Erfindung des V-Modells wurde ein wichtiger Meilenstein für die Addressierung solcher Probleme gesetzt [Sch05].

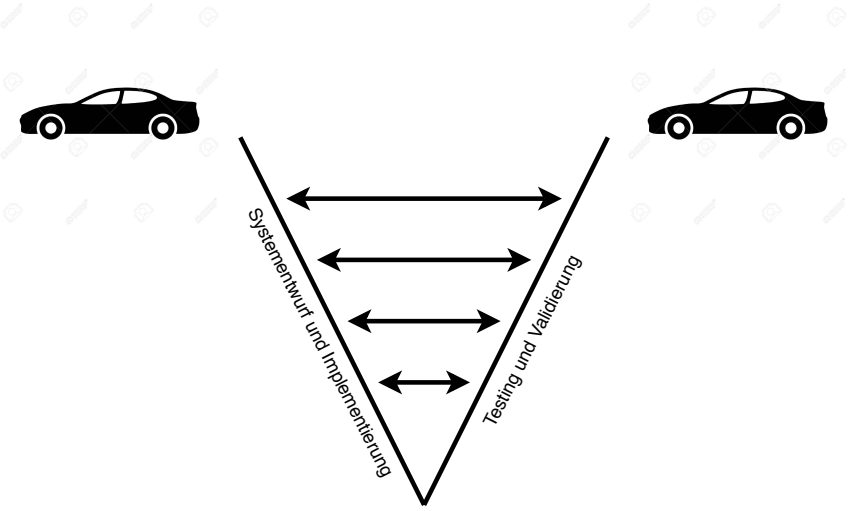


**Abbildung 2.4.:** Eine abstrakte Darstellung des V-Modells [Sch05].

Die unterste Spitze des V-Modells sieht vor, dass die unterschiedlichen Bereiche, Entwicklung und Testing, zu einem bestimmten Zeitpunkt im Projektverlauf zusammenfinden. Nachdem die Anforderungen auf Systemebene (Automobilebene) festegelegt sind, werden sie anschließend auf Bauteile- bzw. Modulebene heruntergebrochen. Auf dieser Grundlage werden später die Anforderungen auf die einzelnen Meilensteine verteilt. So werde im Testing- und Validierungsschritt zunächst Subsysteme der Bauteile gebildet (z.B. Baugruppen), die dann später mit anderen Subsystemen zusammengefürt werden, bis das gesamte Fahrzeug entstanden ist [BBW+16]. Dies wird als Bottom-Up Prozess bezeichnet [Rei14]. Dies fordert insbesondere, dass, bevor ein Gesamtsystem oder ein Subsystem getestet werden kann, alle Komponentenbeschreibungen und Validierungsschritte erstellt und implementiert worden sein müssen [BBW+16]. Sein großer Vorteil liegt in seinem V-förmigen Aufbau. Es erlaubt die Ergebnisse der Entwicklung und die des Testings gegenüberzustellen, was auch als *bilaterale Rückverfolgbarkeit von Anforderungen* bezeichnet wird [Rei14], [JK16]. Diese Eigenschaften haben dazu geführt, dass sich das V-Modell in weiten Bereichen der Automobilbranche als quasi-Standard durchgesetzt hat [FG13].

### 2.2.2. Hauptphasen

Um eine systematische Abarbeitung der Hauptphasen des PEP sicherzustellen, greifen Unternehmen oft auch auf ein Phasenmodell zurück, dass auf eine Zeitachse abbildbar ist (Abbildung 2.5).

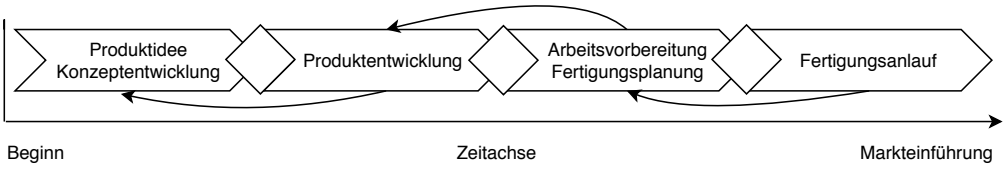


**Abbildung 2.5.:** Traditioneller Entwicklungsverlauf

Die Hauptphasen sind durch Meilensteine getrennt (siehe Rauten in Abbildung 2.5), die auch als Quality Gates bezeichnet werden [BBW+16]. Sie werden dort eingeführt wo das Produkt, durch die Erfüllung vordefinierter Qualitäts- und Sicherheitskriterien, einen bestimmten Reifegrad (siehe 2.2.3) erreichen muss, um in die nächste Phase transitieren zu können und damit ein *Freigabeprozess* stattfinden kann. Die sequentielle Anordnung erzwingt den Abschluss einer Phase bevor die unmittelbar nachfolgende freigeschaltet wird. Der Informationsfluss von einer in die nächste Phase findet häufig nur über diese Transition statt. Rekursionspfeile helfen Rückkopplungen und Feedbackmachanismen zu integrieren, die in einen nicht-linearen Projektverlauf oft notwendig sind [BS13], [BBW+16], [Wol18].

### 2.2.3. Der Freigabeprozess

Ein Freigabeprozess wird durch eine detaillierte Releaseplanungssysthematik definiert. Sie wird insbesondere bei nicht-linearen Entwicklungsverläufen eingesetzt, um ein komplexes Produkt auf Teilprodukte herunterzubrechen und den Entwicklungsfortschritt durch eine regelmäßige Taktung im Prozess zu koordinieren. Dieses Vorgehen ist auch als *Divide and Conquer*-Ansatz bekannt [Zan09]. Für eine exakte Planung ist die frühzeitige Einbringung aller Lieferanten und Prozessbeteiligten sehr wichtig. Im Idealfall erzielt man damit eine erhebliche Verkürzung des Entwicklungszeitraumes und eine Verbesserung der Qualität. Die relevanten Anforderungen werden in einem Requirement Engineering (RE)-Prozess gesammelt. Dies geschieht meistens in enger Zusammenarbeit mit den Stakeholdern und Methoden wie Forschung, Vorentwicklung, Scouting und Design Management. Desweiteren werden sie maßgeblich durch diversen Marktanalysen und Steuergrößen wie die Unternehmenseigene Strategie bestimmt. Sie werden anschließend in technischen Produktbeschreibungen umgewandelt und auf Ebene der einzelnen Bauteile heruntergebrochen, so dass funktionale Lastenhefte entstehen. Die technischen Zielwerte der einzelnen Funktionen werden während der Entwicklungsphase kontinuierlich getestet und validiert (siehe *V-Modell* in Kapitel 2.2.1). Ausgehend von den technischen Baulastenheften werden die Meilensteine definiert. Hier wird terminlich festgelegt, wann welche Anforderung an das Produkt bereitstehen muss. Die konzeptionellen Anforderungen werden anschließend von einem SE-Team (siehe Kapitel 2.2.4) technisch umgesetzt [BS13], [BBW+16],[Wol18].

### Meilenstein

Meilensteine sind wichtige Elemente der Releaseplanungssysthematik. Sie begünstigen eine Synchronisation aller unterliegenden Prozesse. Durch ihre Definition werden Prüfstellen im Prozess integriert. Dort werden vordefinierte Qualitäts- und Sicherheitsaspekte, sowie Anforderungsumsetzungen auf ihre Erfüllung überprüft. Nur so kann die nächste Phase freigeschaltet und ein Fortschritt im Projekt erzeugt werden. Es entsteht somit ein Synchronisationsmechanismus, der die Leistungsabgaben aller SE-Teams zusammenführt. Eine erfolgreiche Meilensteinsturktur bzw. Releaseplanungssysthematik bestimmt maßgeblich die Dauer der Produktentwicklungsphase. Sie kann je nach Aufgabenumfang oder außerplanmäßigen Nachlieferungen von Anforderungen bzw. Funktionalität variieren. Ein Synchronisationspunkt umfasst unterschiedliche Aufgaben (Abbildung 2.6). Der Prozess wird durch die Abgabe der entwickelten elektronischen Steuergeräte und Softwarekomponenten der operativen Ebene initiiert (siehe 2.2.4). Danach findet eine Integrationsphase aller Komponenten in eine Fahrzeugumgebung statt (siehe 2.2.3). Dort werden die Komponenten zunächst zu Subsystemen und anschließend zu einem Gesamtsystem zusammengeführt. Nach jedem Zwischenschritt findet ein Verbundtest statt der fertigen Komponenten [BS13], [BBW+16], [Wol18].

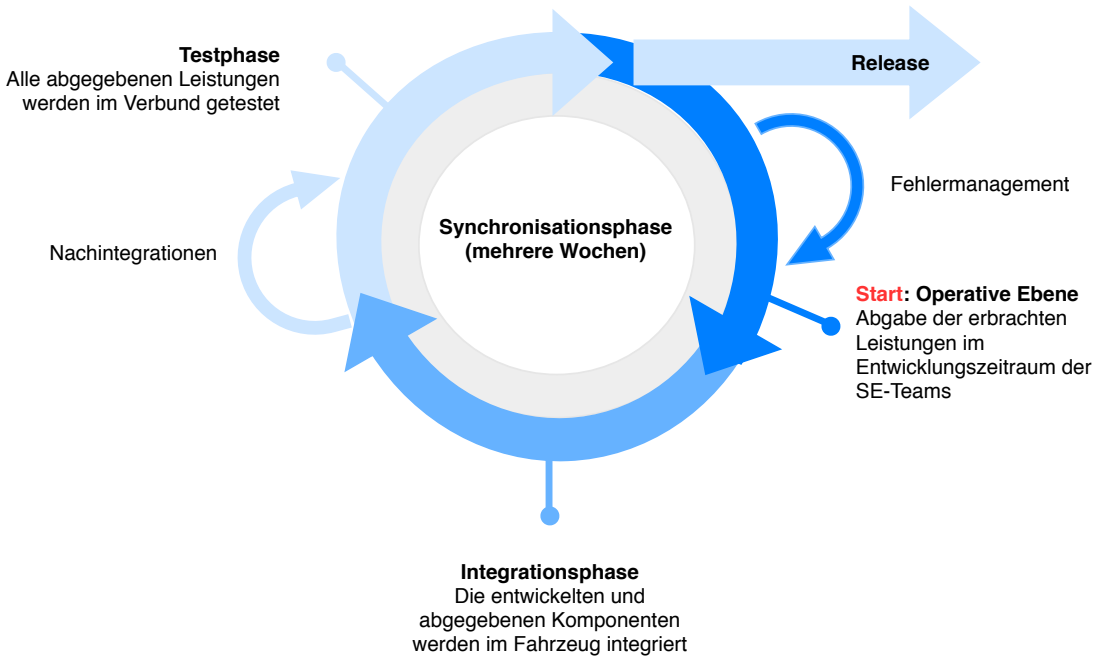


**Abbildung 2.6.:** Eine eigene Darstellung der Meilensteinstruktur.

#### Testabläufe im Meilenstein

Die erbrachten Leistungen werden unterschiedlichen Tests unterzogen. So wie die funktionalen Anforderungen auf Bauteileben heruntergebrochen werden, um das System besser zu beherrschen, werden sie auch wieder Schritt für Schritt zusammengeführt und validiert. Alle nachfolgenden Tests finden zu jedem Meilenstein statt und werden in Abbildung 2.7 grafisch vorgestellt [Wol18].

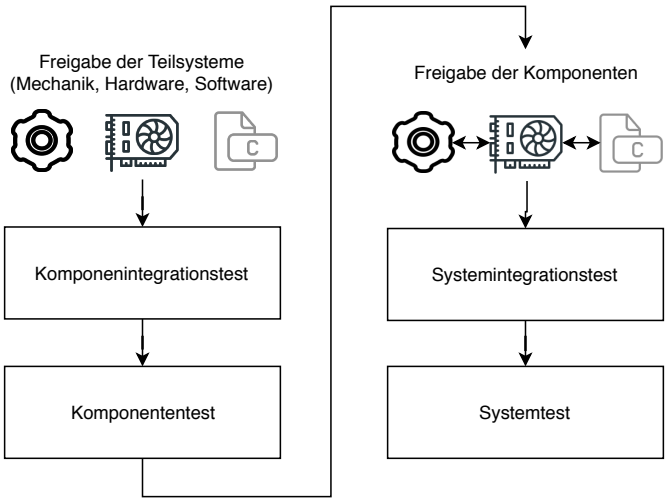


**Abbildung 2.7.:** Meilenstein

Die Freigabe der Teilsysteme Mechanik, Hardware und Software initiiert den Komponentenintegrationstest. Hier werden alle Komponenten nach zuvor definierten Schritten zusammengeführt. So soll in erster Linie die FUnktionalität in Kombination mit den anderen Komponenten getestet werden, um ebentuelle Inkompatibilitäten und Inkonsistenzen zu entdecken. Im nächsten Schritt, findet der Komponententest statt, der direkt auf vollständig integrieren Kompoenenten durchgeführt wird. Hier wird die Komponente getestet und in Abwesenheit von Fehler freigegeben. Ziel des Komponententests ist es nachzuweislich die korrekte und vollständige Umsetzung der Komponentenanforderungen zu bestätigen [Wol18]. Anschließend findet ein Systemintegrationstest statt. Inkonsistenzen zwischen allen verbundenen Komponenten werden mit Hilfe des Integrationstest identifiziert. Im Anschluss und als letzten Schritt wird das Gesamtsystem freigegeben. Hier findet die korrekte und vollständige Überprüfung der Systemanforderungen statt. Spezfische Tests werden in Kapitel 2.3.2 vorgestellt [Wol18].

#### Reifegradmodelle

Um den Fortschritt der Entwicklung zu quantifizieren, werden Reifegradmodelle eingesetzt. Unter den bekannstesten Reifegradmodellen für die Software findet man das “ Capability Maturity Model (CMM)” [PCCW93]. Das Model kategorisiert die Reife in 5 Stufen:

1. *initial*: Prozess besitzt eine kaotische und untransparente Vorgehensweise.
2. *repeatable*: Prozess ist wiederholbar und hat Quality Gates/Meilensteine.

3. *defined*: Prozess ist definiert und als Organisationsweit gültiger Software-Prozess eingeführt.
4. *managed*: Die Qualität der Produkte und Produktivität der Prozesse wird durch ein organisationsweites Metrikprogramm quantitativ gemessen. Das Management hat dadurch eine objektive Basis, um Entscheidungen treffen zu können.
5. *optimizing*: Ständig neue und verbesserte Wege der Softwareentwicklung werden erprobt. Die gesamte Organisation ist auf kontinuierliche Prozessverbesserung eingestellt.

Als Erweiterung entstand das Capability Maturity Model Integration (CMMi), [CKS03]. Das CMMI beinhaltet *Software Engineering CMM*, *System Engineering CMM*, *Integrated Product und Process Developement* und fasst alles in ein Reifegradmodell zusammen [CKS03]. Somit wurde die Kompatibilität mit dem Software Process Improvement and Capability Determination (SPICE)-Standard erreicht.

SPICE ist eine internationale Förderungsinitiative die im ISO/IEC 15504 Standard enthalten ist [WD08]. Dieser stellt einen internationalen Standard für die Bewertung von Software (Entstehungs-)Prozesse dar. Für die Automobilbranche entstand zudem der Standard Automotiv SPICE Automotiv Software Process Improvement and Capability Determination (ASPICE), [MHDZ08].

#### ASPICE

In der Softwareentwicklung wird zur Bewertung von (Entwicklungs-)Prozessen der Standard SPICE bzw. die ISO/IEC 15504 [WD08] verwendet. Diese Standards haben als Anhaltpunkt die ISO-Norm ISO/IEC 12207 *Prozesse im Software-Lebenszyklus*. Für die Bewertung des (Entwicklungs-)Prozesses im Automobilbereich, gibt es eine domänenspezifische Variante des Standards unter dem Namen ASPICE [MHDZ08]. Dabei wird der (Entwicklungs-)Prozess in eine der fünf SPICE-Level eingestuft.

### 2.2.4. Operative Projektebene

Die Releaseplanung wird von der operativen Projektebene auf Bauteilebene umgesetzt [BBW+16]. Hier arbeiten eine Vielzahl an Teams, aus unterschiedlichen Fachbereichen (z.B. Fahrwerk, Elektrik/Elektronik etc.), sowie externe Lieferanten, in einem SE-Prozess quasi-parallel an den einzelnen Funktionen- und Kernkomponenten des Produktes [BS13]. Sie werden als Cross-Funktionale Teams bezeichnet, die integrierte Mechanik- und Softwareentwicklung betreiben [DK01]. Mit SE-Teams wurden bereits wichtige Managementrichtlinien eingeführt [BBW+16]. Aus ihren Leistungen entstehen mechatronische Systeme, die aus den Bereichen Mechanik, Elekrotechnik und Informatik zusammengeführt werden. Auf dieser Ebene finden immer mehr hybride Sturkturen Einsatz (siehe Kapitel 2.4), die klassische und agile Projektmethoden vereinen [**NuhnMartini2016**].

Dies stellt das Management automobiler Projekte vor neuen Problematiken. Besonders die Releaseplanung in hybriden Projekten, gekoppelt mit der Anforderungsumsetzung und der anschließende Gesamtsystemtest, stellt eine große Herausforderung dar. Denn Abhängigkeiten zwischen den involvierten Komponenten aus den Bereichen Mechanik, Elektrotechnik und Informatik, erschwerten einen reibungslosen Gesamttest und termingerechten Releasezklus. Eine Folge der genannten Problematiken und oft auch eine notwendige Maßnahme auf operativer Ebene, ist die *technische Schuld*, die nachfolgend erklärt wird [KNO12].

### Technische Schuld

Die Defintion der technischen Schuld stammt aus der Softwareentwicklung, kann aber in diesen Kontext, durch die Vernetzung der Bereiche, auch für die Hardware verwendet werden. Eine technische Schuld entsteht durch eine suboptimale technische Umsetzung von Software bzw. Hardware. Unter technischer Schuld versteht man den zusätzlichen Aufwand, den man für Änderungen und Erweiterungen an schlecht geschriebener Software im Vergleich zu gut geschriebener Software einplanen muss. Es werden folgende Arten von technischen Schulden unterschieden [KNO12].

- rücksichtlos (bewusst)
- umsichtig (bewusst)
- rücksichtlos (versehentlich)
- umsichtig (versehentlich)

Eine bewussten technische Schuld geht man ein z.B. um ein Meilenstein zu erreichen. Eine unwissentliche technische Schuld geht man ein, bspw. weil das Team gängige Codingrichtlininen nicht befolgt. Eine rücksichtslose technische Schuld entsteht, wenn bspw. die Bugfixingphase unzureichend eingeplant wurde. Eine umsichtige technische Schuld entsteht, wenn bspw. bewusst suboptimale Softwarelieferstände abegeben werden.

## 2.3. Sicherheitskritische Elektroniksysteme

Im weiteren Verlauf der Arbeit wird vor allem auf die Entwicklung sicherheitskritischer Elektroniksysteme eingegangen. Sicherheitskritische Systeme inkludieren die Gefahr, dass bei Fehlern oder Systemausfälle Menschenleben unmittelbar gefährdet werden [Her14]. Deshalb fordern sie ein hohes Maß an Sicherheit und Zuverlässigkeit. Dies wiederspiegelt sich in den Methoden die für die Systementwicklung und den Systementwurf eingesetzt werden. Für die Releaseplanungssysthematik, wie sie in Kapitel 2.2.3 beschrieben wurde, werden oft Modelle angestrebt, die die Entwicklung standardisieren und gleichzeitig kostenoptimal gestallten. Es werden besonders Modelle bevorzugt, die eine Rückverfolgbarkeit von Anforderungen über den gesamten Entwicklungszeitraum ermöglichen. Denn die Zuverlässigkeit und Sicherheit des Systems muss zu jedem Zeitpunkt der Entwicklung kontinuierlich und stetig nachweisbar und rückverfolgbar sein. Die elektronischen Steuergeräte sind ein besonders wichtiger Bestandteil eines mechatronischen Systems und als sicherheitsrelevant eingestuft [Wol18] [Sch05]. Sie werden nach ihrer sicherheitskritischen Rolle, die sie im Fahrzeug ausüben, kategorisiert. Hierfür gibt es das ISO Standard 26262 [WEW15] für funktionale Sicherheit, worauf in Kapitel 2.3 näher eingegangen wird. Steuergeräte mit einem hohen sicherheitskritischen Level werden beispielsweise in der Motorsteuerung eingesetzt, denn sie könnten bei einer Fehlfunktion unter umständen Menschenleben in Gefahr setzen [Her14]. Ihre hohe Sicherheitsrelevanz bestimmt sich somit dadurch, dass sie deterministisch über die Dauer des Lebenszyklus eines Autos funktionieren und den Autofahrer in all seinen Vorhaben sicher unterstützen müssen. Bei sicherheitsrelevanten Systemen muss die Sicherheit des Fahrzeugs gleichermaßen wie die Systemfunktion berücksichtigt werden [WEW15]. Ein mechatronisches System besteht desweiteren aus den Teilen Mechnik und Software sowie Aktore und Sensoren die das System vervollständigen [Wol18]. In der Releaseplanung und insbesondere für die Test- und Validierungsphase der Steuergeräte im Meilenstein müssen alle nachfolgend definierten Regeln und Normen, sowie Sicherheitsstandards eingehalten werden. Insbesondere die Testphase, die für eine qualitative Durchführung der Tests (siehe 2.2.3 und 2.3.4) zuständig ist, nimmt eine besonders wichtige Rolle im Verbundrelease ein.

#### Funktionale Sicherheit - ISO-Norm 26262

Abhängig von der Kritikalität der Funktion werden bei der Entwicklung sicherheitskritischer Systeme Richtlinien für den Entwicklungs- und Freigabeprozess vorgegeben. Die ISO-Norm 26262 “Road vehicles - Functional safety” [WEW15] definiert vier Kategorien für die Einstufung der Kritikalität einer Funktion. Sie werden unter dem Akronym Automotive Safety Integrity Level (ASIL) zusammengefasst und in den Risikostufen ASIL-A, ASIL-B, ASIL-C und ASIL-D aufgeteilt. Der ASIL-Level wird jeweils zu Beginn eines Entwicklungsprozesses bestimmt. ASIL-A hat die geringste Risikostufe und ASIL-D ein zehntausendfach höheres Risikopotential gegenüber ASIL-A [WEW15]. Je nach ASIL-Level werden Maßnahmen zur Sicherstellung der Qualität der Funktion gefordert. Eine wichtige Forderung ist z. B. ein detailliert dokumentierter Entwicklungsprozess zur Sicherstellung der Qualität des Entwicklungsprozesses [WEW15]. ASIL bezieht sich auf die Verletzungen des Fahrers und anderer Verkehrsteilnehmer, im Falle eines Systemfehlers. Das ASIL-Level ist aus dem Safety Integrity Level (SIL) hervorgegangen. Das ist eine Einstufung für die Sicherheitsanforderung, die gemäß IEC 61508/IEC61511 [LR09] international normiert ist. Mit Einführung des Automotive-Standards ISO 26262 wurden die SIL- durch die ASIL-Spezifikationen ersetzt.

### 2.3.1. Automotive V-Modell

Als quasi-Standard für die Entwicklung sicherheitskritischer Elektronik hat sich in weiten Bereichen der Automobilindustrie das V-Modell durchgesetzt, wie bereits in Kapitel 2.2.1 eingeführt. Dieses ist nach der VDI-Richtlinie 2206 für die Entwicklung mechatronischer Systeme standardisiert und in Abbildung 2.11 schematisch dargestellt ([FG13],[PL11]).

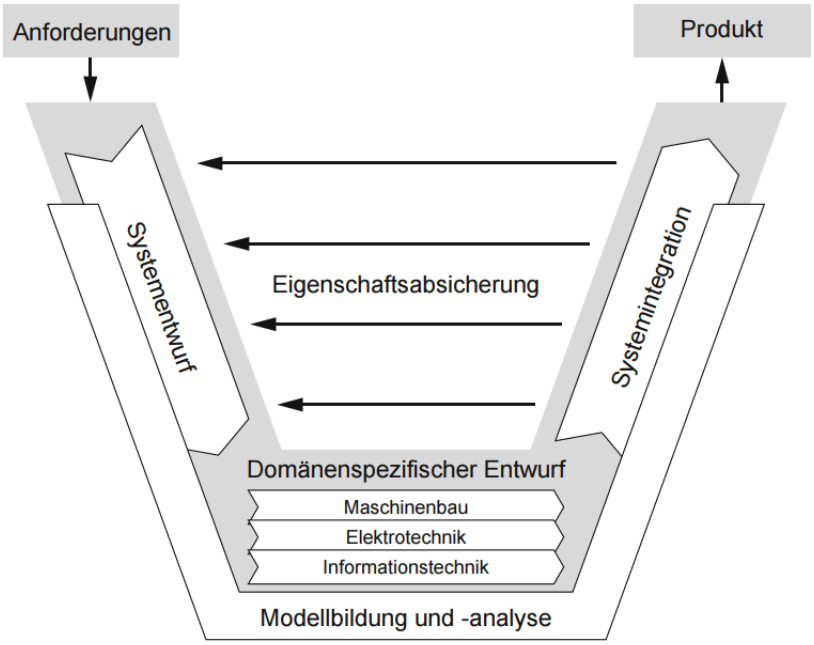


**Abbildung 2.8.:** V-Modell für die Entwicklung mechtronischer Systeme nach VDI 2206 [FG13].

Die Anforderungen fließen in einen Systementwurf hinein, der für alle drei involvierten Teildisziplinen *Maschinenbau*, *Elektrotechnik* und *Informationstechnik* spezifiziert wird. Wobei sie in diesem Kontext nicht als getrennte Einheiten betrachetet werden können. Denn die Software wird zur Ansteuerung der Elektronik eingesetzt die im nächsten Schritt die Mechanik ansteuert. Die dargestellte “Eigenschaftsabsicherung” entspricht einem Kontrollvorgang, der die entwickelte Anforderung zu definierten Zeitpunkten entgegen bestimmten Testanforderungen validiert, (siehe *bilateratle Rückverfolgbarkeit von Anforderungen* Kapitel 2.2.1). Ziel ist es ein vollumfänglich geprüftes Produkt mit hohen Sicherheits- und Qualitätsstandards zu produzieren [WEW15].

### 2.3.2. Steuergerätesoftware

Nachfolgend wird das V-Modell für die Entwicklung der Funktionen mittels Software vorgestellt, mit Hinblick auf die Schnittstellenbereiche Mechanik und Hardware. Die Komplexität der Steuergeräte kann in die Hauptbestandteile Steuergerätesoftware und Hardware dichotomisiert werden. Die Software die in den Steuergeräten eingesetzt wird dient hauptsächlich zur Verarbeitung von Sensordaten und zur Komponentenansteuerung durch Aktore [SZ16]. Dieser Teil des Steuergerätes, besteht aus den funktionalen Anforderungen die in einem Anforderungskatalog oder Lastenheft auf

Systemebene zusammengefasst und auf Bauteilebene/Funktionenebene heruntergerbrochen wird [Wol18]. Die heutige Steuergerätesoftware wird i.d.R. in der Programmiersprache C geschrieben oder aus Modellen generiert [Wol18]. Die Ergebnisse der einzelnen Fachdisziplinen werden nach Abschluss der Entwicklung in erster Linie zu Subsystemen und dann zu Systemen integriert, wie in Kapitel 2.2.3 bereits vorgestellt, und hinsichtlich der Erfüllung der an sie gestellten Anforderungen validiert.

### V-Modell für die Softwareentwicklung

Nachfolgend wird das V-Modell für die Entwicklung der Steuergerätesoftware vorgestellt (Abbildung 2.9).

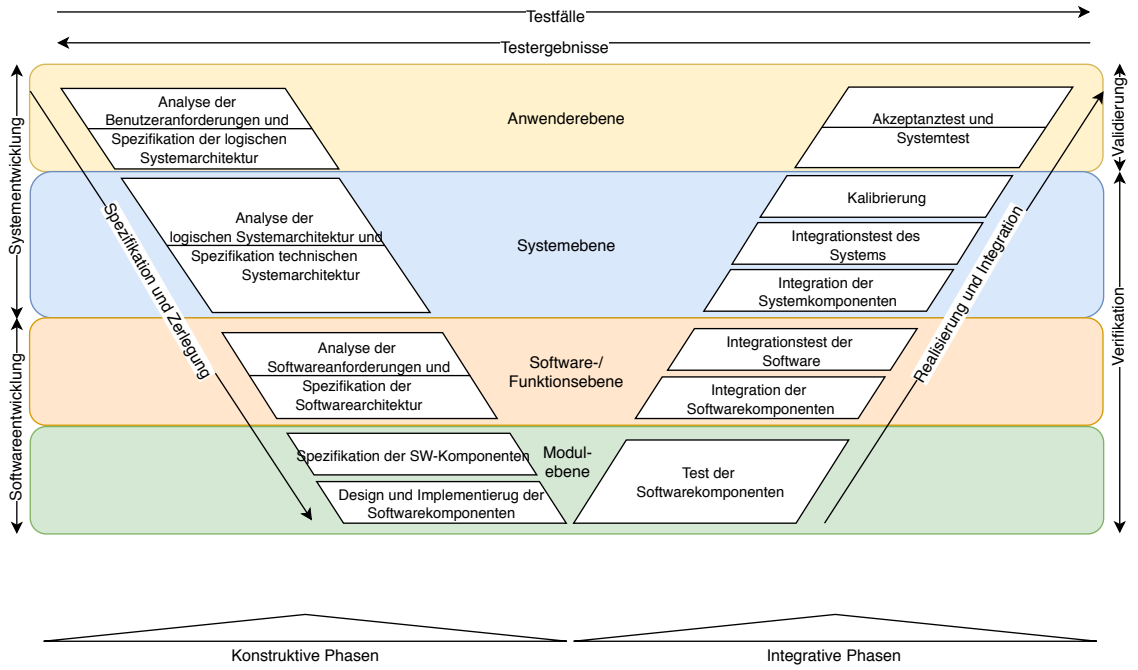


**Abbildung 2.9.:** Die Softwareentwicklung nach V-Modell. [Wol18]

### Spezifikation und Zerlegung

Zu Beginn steht die Vision des neuen Fahrzeugprojekts. Aus definierten Eigenschaften wie z.B. der Kraftstoffverbrauch oder die Art des Motors werden konkrete Anforderungen auf Anwenderebene detailliert. Daraus werden später die die Fahrzeugfunktionen abgeleitet. Sie dienen als Entwurfsbasis für die Spezifikation der technischen Systemarchitektur. Alle technischen Details und Spezifikationen an die Funktionen und Steuergeräte werden darauf basierend auf Software- und Funktionsebene detailliert. Hier wird besonders auf die Gestaltung der Steuergerätehardware und das *Funktionsmapping* Wert gelegt, wie in **??** bereits erklärt. Nach der Implementierung und Fertigstellung der Hard- und der Softwarepakete werden die Softwarekomponenten in die Steuergeräte integriert [**Wolf**].

#### Realisierung und Integration

Alle Prozessschritte auf der rechten Seite des V-Modells dienen der Qualitätssicherung. Auf Modulebene werden Einzelsteuergerätetests der Steuergeräte durchgeführt. Im folgenden Prozessschritt (Integration der Softwarekomponenten und Software) werden die Steuergeräte und die Softwarefunktionen im Verbund abgesichert. Dazu werden die Steuergeräte wie im realen Fahrzeug mit Hilfe der Bussysteme vernetzt. Nach der Integration der Steuergeräte ins Fahrzeug kann im letzten Prozessschritt validiert werden, ob die spezifizierten Eigenschaften des Fahrzeugs erfüllt werden.

### 2.3.3. Modellgetriebene Softwareentwicklung

Modellgetriebenen Softwareentwicklung (engl.: Model-Driven Software Development (MDSD), wird oft eingesetzt, um die Projektkomplexität frühzeitig zu beherrschen [Luí18]). Die Struktur und das Verhalten können Computergestützt, z. B. mit Hilfe von der Software MATLAB oder Simulink modelliert werden [CADR18]. Nachdem die Funktionsmodelle erstellt wurden kann die Softwarekomponente daraus generiert werden, die dann später im Steuergerät integriert werden kann.

#### Architekturstandards

Neben einem einheitlichen Entwicklungsstandard spielt auch die Architektur der Softwaresysteme und die Wiederverwendung von bestehenden Komponenten eine wichtige Rolle bei der Entwicklung der Steuergeratesoftware. Für die Entwicklung standardisierter und wiederverwendbarer Steuergeratesoftware ist unter einer Initiative der deutschen Automobilhersteller in den letzten Jahren eine einheitliche Architektur unter dem Namen Automotive Open System Architecture (AUTOSAR) entstanden [CADR18] (Abbildung 2.14). Zu den Gründungsmitgliedern gehören BMW, Bosch, Continental, DaimlerChrysler (heute Daimler), Siemens VDO und Volkswagen. Mit diesem Standard versuchen Automobilhersteller und Zulieferer, die Entwicklung von Steuergerätesoftware in Fahrzeugen zu vereinheitlichen. Hierbei wird eine logische Aufteilung der Steuergeratesoftware in steuergeratespezifische Basis-Software und steuergeräteunabhängige Anwendungs-Software vorgeschlagen. Die einzelnen Komponenten werden dabei über eine Middleware, die AUTOSAR-Laufzeitumgebung, verbunden [CADR18].

### 2.3.4. Testarten in der Automobilindustrie

Die Vorgestellten Testarten sind in modifizierter Form auch in anderen Industriezweigen zu finden, deren Fokus, neben der Entwicklung von Software, auch auf die Herstellung von mechatronischen Systemen und Komponenten liegt. Die Funktionsabsicherung der Steuergerätefunktionen nimmt eine besondere Stellung bei der Überprüfung der Gesamtsystemfunktionalität ein. Zumal ein positives Testergebnis der Nachweis für die korrekte Anforderungsumsetzung ist und ein hohes Sicherheitsniveau, durch die zugrundeliegenden Sicherheits- und Qualitätsstandards, erreicht wurde [ADS+10], [Her14]. Die Absicherung der Fahrzeugfunktionen soll hinsichtlich Kosten- und Zeitressourcen optimal, hoch automatisiert und zuverlässig ausführbar sein. Tests in simulierten Umgebungen werden als erste Testing Maßnahme bevorzugt. Sie sind unter dem Begriff *X-in-the-Loop* bekannt [ADS+10].

### X-in-the-Loop Testing

X-in-the-Loop Methoden umfassen Simulationstests wie Model-in-the-Loop-, Software-in-the-Loop-, Processor-in-the-Loop- und Hardware-in-the-Loop-Simulationen. Verteilte Funktionen auf verschiedene Steuergeräte, ebenso wie die beschriebene Funktionsweise in **??**, können mit Hilfe solcher Simulationen im Verbund vorzeitig getestet werden. Für den Test der Funktionen müssen die Steuergeräte wie im realen Fahrzeug vernetzt sein. Ebenso spielt der aktuelle Zustand des Fahrzeugs und der Fahrzeugumgebung eine entscheidende Rolle, da die Funktionen immer situativ reagieren. Mit X-in-the-Loop Methoden wird dem Manko des V-Modells entgegengewirkt, dass man ein System oder ein Subsystem erst testen kann wen alle Komponentenbeschreibungen erstellt und implementiert sind. Hierdurch kann eine Anwendung frühzeitig validiert werden, bevor Ergebnisse aus nachfolgenden Entwicklungsschritten vorliegen. Die verbreitesten X-in-the-Loop Testsvarianten sind *Model-in-the-Loop*, *Software-in-the-Loop*, *Process-in-the-Loop* und *Hardware-in-the-Loop* [ADS+10].

## 2.4. Agile Produktentwicklung

Die aktuellen Marktbedingungen sowie die Möglichkeit, mehrere Vorgehensweisen innerhalb einer Projektstruktur zu verbinden, um den projektspezifischen Anforderungen bestmöglich gerecht zu werden, führen zu Mischformen verschiedener Vorgehensmodelle. Häufig spricht man in diesem Kontext von *Hybride Projektstrukturen*. Sie vereinen Muster aus der klassischen und agilen Welt, um den heutigen Marktanforderungen gerecht zu werden. Darunter fällt der steigende Bedarf an Flexibilität und die schnelle Reaktion auf außerplanmäßige Anforderungen. Unter den Begriff klassisches Vorgehen versteht man Projekt- bzw. Entwicklungsmethoden, die plangetrieben sind und nach bestimmten, vordefinierten Prozessschritten arbeiten, wie sie in den vorgerigen Kapiteln (z.B. 2.2.1) vorgestellt wurden. Agile Methoden hingegen ermöglichen einen hohen Grad an Flexibilität bzgl. der Entwicklungs- und Planungssysthematik. Agile Methoden stehen im weiten Sinne nicht im kompletten Kontrast zu den klassischen Methoden. Sie werden im Großteil der Fälle, und vor allem in der Automobilbranche, als Erweiterung der klassischen Welt gesehen [Bor18]. Auch in der Automobilbranche finden *Hybride Strukturen* immer mehr Einzug. Auf operativer Ebene, in den SE-Teams, widerspiegelt sich dieses Phänomen in den angewendeten Entwicklungs- und Projektmangement Methodiken. Hier findet man häufig keine reine Anwendung von agilen Methoden, vielmehr angepasste Varianten agiler Methoden. Gerade bei der Entwicklung von Steuergeräten sind Aufgrund der involvierten Bereiche (Mechatronik, Elektrotechnik und Software) Komponenten, und somit von Cross-Funktionalen Teams, unterschiedlichen Hardware und Software Schwerpunkte zu erkennen. Unterschiedliche Bereiche führen dazu, dass der Grad der anwendbaren Agilität nicht immer gleich ist. Für die Steuergerätesoftware kann ein realtiv hoher Grad an Agilität angewendet werden, aufgrund der immateriellen Eigenschaften und leichte Änderbarkeit der Funktionalität durch Coding Maßnahmen. Dieser Agilitätsgrad für die Hardwareentwicklung ist, vor allem in der Automobilbranche eingeschränkt. Dies ist zum Teil auch der übergeordneten Meilensteinprozessstruktur zuzuschreiben, die die Einhaltung der terminlich festgelegten Releasezyklen auffordert und somit die Fertigstellung von Anforderungen und Bauteile zu bestimmten Meilensteinen verlangt. Die unmittelbare Abhängigkeit der Hardware und Software erfordert das vorhandensein beider Komponenten zum Zeitpunkt des Testen und schränkt die Flexibilität des Prozesses ein. Denn die Hardware ist von den Softwarefunktionen, die zur Ansteuerung der Elektronik eingesetzt wird, abhängig. Die korrekte Funktionsweise der Software muss demensprechend auch in Verbindung mit der Hardware überprüft werden. Besonders der Testvorgang kann dadurch nicht unabhängig und hochgradig automatisiert ablaufen. Diese Phänomene haben eine direkte Auswirkungen auf den Produktentwicklungsprozess und die Entwicklungszeiten. Um klassische und agile Methoden genauer zu differenzieren, wird im nachfolgenden Kapitel auf die Unterschiede dieser beider Welten eingegangen [BH17].

## 2.5. Agile Projektorganisation und -Strukturen

### 2.5.1. Requirementmanagement

Im Gegensatz zur Anforderungserhebung in plangetriebenen Projekten, wie sie in Kapitel 2.1 vorgestellt wurde, stehen die Anforderungen in agilen Projekten in der Regel nicht komplett fest, bevor mit der Umsetzung begonnen wird. Die Prozesse ermöglichen es nach jeder Implementierungsphase ein potentiell lieferfähiges Produkt zu generieren und veranlassen somit ein ständiges Testen der

Zischenprodukte [BLM+18]. So haben Anwender die Möglichkeit, frühzeitig und regelmäßig Produktversionen zu testen und ihre Anforderungen in laufenden Projekten anzupassen und zu erweitert. Details werden möglichst erst direkt vor der Umsetzung zwischen Anwendern und Entwicklern diskutiert und festgelegt. Die Reihenfolge der Umsetzung von Anforderungen ist dynamisch und richtet sich nach einer vom Kunden vorgegebenen Priorität gemäß seiner Bedürfnisse.

### 2.5.2. Projektteams

*Klassische Teams* sind viel Prozessgetriebener und führen Anweisungen von der unmittelbar Übergeorndeten Projektebene aus. Eine klassische Prozessstruktur in großen, verteilten Projekten bringt den Vorteil, dass sich die Fachbereiche und SE-Teams auf ihre Arbeit konzentrieren können, die dann vom Meilensteinprozess abgefangen und überprüft werden [Wol18].

In *agilen Projekten* hingegen wird viel Wert auf die Flexibilität und Einbeziehung aller Stakeholder zu jedem Zeitpunkt des Projektes gelegt. Agile Teammitglieder sind es gewohnt, selbstständig zu arbeiten und einen hohen Grad an Eigenverantwortung zu tragen [Tie14]. Ihr Einsatzgebiet ist nicht nur auf ihre Rolle beschränkt. Sie können beispielsweise, vor allem in der Softwareentwicklung, neben ihrer Entwicklungstätigkeit auch die Testerrolle übernehmen. Dadurch sind sie vielfältig einsetzbar. Im optimalen Fall haben sie unterschiedliche Fähigkeiten in sich vereint und sind ohne äußere Einwirkungen und Direktiven dazu in der Lage, diese für die Umsetzung von Anforderungen zielgerichtet einzusetzen. Im *Agile Manifesto* sind Leitsätze und Werte agiler Teams zusammengefasst, die erstmals im Februar 2001 veröffentlicht wurden [BLM+18]. Dort werden Prozessbestandteile priorisiert, so dass beispielsweise die Kommunkation im Team einen höheren stellenwert als die Dokumentation bestimmter Arbeitsschritte einnimmt. Die Prinzipien finden mit hoher Varianz in den einzelnen Methoden Einsatz.

#### Kommunikation in agilen Projektteams

Die Kommunikation in *agilen Teams* erfolgt viel häufiger und ist viel aktiver im Prozess integriert. Viele agile Vorgehensmodelle, wie sie in Kapitel 2.5.4 vorgestellt werden, führen bewusst Kommunikationsphasen im Prozess ein, die regelmäßig stattfinden. Sie sind somit prozessbegleitend und enstehen nicht nur in Extremsituationen oder mit den unmittelbaren Schnittstellen, wie das für prozessgetriebene, klassischen Teams oft der Fall ist. Häufig wird ein bewusster Austausch zwischen unterschiedlichen Fachbereichen gewünscht und herbeigerufen, um den Informationsfluss zwischen Fachfremden und Fachverwandten Abteilungen zu steigern [BLM+18], [Hen10] .

### 2.5.3. Releaseprozess

In der Softwareentwicklung wird ein Prozessablauf durch Iterationen und Releases gegliedert. Vor und während einer Iterationsphase werden nach bestimmten Kriterien, wie z.B. Daily Stand-Up Meetings etc. die zu entwickelnden Anforderungen geplant, programmiert und getestet. Während einer Iteration wird ein sog. *Productincrement* erstellt. Hierbei wird die Funktionalität des Programmes um Features erweitert. Am Ende dieser Phase, die in der Regel 2-3 Wochen dauert, steht ein potentiell lieferfähiges Produktinkrement zur Verfügung, das vom Kunden regelmäßig getestet wird. Die Anzahl der notwendigen Iterationen und Releases muss nicht von Beginn an feststehen,

sondern kann während des Projektes dynamisch angepasst werden [BLM+18]. Im agilen Umfeld wird das Testen während der Entwicklung durch sogenannte Integations-Pipelines unterstützt. In der Software wird oft die *Continous Integration*, also das ständige Integrieren von Teilergebnissen im Programmcode, angewendet die zur Folge die *Continous Delivery* hat [Hen10].

### 2.5.4. Agile Projektmanagementmethoden

Nachfolgend werden agile Vorgehendmodelle vorgestellt, die die oben geannten Prinzipien beinhalten []. Agile Methoden wurden Anfang der 90er Jahre entwickelt und vorwiegend im Bereich der Softwareentwicklung angewendet []. Darauf basierend sind mit der Zeit einige wichtige Methoden entstanden, die mit Erfolg eingesetzt werden und auch in der Automobilbranche ein immer größeres Interesse generieren. Die bekanntesten Methoden in der Agilen Szene werden nachfolgend vorstellt.

#### Kanban

Das Wort Kanban stammt aus dem Japanischen und bedeutet wörtlich übersetzt *Schild* oder *visuelles Signal* [BLM+18], [Hen10]. Diese Vorgehensmethode wurde im Jahre 1999 von David Anderson und Jeff De Luca entwickelt. Mit Kanban werden keine strickten Vorgaben gemacht, welche Rollen zu besetzen sind, welche Artefakte verwendet werden müssen oder welche Entwicklungspraktiken angewendet werden sollen. Die Methodik wirkt strukturgebend für das Projekt, indem sie die Priorisierung der eigentlichen Aufgaben und die Durchführung genau trennt. Kanban Boards können nur für eine Person erstellt werden oder um ein gesamtes Team zu synchronisieren. Dies wird anhand von vier Prinzipien veranlasst [BLM+18], [Hen10]:

- *Visualisierung*: Die Visualisierung der aktuellen Arbeitsabläufe steht in erster linie im Vordergrund. Dazu werden alle wichtigen Prozessschritte (wie z.B. TO-DO, In Progress, Deployment, Test und Release) und Verantwortliche für alle visuell kenntlich gemacht. Dies kann mit Hilfe von Softwaretools für das Projektmanagement oder ganz klassisch mit einem sogenannten *Kanban-Board* umgesetzt werden.
- *Das Pull-Prinzip*: Im Pull-Prinzip wird festgelegt, dass sich die Entwickler aus der Menge der anstehenden Aufgaben welche herausziehen, sobald sie wieder die Kapazitäten dafür haben. Außerdem sollen Aufgaben Prozessgerecht erledigt werden und niemals in die nächste Prozessiteration geschoben werden.
- *Work-in-Progress Limitierung*: Hier wird darauf geachtet, dass die Menge an Aufgaben die gleichzeitig ausgeführt werden hart und exakt limitiert wird [Bor18] Die Limitierung wird im Board kenntlich gemacht, indem für die aktuelle Ausführung z.B. nur 3 Aufgaben gleichzeitig bearbeitet werden dürfen.

#### SCRUM

SCRUM ist eine sehr beliebte Projektmanagement- und Entwicklungsmethodik in der Softwareentwicklung, sodass sich dieses Verfahren weit verbreitet hat und sich als Standardvorgehen eingesetzt wird [Bor18]. Im SCRUM *Product Backlog*, werden alle Anforderungen aufgenommen,

die im gesamten Produktentstehungsprozess anfallen. Im *Product Backlog* werden sog. Userstorys aufgenommen. Eine Userstory ist eine detaillierte Beschreibung einer Anforderung, die je nach Komplexität weiter in Teilanforderungen heruntergebrochen werden kann. Die Teilaufgaben werden i.d.R. sequentiell abgearbeitet. Eine Userstory ist demnach komplett abgeschlossen, wenn auch alle ihre Teilaufgaben abgeschlossen sind. Anforderungen die erst nach Projektstart formuliert oder modifiziert werden können auch im *Product Backlog* aufgenommen werden. Ein stocken im Entwicklungsvorgang wird für jeden ersichtlich gemacht, sodass man gemeinsam an eine Lösung arbeiten kann. Die Iterationsphasen, also da wo das Produkt um Funktionalität erweitert wird, wird als Sprintphase bezeichnet. Jede Sprintphase wird in einem Sprintmeeting geplant. Hier werden alle relevanten Anforderungen für den anstehenden Sprint mit dem Kunden und den Entwicklern besprochen und im Sprintbacklog aufgenommen. So kann aus dem großen Pool an Anforderungen eine Menge priorisierter Anfoderungen ausgewählt werden und innerhalb des Sprints, der i.d.R. 2-3 Wochen dauert, vom *Projekt Team* umgesetzt werden. Die Sprintphase von 2-3 Wochen wird von einem Daily Stand-Up Meeting begleitet. Täglich treffen sich alle Scrum Rollen (Product Owner: Verantwortlich für das Produkt, SCRUM Master: das Pendant zum Projektmanager in klassischen Projekten und das Entwicklungsteam: die umsetzende Einheit), um den aktuellen Status zu besprechen die zu erreichten Tagesziele und die möglichen Herausforderungen die aufgekommen sind. Die bis hierhin vorgestellen Methoden bestimmen die Prozessstruktur für das Projekt und sind gleichzeitig auch Vorgehensweisen in der Entwicklung.

### 2.5.5. Hochskalierbare Agile Methoden

Da agile Vorgehensmodelle sehr beliebt sind in der Softwareentwicklung und Projektorganisation versucht man die Grundprinzipien auf Unternehmensebene zu skalieren. Ein heutzutage bekanntes Beispiel dafür ist unter der Bezeichnung SAF bekannt. Eine schematische Darstellung ist in Abbildung 2.10 zu sehen.

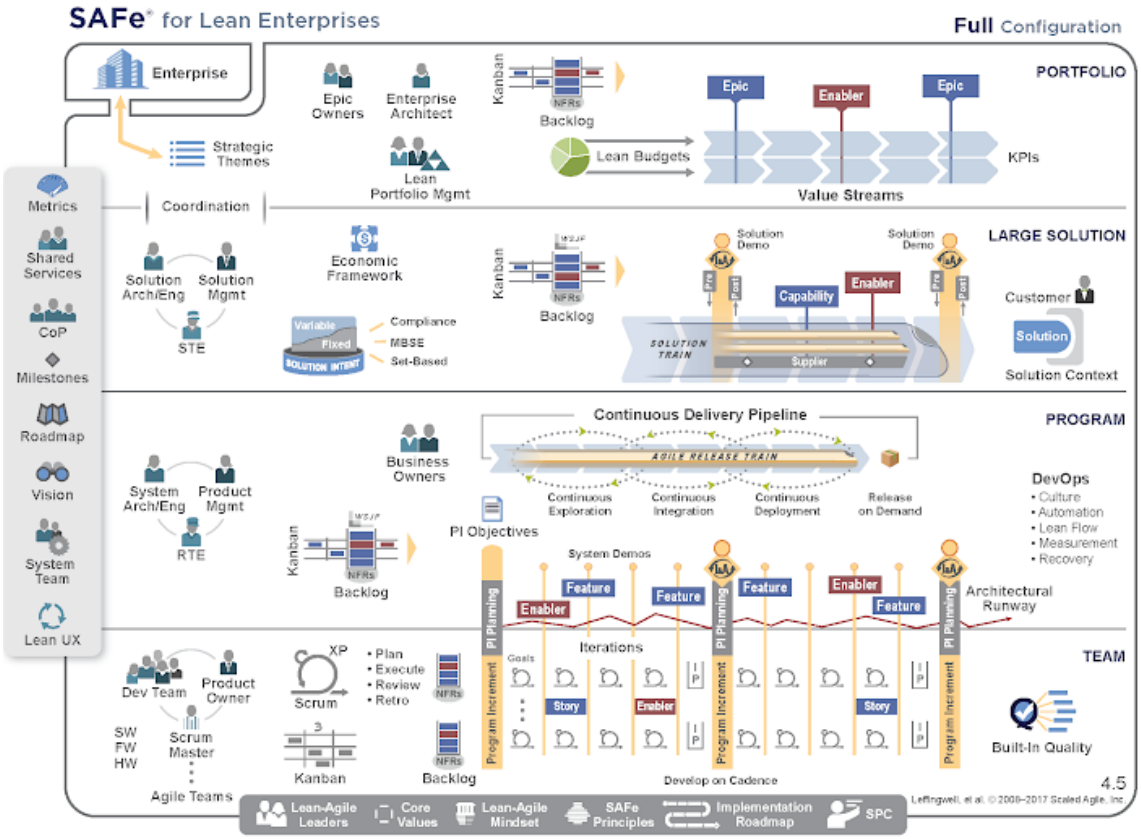


**Abbildung 2.10.:** SAF-Framework [Agi]

Das Vorgehen in der Abbildung ist eine Top-Down Ansicht. Auf *Portfolio*-Ebene, das Pendant zur Unternehmensebene, entsteht die Vision des Produktes und die Frage, welchen Mehrwert man an den Kunden liefern möchte. So entstehen auf oberster Ebene die Epics und Visionen für das Produkt. So können, wie beim klassischen Projektmanagement, Anforderungen definiert und die Komplexität durch das definierten von Teilanforderungen reduziert werden. So können die verschiedenen Stakeholder auf allen Fachbereichen und die Lieferanten zeitnah involviert werden und das Produkt kann schrittweise Form annehmen. Die Anforderungen wandern dann von oben nach unten (Top-Down) bis man auf Teamebene gelangt, dort wo die Entwicklung der Anforderungen stattfindet. Diese Ebene ist vom SCRUM-Vorgehen bekannt. Die Sprints dauern i.d.R. auf Teamebene wie bei SCRUM ca. 2-3 Wochen. Die Teilergebnisse der SE-Teams werden dann in einem *Programm Increment (PI)* zusammengefasst. Das *PI* dauert üblicherweise 2 Monate. Auf diese weise kann eine Taktung für die Synchronisation der unterliegenden Teams einführt werden. In einem *PI* auf *Program*-Ebene finden Meetings mit anderen Projekten/Schnittstellen/Partnern statt. Dort plant man dann die nächste *PI*-Phase mit den relevanten Zielen die an die Projektteams nach unten weitergereicht werden. Die Komplexität der Anforderungen wird dann wieder runtergebrochen und nach der festgelegten Taktung (X-Monate), werden die Ergebnisse integriert und es findet erneut eine *PI*-Planung statt. Werkzeuge, um den Fortschritt anzuzeigen sind sogenannte *System Demos*. Dort stellt jedes Team seine erreichten Ziele aus der *PI*-Phase in den Gruppen-Meetings vor [Agi].

# 3. Verwandte Arbeiten

## 3.1. Verwandte Studien zur Releaseprozessysthematik

Sax et al. [SRGK17] haben eine Studie im automotive Umfeld durchgeführt, um den aktuellen Stand der Praxis und die vorherrschenden Herausforderungen bei der Verwaltung von Softwarereleases und insbesondere die zukünftige Wichtigkeit von Software-Over-The-Air-Updates. Das Ergebnis ergab, dass Aktualisierungen bei ausgelieferten Fahrzeugen, aufgrund des hohen Softwareanteils eine immer wichtigere Komponente in zukünftigen Fahrzeugen wird die über drahtlose Kommunikation realisiert werden kann. Außerdem beschreiben sie die Tendenz hin zu kürzeren Release- und Update-Zyklen sowie diverse Herausforderungen in Bezug auf die Vereinigung multidisziplinärer Teams in der Entwicklung. Für Bestfleisch, Herbst und Reichert [BHR05] ist Workflow-Unterstützung ein wichtiger Schlüsselfaktor, um die Komplexität des E/E-Bereichs eines Automobils zu beherrschen. Sie definieren damit Anforderungen die zur Steuerung und Monitorierung von abhängigen Releaseprozessen. Müller et al. [MHHR06] definieren Anforderungen zur Verbesserung des Release-Managements über die IT-Unterstützung von Automobilentwicklungsprozessen. Lindgren et al. [LLNW08] entdecken, durch die Untersuchung von Software- und Systementwicklungsprojekten, fundamentale Aspekte der Releaseplanung. Außerdem erfassen sie den Stand der Technik für die Release-Planung. Dabei wird ersichtlich, dass nur Arbeiten aus einem Automobilen Kontext die Releaseplanug in der Systemtechnik behandeln. Danesh et al. [DAST12] untersuchen die unterschiedlichen Ansätze der Unternehmen für die Planung von neuen Software-Releases. Sie betonen, dass es zu diesem Zeitpunkt noch keinen einheitlichen Planungsansatz für den Release gibt. Heikkilä et al. [HPLE13] stellen fest, dass die Scrum-Entwicklung in hoch skalierten Enwticklungsumgebungen einen speziellen Release-Planungsprozess erfordert. Dieser soll die agile Arbeitsweise und Planung unterstützen. Darauf basierend wurde eine Fallstudie vorgestellt, die beschreibt wie eine solche Planung die Releaseplanung durchgeführt wird und welche Herausforderungen und Vorteile damit verbunden sind. Heikkilä et al. [HRJ10] beschreiben die Qualifizierungsphase de in einem Verbundrelease. Daraus resultiert eine Fallstudie zur Erstellung einer Release-Planung im agilen Umfeld für mehrere Teams. Somit liegt der Fokus der verfügbaren Arbeiten ausschließlich auf die agile oder klassische Releaseplanung. Keine verwande Arbeit wurde gefunden, die die Koexistenz von klassischen und agilen Projektstrukturen in der Release-Planund berücksichtingen und die Herausforderungen bei der Synchronisation dieser kennen. Diese Berichten oft nur über die Schnittstellenprobleme von agiler Entwicklung in einem klassischen Umfeld (siehe HELENA-Studie [KHF+17], [KDM+19]). In der systematischen Literaturstudie die von Karvonen et al. [KBOK17] durchgeführt wurde, findet man Praktiken des Agile Release Engineering. Ameller et al. [AFFR16] fassen in ihrere Literaur alle Modelle der Planung von Softwareversionen zusammen.

# 4. Methodisches Vorgehen

Ziel dieser Arbeit ist es die Herausforderungen in der Releaseplanung im Automtiv Umfeld zu untersuchen. Dazu soll ein qualitatives Forschungsverfahren angewendet werden, die *GTM*, deren Hauptmerkmale nachholend erklärt werden.

## 4.1. GTM

Die GTM eignet sich als Datenauswertungsmechanismus, da die Datenerhebung aus einem kontinuierlichen Prozess von Expertensuche und Datenauswertung besteht. Nach diesen Grundzügen arbeitet auch die GTM, denn die Datenauswertung geschieht nicht am Ende einer abgeschlossenen Datenerhebungsphase. Die Datensammlung und Datenanalyse wird in einem endlichen Kreislauf ausgeführt, bis im optimalen Fall die sogenannte *theoretische Sättigung* erreicht wird. Die *theoretische Sättigung* tritt ein, wenn keine neuen Erkenntnisse mehr aus den Daten gewonnen werden.

Das von Barnery G. Glaser und Anselm L. Strauss entwickelte Analyseverfahren aus dem Jahre 1967, hat sich als standardempirische Forschungsmethode durchgesetzt. Die GTM ist flexibel in ihrer Durchführung, doch verfolgt sie einige grundlegende Prinzipien. Ziel dieses Verfahrens ist die Theoriegenerierung aus empirisch erhobenen Daten. So werden basierend auf ein Untersuchungsgebiet die releavanten Bereiche und Kernaussagen nach und nach im Forschungsprozess generiert. Der Untersuchungsbereich in dieser Arbeit ist die *Releaseplanungssysthematik im Automotiv Umfeld*.

Die kontinuierlichen und sich auswechselnden Phasen aus Datenerhebung und anschließende -Auswertung bzw. -Analyse generieren neue Theorien aus den erhobenen Daten. Daher ist in der GTM ein zyklischer Ablauf des Forschungsprozesses vorgesehen [MM11b], [MM11a].

Aufgrund der strukturierten Durchführung durch das “Paradigma Modell”, wird in dieser Arbeit die GTM nach Strauss verwendet. Die GTM nach Strauss sieht drei Stufen zur Theoriegewinnung vor die nachfolgend aufgelistet werden.

1. Offenes Kodieren
2. Axiales Kodieren
3. Selektives Kodieren

Unter Kodierung versteht man das versehen von Textstellen mit geeigneten, oft auch zusammenfassenden Überschriften []. Beim offenen Kodieren, wird zu Beginn der Text Zeile für Zeile, satzweise, paragraphweise oder über ein gesamtes Dokument kodiert. So verschafft man sich einen generellen Überblick der Daten. In diesem Schritt werden auch die ersten *Kategorien* gebildet, die im weiteren Verlauf der Datenerhebung weiter verfeinert werden. Eine Kategorie ist eine Kernaussage, die

sich aus einem wiederholten Datenerhebungs- und Analyseprozess über mehrere Datensamplings herauskristallisiert hat [MM11a]. Bei der axialen Kodierung werden die gefundenen Kodings mit weiteren Kodings aggregiert, die sich in ihrer Aussagekraft sehr ähneln. Dazu wird das “Paradigma Model” verwendet. Das “Paradigm Modell” ist ein Instrument für die Kategoriebildung. Dieses stellt ein sogenanntes Phänomen im Mittelpunkt, das aus den ursächlichen Bedingungen, die sich daraus ergebenden Folgen und die angewendeten Maßnahmen aus den Daten herausfiltert. Zum Schluss wird durch das selektive Kodieren die Hauptkategorie bestimmt, die alle anderen Kategorien miteinander verbindet.

## 4.2. Coding

Über geeignete Kodings werden Textstellen mehrerer Interviewteilnehmer verglichen und ausgewertet. Die Codings werden zu Beginn frei gewählt, um sich einen Überblick der Themengebiete, die besprochen werden, zu verschaffen. Der Fokus liegt dabei auf Problematiken zu stoßen, die im Zuge einer optimalen Integration aller Komponenten in einem Verbundreleaseprozess vorherrschen.

## 4.3. Studiendesign und Methodik

Die durchgeführte Studie in dieser Arbeit, ist eine Wiederholung einer bereits bei der Dr. Ing. h. c. F. Porsche AG durchgeführten Studie. Beide Studien haben das Ziel die Herausforderungen in der Releaseplanungsysthematik von Unternehmen im automotiv oder automotiv ähnlichen Umfeld zu analysieren.

### 4.3.1. Fragebogen

Das Instrument zur Datensammlung ist ein Fragebogen bestehend aus 35 Fragen. Der Fragebogen im Originalformat beinhaltet offene und geschlosse Fragen die in sechs Kategorien strukturiert sind. Um ein offenes Interview zu führen, wurden die geschlossenen Fragen, ohne den Inhalt zu verändern, zu offenen Fragen umstrukturiert (siehe A).

Nachfolgen sind die Kategorien aufgelistet:

- Context
- Prozesssystematik - Verbundrelease
- Prozesssystematik - Planung
- Prozesssystematik - Integration
- Entwicklungsmethodik - Abstimmung
- Entwicklungsmethodik - Testen

In der erste Kategorie, die *Context*-Kategorie, wurden Fragen gestellt, um den Befragten in seinem Arbeitsumfeld genau einordnen zu können. Beispielsweise werden Fragen zur ausgeübten Rolle und zur Dauer der Tätigkeit des Befragten gestellt, so wie die Kategorisierung des Bereichs in welchem er tätig ist. Außerdem wird die verwendete Projekt- bzw. Entwicklungsmethodik abgefragt (traditionell, agil oder hybrid). In der zweiten Kategorie, die *Verbundreleasephase*, werden Fragen zum Verbundrelese gestellt. Dazu gehört beispielsweise die aktuelle Anzahl an Meilensteine sowie die Dauer. In der dritte Kategorie, die *Planungsphase*, wird die Notwendigkeit und Sinnhaftigkeit einer initialen Aufplanung der Inhalte für einen Meilenstein abgefragt. In der vierten Kategorie, die *Integrationsphase*, wird die Auswirkung der Bugfixinphase auf den eigentlichen Entwicklungsprozess abgefragt, ebenso welche Aktivitäten den Arbeitsalltag während eines Verbundreleases dominieren etc. In der fünften Kategorie, die *Koordinationsphase*, wird die Qualität der Prozesstransparenz abgefragt. In der letzten Kategorie, die *Testphase*, werden alle Fragen rund um die Testumfänge in einem Meilenstein abgefragt.

### 4.3.2. Methode

Es fehlt derzeit aber noch ein Überblick über die Herausforgerungen in der Releaseplanung im automobilen Umfeld. Um diese Lücke zu schließen, wurde 2018 eine entsprechende Studie bei der Dr. Ing. h. c. F. Porsche AG aufgesetzt. Teile davon werden in dieser Arbeit vorgestellt. Im Mittelpunkt der Studie steht die Verbundreleasesystematik, die eine inkrementelle Entwicklung eines Fahrzeugs veranlasst. Um die Herausforderungen zu identifizieren wurde die Studie mit Experten aus der automotive und automotive ähnlichen Branchen durchgeführt, die direkte oder indirekte Berührungspunkte mit dem Prozess haben. Die Befragung erfolgte in einem offenen Gespräch mit den Experten, das ca. ein bis zwei Stunden durchschnittlich in Anspruch genommen hat. Alle 35 Fragen aus dem Fragebogen wurden gestellt, sodass zusätzlich eine perspektivische Analyse durchgeführt werden konnte. Insgesamt konnten nur fünf Fragen nicht ausreichend beantwortet werden.

## 4.4. Samplingstrategie

Für die Interviews wurden Experten mit einer Projektmanagementrolle oder aus den Entwicklungsabteilungen ausgewählt. Den Schwerpunkt bildeten dabei Vertreter der Original Equipment Manufacturer (OEM), sowie von Unternehmen die in einer verwandten Branche tätig sind und dadurch bedingt ähnliche Prozesse adaptieren und die Problematiken kennen. Kontakte wurden zum einen direkt aus der Universität Stuttgart bereitgestellt und zum anderen aus dem persönlichen, indirekt erweiterten Netzwerk gewonnen.

Repräsentativität konnte dadurch nicht erreicht werden, aber die Zusammensetzung stellt den aktuellen Stand in der Automobilen Releaseplanung doch in guter Art und Weise dar und bildet damit eine Grundlage für weitere empirische Forschungen in diesem Gebiet, sowie die gezielte Durchführung von Prozessanalysen um die

Das Sample enthielt (IT-)Projektleiter, Softwareentwickler und Steuergeräteentwickler. Die Experteninterviews waren standardisiert und wurden im offenen Gespräch durchgeführt. Sofern keine wichtigen Gründe dagegen sprachen, wurden die Antworten der Gesprächspartner mit der Recorder Applikation eines Smartphones aufgezeichnet und anschließend von Hand transkribiert.

# 5. Auswertung und Vergleich zu den Porscheergebnissen

In diesem Kapitel werden die Ergebnisse der Expertenbefragung vorgestellt. Die Ergebnisse werden in der Reihenfolge der jeweiligen Kategorien im Fragebogen vorgestellt. Außerdem findet ein unmittelbarer Vergleich zu den jeweiligen Fragen mit den Ergebnissen aus Porschestudie.

## 5.1. Context

Insgesamt wurden Experten von sechs Unternehmen aus dem automotive und automotive ähnlichen Bereich befragt, mit einer Gesamtzahl von 12 Interviewpartner. Die durchschnittliche Berufserfahrung lag bei 4,5 Jahren, mit einem Minimum von einem halben Jahr und einem Maximum von 15 Jahren (F2). Desweiteren wurden die Teilnehmer(T) anhand ihrer Rolle unterschieden, die sie im Unternehmen ausüben (F1). Folgende Rollen sind im Sampling vorzufinden:

| | |
|---|---|
| T1: | Kundensystem-Ingenieur bei einem Automobilhersteller |
| T2: | IT-Projektleiter und Scrum Master für Automotive Softwareprojekte |
| T3: | Support und Qualitätssicherung in der Integrationsphase |
| T4: | Manager eines SAP-Logistik Templates bei einem Automobilhersteller |
| T5: | Produkt Owner für Applikationen bei einem Automobilhersteller |
| T6: | Software Ingenieur |
| T7: | Software Ingenieur und funktionaler Sicherheitsbeauftragter |
| T8: | Projektingenieur in der Forschung und Entwicklung einer automobilverwandten Branche |
| T9: | Basissoftwareentwickler im Kommunikationsstack für Steuergeräte nach Autosar bei einem Automobilhersteller |
| T10: | SE-Teamleiter/Fachabteilungsleiter in einer automobilverwandten Branche |
| T11: | SE-Teamleiter/Fachabteilungsleiter in einer automobilverwandten Branche |
| T12: | Verbundrelease Planer in einer automobilverwandten Branche |

**Tabelle 5.1.:** Rollen der befragten Experten

Somit sind sechs Teilnehmer (n=6; 50%) in Management- und Projektplanungspositionen involviert. Ein Teilnehmer (n=1; 8,3%) ist in der meilensteinspezifischen Qualitätssicherung involviert. Ein Teilnehmer (n=1; 8,3%) ist Produktverantwortlicher bei einem Automobilhersteller. Ein Teilnehmer (n=1; 8,3%) ist in der Verwaltung eines SAP-Logistik Template involviert. Drei Teilnehmer (n=3; 25%) sind in Entwicklungstätigkeiten involviert und somit Stellvertretend für die *operative Ebene*.

Desweiteren wurde die genaue Projekttätigkeit der Experten erhoben, die nachfolgend vorgestellt wird (F3). Ein Experte (n=1) ist als Kundensystemingenieur, die Schnittstelle zwischen Kunde und Entwicklung und verantwortet die Entwicklung von E/E- sowie Softwarekomponenenten. Ein Experte (n=1) ist für die Security der Softwarelogistik zuständig, also die Verwaltung der Softwarestände,

die im Fahrzeug eingespielt werden und somit auch in der Backend Vernetzung des Automobils tätig. Ein weiterer Experte (n=1) arbeitet an der Qualitätsicherung der Abgabeleistungen. Ein weiterer Experte (n=1) arbeitet im Logistikbereich an einem Template, um alle Aktivitäten aufzufassen für die Verwaltung von Lagerbeständen und der Produktion. Die Aussagen dieses Experten können nur bedingt in der Analyse mitberücksichtigt werden, da die Berührpunkte zu den Fahrzeugmeilensteinen relativ gering sind. Ein Experte (n=1) verantwortet die Implementierung von verschiedenen Applikationen für die Optimierung und Unterstützung der Prozesse von Logistikunternehmen. Ein Experte (n=1) arbeitet an der System-Entwicklung von Automatisierungssystemen. Drei Experten (n=3) arbeiten in einem Steuergeräteprojekt als Funktionsentwickler. Zwei Experten (n=3) arbeiten in einer verwandten Branche jeweils an der Planung und Verwaltung des Gesamtsystem.

Für eine genauere Kategorisierung wurde der aktuelle Projektbereich der Experten abgefragt. Dies umfasst folgende Bereiche (F4):

- Motorelektronik (n=1; 8,3%)
- Sicherheitsrelevant (n=1; 8,3%)
- IT-Logistik (n=2; 16,6%)
- Applikationsentwicklung/Softwareentwicklung (n=4; 33,3%)
- E/E-Bauteile (n=3; 25%)
- Kabelbaum und Steuergeräteproduktion (n=1; 8,3%)

Ein Teil der Befragten (n=1) verwendet als Management- bzw. Entwicklungsmethodik das SAF-Framework **(F5)**. Weitere Befragte (n=2) gaben an, SAF und SCURM für größere Projekte zu verwenden als Managementmethodik, sonst klassisch nach Wasserfall zu arbeiten. Desweiteren verwenden einige Befragten (n=5), ein Teil davon aus der automotive ähnlichen Branche (n=3), ein Mix aus V-Modell und agilen Methoden. Oft ist es aber so, dass obwohl sie angeben agil zu arbeiten, in der Realität klassische bzw. traditionelle Entwicklungsmethodiken nach V-Modell dominant auf den Prozessverlauf einwirken. Ein weiterer (n=1) gab an klassich nach V-Modell zu arbeiten. Ein Experte (n=1) setzt für für die Projektorganisation und Entwicklung Kanban und vereinfachtes SCRUM ein. Ein weiterer Experte gab an reines SCRUM zu verwenden (n=1).

## 5.2. Prozesssystematik: Verbundrelease

Die Experten sollten außerdem die aktuelle Anzahl an Verbundreleases über die gesamte Projektlaufzeit beurteilen **(F6)**. Aus Managementsicht wird die Anzahl an Verbundreleases als zu hoch empfunden (n=5; 41,7%). Ein Experte (n=1; 8,3%) mit Managementbackground gab an, dass die Anzahl per se nicht so hoch ist, aber der bürokratische Aufwand für die Vorbereitung der Meilensteine zu hoch ist. Aus Entwicklersicht (n=4; 33,3%) ist die Anzahl deutlich zu kurz, sodass oft etwas nachgeliefert werden muss. Zwei Experten konnten keine genaue Aussage treffen.

Desweiteren wurde gefragt, wie sie die Dauer des Verbundreleases empfinden **(F7)**. Den Verwantwortlichen für die Leitung der Projekte (n=3) ist er zu lang, denn die Produktzyklen werden immer kürzer, so muss das Produkt auch schneller Marktreif sein. Softwareentwickler (n=4) finden die Dauer der Releasezyklen nicht immer optimal, aufgrund der Tatsache, dass ihre Entwicklung stark

von den Steuergeräten Abhängt, wo sie später geflashed werden. Wenn Abhängigkeiten bestehen werden oft mehr Meilensteine benötigt als vorgegeben. Einige Entwickler (n=2) haben teilweise den Überblick verloren, auch Aufgrund von Nachlieferungen, die einer standardisierten Vorgehensweise entgegen wirken.

Ein Experte gab an (n=1), dass alle zwei Monate abgabereife Software-Lieferleistungen erbracht werden können **(F8)**. Zwei weitere Experten behaupten (n=2), dass sie realistisch alle drei Monate releasen können, aber die Tendenz Richtung zwei Monate geht. Im Logistik Bereich finden die Releases maximal zwei Mal im Jahr statt (n=2). Ein weiterer Experte behauptet (n=1), dass sinnvolle Releases alle 4 Monate erbracht werden können, aber die Abgabereife sei projektabhängig, sodass auch kürzere Zyklen erreicht werden können. In den Automobilverwandten Branchen wird ein Mal im Jahr released (n=3). Dies ist durch die hohe Komplexität des Produktes determiniert. Drei Enwickler sagen (n=3), dass es von der zu entwickelnden Software abhängt. Dabei bewegt sich die Dauer der Releasezyklen zwischen zwei Wochen (Applikationssoftware) und zwei bis sechs Monate (Basissoftware und Bootloader).

Zudem wurde gefragt, ob die Experten rechtzeitig Rückmeldung aus dem Verbundrelease erhalten **(F9)**. Zwei Experten (n=2; 16,6%) konnten keine genauen Angaben dazu machen, da sie zu wenig im Verbundrelease eingebunden sind. Drei Experten (n=3; 25%) meinen, da vieles nicht vollautomatisiert getestet werden kann, erhält man oft nur Feedback, wenn der Kunde direkte Rückmeldung gibt. Aus Managementsicht (n=5; 33,3%) wird rechtzeitig Rückmeldung gegeben. In der automotive verwandten Branche bestehen keine Probleme bzgl. der rechtzeitigen Rückmeldung aus dem Verbundrelease (n=2; 25%).

In den Fahrzeug Verbundreleases ist es oft schwer zu wissen, wer die Testingenieure sind, welche Softwareversion aktuell auf den Steuergeräten läuft oder Allgemein welche Tests bereits durchgeführt wurden. So verliert man schnell die Übersicht, weil das Testhaus und die Entwicklung oft nicht in derselben Abteilung sitzen. Aus Entwicklerseite wird angeführt, dass es schwer ist die Testfahrzeuge oder Geräte in der Kombination zu bekommen, wie sie später im Fahrzeug vorzufinden sind. So kann man bestimmte Testfälle erst spät im Prozess bzw. kurz vor SOP testen. Da eingie Komponentenreleasezyklen zu lange dauern, sind oft nur lokale Tests möglich. Dies ist aber nicht ausreichend, um alle Testfälle in der realen Umgebung vollständig abzudecken. Die Feedbackzyklen dauern dementsprechend auch länger, wenn die Testgeräte nicht verfügbar sind. In Infotainment Systemen werden oft nur qualitative Tests durchgeführt und die Menge an Tests, die sich über eine lange Zeit erstrecken ist oft zu gering, sodass das Feedback zum einen sehr spät oder teilweise gar nicht zurückgegeben wird. Ein Mangel an Testingenieuren führt dazu, dass nicht alle Testfälle abgedeckt werden können. Der vollumfängliche Test kann somit erst später beim Kunden stattfinden. Entwickler fühlen sich nicht vollständig in diesen Feedbackprozess involviert. Sie kapseln sich vom Prozess an und arbeiten dediziert an ihren Aufgaben weiter.

Wenn die Testumgebung und Enwicklung im selben Haus sitzen können sie teilweise dem Verbundrelease-Druck entkommen. So können sie den Reifegrad unabhängig vom Verbundrelease überprüfen. Eine Transparenz zu den direkten Lieferanten ist sehr wichtig, um Prozessverzögerungen zu vermeiden, zumal die Entwicklung oft direkt an ihnen gekoppelt ist. Zwei Experten konnten keine Aussage treffen, da das Feedback direkt an den Software Projektleiter berichtet wird und dies unter Umständen verzögert und priorisiert an die jeweiligen Entwicklungsabteilungen weitergeleitet wird. Die Entwickler nutzen die Testphase, um die Entwicklung und das lokale Testing voranzutreiben. So werden meitens, bis das Feedback vom Fehlermanagement zurückgegeben wird, die Fehler bereits gefunden und behoben.

Laut drei Experten sollte ein Verbundrelease alle 3 Monate stattfinden, damit man immer einen abgabereifen Softwarestand liefern kann (n=3) **(F10)**. Zwei Experten konnten aufgrund ihres Backgorunds keine genauen Aussagen treffen (n=2). Aus Entwicklersicht und in kombination mit agilen Methoden wie SCRUM (n=4), spielt die Taktung keine große Rolle, aufgrund der flexiblen Gestalltungsmöglichkeit der EPICs. Desweiteren wäre ein Verbundrelease zum Ende eines jeden Monats ideal aus Entwickler- und Managementsicht (n=3), sodass man während der Entwicklungsphase nicht zu viele Abhängigkeiten produziert und der Testumfang nicht so ausgedehnt ist wie nach 3 Monaten.

Zusätzlich sollte man darauf achten, dass man die Taktung nicht zu klein wählt, damit die Entwickler die Zeit haben die Fehler zu beheben und nicht nach kurzer Zeit wieder den nächsten Verbundrelease vorbereiten. Nur so kann eine stabilde Systemversion generiert werden. Oft werden Tests durchgeführt die dann aus Zeitmangel nicht überprüft werden können. Die Softwareseite ist immer etwas flexibler und man kann jeden Prozess adaptieren. Wenn ein Feature die gewünschte Reife nicht erreicht hat, wird es ins nächste Release geschoben. Dafür können aber die fertigen Features ausgerollt werden, sofern keine Abhängigkeiten bestehen. Dennoch bleibt die Hardware auf Fahrzeugseite tonangebend, sodass Features meistens priorisiert bearbeitet werden, um keine gravierenden Verzögerungen im Prozessverlauf und keine schwerwiegenden Featureverluste in den Releasephasen zu verursachen. Die Steuergrätesoftware sollte demnach so konzipiert sein, dass man zu jedem Zeitpunkt ein releasefähiges Inkrement produziert. Dadurch wird man auch schneller in der Entwicklung und die Fehlersuche kann frühzeitig und iterativ über den gesamten Projektverlauf ausgeführt werden. Eine weitere Entwickler- und Managementsicht betont die Wichtigkeit der Regelmäßigkeit. Durch Nachintegrationen verschieben sich die Releasephasen, sodass der Prozess nicht standardisiert abgearbeitet werden kann. So können Abhängigkeiten nicht getestet werden, weil aufgrund unterschiedlicher Releasezyklen keine gemeinsame Testphase stattfinden kann. Der vollumfängliche Test muss aber immer in Kombination erfolgen. Das Problem ist somit nicht die Reife der Abgegebenen Leistung, sondern die fehlende Testumgebung für die fertige Softwarekomponenten. Drucksituationen mit mehr Druck bzgl. der Entwicklungsreife auszugleichen ist kontraproduktiv aus Entwicklersicht . Man sollte die Software von der Hardware so unabhängig wie möglich gestalten, sodass sich die Hardwareseite aus dem Pool an feritgen Softwareentwicklungsständen selber bedienen kann. Ein gutes, persönliches Verhältnis zu den Lieferanten und den direkten Schnittstellen aufzubauen ist sehr hilfreich. So können die Sichtweisen und Schwierigkeiten besser nachvollzogen werden.

Aus Management- und Entwicklungersicht werden zusätzliche Verbundreleases in Form von Teilverbünden mit reduziertem Testumfang zur Absicherung abhängiger Steuergeräte bzw. Komponenten als hilfreich angesehen (n=5) **(F11)**. Einige Experten behaupten (n=3), dass das nicht so hinfreich wäre. Weitere Experten (n=4) sehen das als sinnvoll.

Das herunterbrechen der Komplexität des Verbundreleases hilft das Problem besser zu beherrschen, indem die Testumfänge gleichzeitig reduziert werden. Wenn die Komplexität im nächsten Release zu hoch ist besteht die Gefahr, dass die Projektbeteiligten das Problem nicht verstehen, oder vielleicht nicht alle Aspekte berücksichtigen. Auf Steuergeräteseite gestaltet sich dies evtl. schwieriger, da man eben die starke Abhängigkeit mit der Software hat und die Bauteilebene teilweise schwieriger herunterzubrechen ist. So dauert dieser Prozess unter Umständen länger, aber die Komplexität wird reduziert und die Qualität wird verbessert. Ebenso aus Softwaresicht reduziert die Entwicklungsdauer von einem Monat den Testumfang. Der Testanspruch ist sehr hoch, sodass ein geringerer Umfang oft viel einfacher zu beherrschen ist und die Releases somit besser abgesichert werden

können. Abhängigkeiten sollte weitgehends automatisiert testen und klare Schnittstellen definieren. Schließlich sollten bilateral Integrationstests geschreiben und ausgeführt werden. Alle Inkonsistenzen zwischen den Schnittstellensystemen werden im Großteil der Fälle in diesem finalen Schritt aufgedeckt. Bei kleineren Releasezyklen steigt die Testeffizienz, da man eben nur Teilabschnitte testet (n=6). Auf operativer Ebene könnten kleinere Releasezyklen gebildet werden, um frühzeitig den eigenen Stand und integrativ testen zu können. So kann ein iterativer Aufbau der Komponente stattfinden und auch die Testphase im Verbundrelease reduziert werden. Der Durchblick wird besser wenn man eben Schrittweise entwickelt und möglichst hoch automatisiert testet. Oft ist das Problem, dass die Entwicklung ein bis drei Monate dauert und dafür Ressourcen gebunden werden. Wenn Fehler auftreten sind die Entwickler im Zweifel nicht mehr da oder die externe Firma ist nicht mehr beauftragt. Jahreswechsel spielen auch eine große Rolle, da dort alle neu beauftragt werden müssen. Wenn die Entwicklung z.B. drei Monate dauert und das Produkt drei in einer anderen Abteilung getestet wird fehlt durch mangelnde Kommunikation die Übersicht der durchgeführten Tests aus Entwicklersicht. Ebenso wissen die Testingenieure nicht wie eine Funktion oder ein Bauteil entwickelt wurde. Die intrinsiche Motivation seitens der Testingenieure Fehler zu beheben fehlt, weil sie sich eben nur auf ihre Testing Aufgaben beschränken. Manche Tests im Verbundrelease sind viel zu aufwändig und oft auch nicht sinnvoll. Für solche Fälle sind kleinere Prüfstände mit einem Teilverbund der wichtigsten Komponenten, inklusive diversen Simulationen sinnvoll.

Aus Softwareentwicklersicht ist ein gravierendes Problem die große Abhängigkeit zu Partnersteuergeräten. Wenn die unterschiedlichen Releasezyklen zu unterschiedlichen Fertigstellungsterminen führen, sodass beispielsweise fertige und gut getestete Software nicht vollumfänglich in Verbindung mit dem Steuergerät getestet werden kann, dann entstehen Verzögerungen im Prozess. Funkionalitäten die mit dem Steuergerät getestet werden müssen, können erst bei Fertigstellung des Steuergerätes durchgeführt werden. Dies führt dazu, dass Features erst in den nächsten Releasezyklen ausgeliefert werden können. Zusammenfassend bedeuet dies, dass unabhängige Steuergeräte durchaus eigene Releasezyklen haben können. Wenn aber große Abhängigkeiten zwischen Steuergeräten und bestimmten Funktionen bestehen, müssten zumindest gemeinsame Releasepläne erstellt werden.

## 5.3. Prozesssysthematik: Planung

Eine Initialaufplanung der Inhalte zu Projektstart, für die gesamte Laufzeit ist für einen Großteil der Befragten möglich jedoch nicht sinnvoll **(F12)** (n=8).

Die Initialaufplanung ist sinnvoll wenn die Bereichsabhängig durchgeführt wird, z.B. bei einer Werkplanung, oder Maschinenbeschaffung (n=2). Oft sind die Aufgaben zu Beginn noch nicht klar , sodass eine Initialaufplanung auch sinnvoll ist (n=2).

Eine Detailplanung ist jedoch in vielen Fällen aufgrund der sich ständig ändernden Anforderungen sinnlos und zeitaufwendig. So muss die Planung möglicherweise nach wenigen Monaten wieder aktualisiert werden, da sich permanente Verschiebungen ergeben. In der Fahrzeugentwicklung ist jedoch aufgrund der hohen Komplexität das Befolgen eines Plans oft einfacher, als auf Änderungen und Probleme zu reagieren. Eine grobe Planung sollte deshalb bevorzugt werden, um Richtlinien für die Synchronisation der operativen Ebene und gleichzeitig Freiräume zu schaffen. Dies ist oft nicht möglich, da die Systemanforderungen viel zu hoch sind und eine Detailplanung fordern. Anforderungsänderungen muss man aus Projektmanagement Sicht von vornherein großzügig im Budget einplanen, um flexibel mit Ressourcen darauf reagieren zu können. Dieses Budget kann man

auch pro Meilenstein definieren. Ein Fahrzeugprojekt dauert i.d.R. drei bis fünf Jahre und in dieser Zeit hat man möglicherweise neue Gesetze, sowie ganz andere Anforderungen oder das Management ändert sich. Mitarbeiterwechsel sind oft auch ein Problem, da man immer eine Einlernphase hat. Ein weiterer Experte behauptet, dass der Prozess einen detaillierten Plan erfordert, um überhaupt an den Verbundrelease teilnehmen zu können. Projekte detailliert zu planen, die von Grund auf neu aufgestellt werden müssen und die Anforderungen möglicherweise zu Beginn noch gar nicht klar definiert sind, ist fast unmöglich. Die Planung wird aber aus bürokratischen Gründen erstellt um Sicherheits- und Qualitätsstandards zu erfüllen. Die wirklichen Ziele entstehen dann während dem Projekt. Im agilen Kontext hingegen ist die Gesamtplanung nicht zwingend notwendig. Die Kenntnisse über das Ziel und die Kundensicht sind deutlich wichtiger für den agilen Prozess. Der jeweilige Bauteilverantwortliche priorisiert dann welche Funktionalitäten zu welchem Zeitpunkt Benötigt werden und leitet das an die Entwicklung weiter. Oft sind diese Andorderungen noch nicht detailliert, sodass die Planung sich zwangsläufig ändert, sobald die Wünsche des Kunden konkreter sind und die technische Seite detailliert wurde. Außerdem fehlt eine gemeinsame Planung mit den direkten Systemschnittstellen. In beiden automotiv verwandten Branchen wird die Anforderungserhebung über die Hardware- und Softwareanforderungen detailliert.

Bei der Frage, wie oft ist der Inhalt der initialen Planung zu Beginn eines Verbundreleases noch aktuell ist **(F13)** haben alle Experten (n=12) behauptet, dass der Plan nie aktuell ist.

Die Produktvision bzw. die übergeordnete Ebene ist überwiegend aktuell. Es kommen öfters im Jahr Umplanungen vor, sodass man beispielsweise den aktuellen Sprint ändern muss, obwohl die Planung bereits steht. In SAF bzw. agilen Umgebunden ist das oft problematischer, da die *Produkt Inkrements (PIs)* fest definiert sind und Änderungen einen großen Kommunikationsoverhead und das Befragen aller Beteiligten nach sich zieht, sodass die Methode zu einem Nachteil wird. Ein Großteil der Experten sehen als Problem, dass vieles nicht vorhersehbar und dementsprechend nicht planbar ist. Es werden zum Großteil Prognosen über die zukünftigen Anforderungen des Prozesses erstellt, die sich über einen Zeitraum von bis zu fünf Jahre erstrecken. In einigen Bereichen, wie im Infotainment-System, ist es IT technisch fast unmöglich abzuschätzen was in fünf Jahren aktuell ist. Aufgrund des Moorschen Gesetzes kann man die Prozessor- bzw. Hardwareentwicklung grob abschätzen. Dann muss man auf die Entwicklungszyklen der Hersteller warten, die das dann in 5 jahren umsetzen bzw. liefern könnten.

Ein Großteil der Experten aus dem Management- und Entwicklerbereich behauptet (n=8), dass man sehr darauf achtet gewisse Meilensteine im Projekt einzuhalten und eine einheitliche Releaseplanung erstellt und gelebt wird **(F14)**. Nichtsdestotrotz sollten die Meilensteine aus Entwicklersicht nur grob definiert werden, um den Prozess flexibel zu gestalten (n=4).

Vor allem die große Landschaft an Zulieferern wird immer wieder auf gleiche Meilensteine vereint. Die frühzeitige terminliche und fachliche Einbindung aller Lieferanten ist von großer Bedeutung für eine erfolgreiche Meilesteinstruktur. Dies ist dennoch Aufgrund der hohen Sicherheits- und Qualitätsanforderungen im automotive und automotive ähnlichen Kontext nicht möglich, laut allen Befragten (n=12). Fixe Deadlines werden auch gebrauch, weil viele Systemabhängigkeiten bestehen und bei Testphasen oft integrativ mit anderen Systemen getestet werden muss. Die Kommunikation ist bei der Planung und Aufgrund der hohen Anzahl an Schnittstellensysteme eine wichtige Komponenten. Dies ist oft schwierig, da man nicht die Zeit besitzt aktiv einen Fachübergreifenden/-Abteilungsübergreifenden Blick zu halten. Sie wird mit den heutigen (agilen) Vorgehensmodellen, viel mehr gelebt. Dadurch entsteht aber gleichzeitig ein Kommunikationsoverhead, das viel Zeit und Aufmerksamkeit in Anspruch nimmt. Aus Entwicklersicht wird die Planung oft nur der rechtlich

Zusicherung gesehen und um an den Verbundrelease teilnehmen zu dürfen. Die Releasepläne werden in einigen Fällen mit den Schnittstellensystemen zusammen aufgestellt. Man orientiert sich beispielsweise an den Releasezeitpunkt des Steuergerätes. So können Schnittstellensysteme Lieferzeitpunkte besser abgestimmt und rechtzeitig definiert werden. Dann können auch Softwarefunktionen, die nur in Abhängigkeit mit einem Steuergerät in den Verbundrelease gelangen, auch rechtzeitig fertig werden.

Plananungsinformationen zu erhalten, von den relevanten Ansprechpartnern ist relativ schwierig (n=6) **(F15)**. Weitere Experten (n=6) behaupten, dass die Transparenz zu ihren Ansprechpartnern gegeben ist und ein ständiger Informationsfuss stattfindet.

Das hängt in hohem Maße davon ab, wie kommunikationsfreudig diese sind und welchen Wissenstand die Befragten mitbringen. Die Abhängigkeiten im Prozess erschweren den Überblick, sodass die Prozessbeteiligten oft keine genauen Aussagen treffen bzw. Termine festlegen können. Im schlimmsten Fall hat die Komponente so viele Abhängigkeit, dass sie einen SOP verschiebt bzw. verhindert. Systemgestützte Projektüberwachungssysteme erleichtern den Überblick und begünstigen den Informationsfluss. Die Planungsmeetings werden nur auf Managementebene durchgeführt, so fehlt in den untersten Ebenen die nötige Transparenz und Einbindung der Projektmitwirkenden. In diesem Kontext sind agile Methoden aufwändiger, schaffen aber eine erhöhte Transparez der Prozesse und eine größere Kontaktfläche mit erhöhter Austauschmöglichkeit.

Weniger als 50% der Experten (n=5) behaupten, dass ihre Rolle und Aufgaben hinsichtlich des Planungsprozesses unklar bzw. nicht einheitlich dokumentiert sind (F16). Das ist aber stark abhängig von der Qualität der Organisation, von der Kritikalität des Projektes und wieviele Möglichkeiten/Ressourcen man hat. Die Dokumentation ist ein wichtiges Referenzdokument und wird durch entsprechende Normen sichergestellt. Viele Probleme kommen auch davon, dass man die Interaktion und Kommunikation nicht lebt.

Mit der Transformation der Arbeitswelt entstehen zudem neue Rollen und Rollenverständnisse, sowie die Tendenz Richtung Interdisziplinäre Teams, sodass die Rollenverteilung nicht immer für jeden eindeutig ist. Das richtige Mindset, eine hohe Selbstständigkeit und hohes Verantwortungsbewusstsein der Projektbeteiligten ist von enormer Wichtigkeit heutzutage. Aus Managementsicht ist häufig vieles nicht ausreichend Dokumentiert und dementsprächend fehlt auch die Transparenz in den Projekten. Aus Entwicklersicht ist es gut dokumentiert und für jeden im Team verständlich.

Für alle Befragten (n=12) haben Managemententscheidungen äußere Einflussfaktoren bzw. fremdbestimmte Entscheidungen einen starken Einfluss auf Ihren Entwicklungsverlauf **(F17)**. Die Herkunft der Entscheidung bestimmt die Priorität. So gibt es unternehmerische Entscheidungen, produktorientierte Entscheidung und kundenorientierte Entscheidungen. Produktorientierte Entscheidungen kann ein Entwickler innerhalb seines Arbeitssprektrums mitbestimmen und diese auch gezielt in seinen eigenen Zeitplan mit einbauen. Entscheidungen aus unternehmerischer oder kundenorientierter Sicht haben oft eine höhere Priorität und kommen für den Entwickler meistens überraschend, sodass sie nicht wirklich in die eigene Zeitplanung aufgenommen werden können. Diese werden meistens durch Puffer in der Zeitplanung abgedeckt aus Managementsicht. Zusätzlicher Arbeitsaufwand wird zudem dadurch generiert, dass Engpässe oder nicht einzuhaltende Meilensteine meist mit hohem Aufwand in Form von Powerpoint und Meetings beim Management erklärt werden müssen, um Entlastungen in Form von zusätzlichem Personal oder weniger Arbeitsaufgaben zu erreichen. Engpässe haben zur Folge, dass man z.B. Tests nicht wie geplant abschließen kann, und man auf simulative Testmethoden ausweichen muss, die im Automobilbereich keine hohe Zuverlässigkeit

besitzen wie in verwandten Branchen. Für die Releaseplanung ist es wichtig, dass ausreichend Mitarbeiterressourcen zur Verfügung stehen, um Engpässe meistern zu können. Mit Maßnahmen wie Urlaubssperren können diese Ressourcen über mehrere Monate eingeplant werden führen jedoch meistens auch zu Engpässen nach diesen Sperren. Aus Entwicklersicht führen Kosteneinsaprungen zudem zu Motivationsverlust, da die Entwicklung erschwert wird. Zudem führt Krundendruck und verschiedene Stakeholderinteressen zu Konflikten in der Anforderungsumsetzung. Nicht jeder Kunde hat dieselben Prioritäten wodurch eine einheitliche Problemlösung nicht durchgeführt werden kann. Auf operativer Ebene wird man vom übergeordneten, fest getackteten Prozess gehindert, wenn man bspw. versucht, interne Releasezyklen oder Prozesse zu etablieren, um effizienter zu entwickeln.

## 5.4. Prozesssystematik: Integration

Die Bugfixingphase hat einen sehr starken Einfluss auf die terminliche Umsetzung der für den nächsten Verbundrelease geplanten Funktionalitäten **(F18)** (n=8). Laut drei Experten ist die Beeinträchtigung nicht so groß (n=4), sie arbeiten aber auch mit längeren Releasezyklen.

Das Einplanen eines ausgiebigen Zeitpuffer, hängt stark von der Erfahrung des Managements ab. Je nach Priorität der Features kann es zu Beeinträchtigungen der nächsten Releases kommen. Features des nächsten Releases werden vernachlässigt, wenn Bugfixes von höher priorisierten Features aus der vorherigen Phase gelöst werden müssen. Das Bugfixing wird sehr hoch priorisiert, da es im schlimmsten Fall zu einem Produktionsstop kommen kann, welcher in relativ kurzer Zeit einen hohen finanziellen Schaden verursacht. Lokal Tests sind nur möglich, wenn die Testumgebungen zur Verfügung stehen, bzw. im Voraus entwickelt werden. Das stellt aber aus Managementsicht eine Kostenstelle dar. Bei der Projektplanung steht der Featureaufbau an erster Stelle, aufgrund dessen dass in der Automobilindustrie ein hoher Konkurrenzdruck herrscht. Dadurch wird oft das Bugfixing bei der Projektplanung niedriger priorisiert, obwohl letztendlich deutlich mehr Zeit investiert wird als geplant, da die reale Priorität höher ist als bei der Projektplanung festlegt wurde. Entwickler sollten mehr Zeit in die Validierung der Softwarestände investieren, anstatt sich von Zeitplänen beeinflussen zu lassen. Dies ist jedoch ohne Zustimmung des Managements nicht möglich. Zuletzt führt eine fehlende Kommunikation während der Entwicklung dazu, dass Schnittstellensysteme erst spät validiert werden und man somit gemeinsame Testphasen verschieben muss, da bestimmte Teile in den Schnittstellensystemen in der Releaseplanung erst später eingeplant sind als erwartet.

Aus Entwickler- und Managementsicht (n=8) werden sehr oft, vor allem zu Beginn des Projektes, notgedrungen qualitativ oder inhaltlich suboptimale Softwarestände in den Verbundrelease abgegeben **(F19)**. Einige Experten hingegen (n=4) meinen, dass das selten vorkommt, weil sie entweder direkten Zugang zu Testumgebungen haben (da im selben Haus) oder sie extrem darauf achten, dass das nicht passiert.

Qualitativ suboptimale Softwarestände werden abgegeben, weil die Termineinhaltung höher priorisiert wird, selbst wenn der Reifegrad noch nicht erreicht wurde. Viele Entwickler wollen auch nicht unter Beobachtung geraten, wenn sie Termine nicht einhalten können, weil dies mit Sonderkontrollen und Statusberichten verbunden ist, die dann wiederum den eigenen Entwicklungsprozess verlangsamen. Die Drucksituationen die sich daraus ergeben verschlechtern zudem die Qualität der Abgaben der nächsten Releases. Die größte Herausforderung ist es, qualitativ suboptimale Lieferleistung im Nachgang zu verbessern. In Extremsituationen versucht man mit Workarounds

eine vorläufige Lösung zu finden. Dies ist aber auch nur möglich, wenn die Software nicht lebensbedrohlich ist. Außerdem kann es vorkommen, dass zu Beginn bestimmte Qualitätsziele noch nicht vollständig definiert sind die man erreichen muss. So entwickelt man Softwarestände mit geringer Testabdeckung in den Prüfständen.

Zudem wurde gefragt, welche Aktivitäten den Arbeitsalltag während eines Verbundreleases dominieren **(F20)**. Das Management ist mit der Planung der nächsten Releases beschäftigt (n=6). So werden Hürden beseitigt und sichergestellt, sodass die Teams möglichst effizient durchstarten können. Aus Entwicklersicht wird teilweise an den nächsten Features gearbeitet und teilweise die abgegebene Software weitergetestet, um die Qualität der Abgabe nochmal zu verbessern (n=4). Wenn das Testhaus und die Entwicklung beieinander sind, kann auch während der Entwicklung viel getestet werden, ansonsten muss man auf Simulationen ausweichen. Sie decken aber wie schon mehrmals erwähnt nicht alle Fälle ab, die man im realen Fahrzeug begegnet. Außerdem nutzt man die Zeit, um Dokumentationen zu erstellen oder zu vervollständigen, sowie Testabdeckungen zu überprüfen und zu erweitern (n=2).

Nachintegrationen sind oft als notwendig, da der ursprünglich geplante Verbundrelease oft nicht ausreicht **(F21)**. Aus Managementsicht werden sie oft bevorzugt, anstatt erst im nächsten Release auf die fertige Funktionalität zu warten und somit als notwendig empfunden (n=3). Aus Entwickler- und Managementsicht (n=6) werden sie oft als hindernd empfunden, denn man kann somit keinen Standard bzw. keine Regelmäßigkeit im Prozess etablieren. Die Häufigkeit dieser Außerplanmäßigen Maßnahmen ist so hoch, dass sie teilweise wie normale Releases gesehen werden. Nachintegrationen bieten die Möglichkeit Flexibilität im Prozess einzubauen. So kann auch im Nachgang auf Änderungen reagiert werden, ohne dass man ein halbes Jahr auf den nächsten Release wartet bspw. Ein Experte aus dem Logistik Bereich behauptet, dass sie solche Maßnahmen nicht einleiten müssen (n=1). In den befragten automobilverwandten Branchen sind die Releasezyklen teilweise so lang (ein Jahr), dass Nachintegrationen sogut wie nie eingeleitet werden (n=2). Es kann aber vorkommen, dass Zwischenreleases von den Kunden eingeforert werden, so müssen sie zu jedem Zeitpunkt bereit sein einen funktionierenden Zwischenstand abzuliefern. Das kann in diesen Kontext gut erfüllt werden, da die Simulativen Testumgebungen hoch Zuverlässig sind (n=3). Oft führen auch Umpriorisierungen zu Nachintegrationen. Man sollte Nachintegrationen nur dann in Anspruch nehme, wenn man weiß die Abgabe enspricht nicht den Anforderungen und man sich in der Planung verschätzt hat bzgl. des Fertigstellungstermins.

Einige Experten (n=4) empfinden Nachintegrationen als sinnvoll, wenn der ursprünglich geplante Verbundrelease nicht ausreicht **(F22)**. Andere Experten behaupten, dass durch Nachintegrationen nur Verschiebungen und Änderungen im Zeitplan begünstigt werden (n=8).

Zu den Hauptgründen für Nachintegrationen (F23) wurde folgendes aufgezählt:

- Durch Krankheit bedingter Ressourcenmangel (n = 3)
- Fehler die man erst später findet (n = 4)
- Anforderungen die sich ändern (n = 5)
- Technische Probleme (n = 3)
- Abgabe unfertiger Softwarestände (n = 4)
- Unterschätzen von Problemen oder Zeitpläne (n = 6)

## 5.5. Entwicklungsmethodik: Abstimmung

Zu Beginn wird gefragt, ob der aktuelle Entwicklungsstand zu jedem Zeitpunkt transparent ist **(F24)**. Auf Entwickler- und Teamebene wird der Gesamtentwicklungsstand meist als teilweise transparent wahrgewonnen (n=3). In Unternehmen mit klarer Trennung bei der Systemverantwortung sehen die Entwickler den Gesamtentwicklungstand eher als intransparent an (n=2). Auf Teamebene werden jedoch klare Randbedingungen gestellt und danach wird entwickelt. Daher wird der Entwicklungsstand auf Teamebene meist als transparent wahrgenommen (n=5). In Unternehmen mit Tools zur Anzeige des Gesamtentwicklungsstands wird der Entwicklungsstand als transparent gesehen, da diese Tools aktiv zwischen Kunden und Entwicklung eingesetzt werden (n=2). Ein Transparenter Gesamtentwicklungsstand wird generell mit hohem Kommunikationsaufwand verbunden. Ob sich dieser Aufwand lohnt wird sehr unterschiedlich bewertet. Teams, die sich aktiv mit Kunden über den Entwicklungsstand austauschen empfinden den erhöhten Aufwand nicht als störend, da die Kommunikation über Tools aktiv im Entwicklungsprozess eingesetzt wird. Entwickler in Teams mit klarer Trennung der Systemverantwortung stellen den Sinn der erhöhten Transparenz und den damit einhergehenden erhöhten Kommunikationsaufwand eher in Frage.

Generell wird der Entwicklungsstand anderer beteiligter am Projekt als sehr wenig transparent bezeichnet (n=6) **(F25)**. Die Abstimmung erfolgt in Form von Fehlerlösung nach den Meilensteinreleases. Erst nach dem Release wird klar, ob die Reihenfolge der Entwicklungen bei den beteiligten überhaupt Sinn gemacht hat, da zum Beispiel zum Testen bestimmter Funktionen ganz andere Softwarekomponenten wichtiger gewesen wären. Nach den Meilensteinen müssen zunächst die Schnittstellen überhaupt gesucht bzw. bestimmt werden.

**(F26)** Wie wichtig ist die Transparenz des Entwicklungsstands anderer Beteiligter für Sie? Aus der reinen Theorie betrachtet, sollte die Transparenz nicht so wichtig sein, denn der Verbundrelease wurde konzipiert, damit jeder ungestört auf die einzelnen Meilensteine/Zeitpunkte hinarbeiten kann (n=2). Die Signale bzw. die Projekte auf operativer Ebene laufen auseinander und der Verbundrelease stellte einen Trichter dar, der sie einfängt.

Aus Managementsicht (n=5) ist eine Transparenz nicht im Detail nötig, aber zumindest auf grober Ebene, um die Auswirkungen einschätzen zu können. Kontrollpunkte kann man Werkzeuggestützt mit Projektverwaltungsprogrammen wie Jira etc. einführen. So kann man den aktuellen Status der Entwicklung jederzeit beobachten und verfolgen. Viele Unklarheiten müssen dann durch die aktive Statusnachfrage kompensiert werden und man muss sich darauf verlassen, dass bei Engpässe die Schnittstellensysteme das rechtzeitig melden. Aus Entwicklersicht ist eine hohe Transparenz wichtig wenn man direkt Abhängigkeiten zu Systemen hat (n=4). So sollte man bei Abhängigkeiten die Schnittstellensysteme möglichst schnell und oft zusammenführen, sodass ein reger Austausch stattfindet und wichtige Projektdetails zu Beginn schon detailliert werden, z.B. wann welche Softwarefunktionalität bereitstehen muss oder wann eine Steuergerätekomponente zum Testen bereitsteht etc. Außerdem sollen gemeinsame Schnittstellen Validierungen geschrieben werden, die beide Parteien einsehen können und als Entwicklungsbasis und Kontrollpunkt benutzt werden können.

Eine weitere Frage war, wie stark eine fehlende Transparenz des Entwicklungsstandes anderer Beteiligter die eigene Entwicklung beeinflusst **(F27)**.

Die Meilensteinsystematik basiert zum einen auf die Detailplanung und zum anderen auf das Vertrauen, dass der Plan, so wie er detailliert wurde, auch eingehalten wird. Aus Managementsicht (n=5) teilweise, deshalb sollten sich die Entwicklerteams auf ihre Aufgaben konzentrieren und die

Transparenz der Schnitstellensysteme aktiv anfragen. Gleichzeitig muss man sich darauf verlassen können, dass der Plan befolgt wird und deshalb bracht man aus Sicht eines weiteren Managers Kontrollpunkte die die Transparenz erhöhen. Eine fehlende Transparenz kann also aus dieser Perspektive hindernd sein (n=4). Drei Experten empfinden eine fehlende Transparenz nicht als schlimm (n=3). Schnittstellensysteme sollten sich zumindest zeitnah treffen damit zu bestimmten Zeitpunkten im Projekt integrativ getestet werden kann. Integrationstests sollten deshalb genaustens geplant und mit hoher Transparenz bzgl. der Fertigstellungstermine der Schnittstellensysteme durchgeführt werden, da reine Simulationstests nicht alle Fälle abdecken können. Außerdem sei die aktive Kommunikation in diesem Kontext der Schlüsselfaktor für eine erfolgreiche Erfüllung der Meilenstkriterien.

Außerdem wurde die Bewertung der Kommunikation und Abstimmung sowie die Qualität der Abstimmung in den folgenden Bereichen bewertet **(F28)**.

Innerhalb des Teams sind die Schnittstellen bei allen Befragten bekannt (n=10). Oft dauert das aber lange, bis alle Schnittstellen bekannt sind (n=2). Meistens wird die Projektstruktur mittels eines Organigram dargestellt, wo die jeweilige Rolle der Projektbeteiligten aufgezeigt wird. Außerdem ist die Anzahl der Projektmitglieder relativ überschaubar, sodass man ugf weiß, wer welche Rolle ausübt. Die Qualität der Abstimmung ist sehr Teamabhängig, ist aber i.d.R. relativ gut.

Die Qualität der Abstimmung im Team ist auch gut (n=10). In agilen Projekten hat man eine sehr starke Interaktion und viele Statusmeetings. Dort wird die aktive Präsentation von Zwischenergebnissen gefördert, was zum Teil sehr zeitintensiv ist aber im Gegensatz zu klassischen Vorgehen die Probleme dadurch viel früher entdeckt werden. Das wurde früher vom Projektleiter übernommen. Heutzutage wird das immer mehr von den Entwickler selbst übernommen. Die Kontaktfläche ist viel größer und falls Unklarheiten während der Entwicklung auftauchen ist das Team schon angehalten diese Klärung herbeizuführen. Projektmanagementtools wie Jira z.B. erhöhen die Transparenz und schaffen eine bessere Kommunikationsbasis. Dort können alle Details zu einer Aufgabe dokumentiert werden, sodass man in Extremfällen oder bei Ressourcenausfall durch Krankheit die Aufgaben neu auf das Team verteilt werden können.

Zwischen Entwicklung und Testhaus sind die Schnittstellen und Partner nicht immer bekannt. Außerdem ist die Qualität der Abstimmung relativ schlecht, da es meistens zwei unterschiedliche Bereiche sind und sehr oft auch eine räumliche Trennung herrscht (n=5). Man muss die Schnittstellen aktiv kontaktireren und gemeinsame Testsessions planen oder fehlende Komponenten oder Datenbestände anfragen für die Simulationen etc. In einigen Fällen läuft das aus beiden Sichten gut (Entwickler und Manager, n=4), wenn eine enge Abstimmung zwischen beiden Parteien herrscht. Wenn das Testhaus und die Entwicklung im selben Haus sitzen bestehen keine Probleme, da die Testumgebung stätig und zeitnah an den Projektverlauf angepasst werden kann (n=3). Nicht zuletzt erhöht sich auch der Kommunikations-Level zwischen den relevanten Ansprechpartnern, umso näher der SOP rückt, umso wichtiger die Komponente wird und je nach Riefegrad der Komponente. Die Qualität der Abstimmung zwischen Entwicklung und Testhaus ist eher mäßig (n=7). Aus Managementsicht hängt auch immer vom gegenüber ab, wie weit er sich dann Zeit nimmt und der Sache nachgeht. Oft sind die Ansprechpartner völlig überlastet und man wartet ewig auf Antworten. Wenn beide im selben Haus sitzen ist die Qualität der Abstimmung relativ gut (n=3). Zwei Experten konnten keine Aussage treffen (n=2).

Innerhalb des Unternehmens sind die Schnittstellen in den meisten Fällen und vorzüglich auf Managementebene bekannt (n=5). Die Kommunikation und Hilfsbereitschaft ist sehr Personenabhängig und der Informationsfluss hängt dementsprechend auch davon ab, wie aktiv man Informationen anfragt. Einige Experteten und vor allem Entwickler bewertet die übergreifende Kommunikation als nicht so gut (n=3) aufgrund dessen, dass die Ansprechpartner nicht immer klar definiert sind. Einige wünschen sich mehr Transparenz auf dieser Ebene und sehen es als Aufgabe des Managements eine bessere Übersicht zu schaffen (n=2). Außerdem muss man in vielen Firmen über die Hierarchie gehen, wenn man mit höheren Unternehmensebenen einen Informationsaustausch haben will. Durch agile Methoden wie SAF kann diesem Problem ein Stückweit entgegengewirkt werden, da solche Methoden die Kommunikation fördern (n=2).

Die Qualität der Abstimmung Innerhalb des Unternehmens zu bewerten ist für die meisten Befragten eher schwierig (n=9). Das hängt zum einen von der Kommunikationsbereitschaft der Personen und zum anderen von der Kompetenz und vom Wissenstand des Befragten ab.

Innerhalb des Konzerns sind die Schnittstellen und Ansprechpartner auch nicht immer bekannt (n=8). Oft ist es so, dass die Ansprechpartner teilweise eigene Interessen verfolgen (n=2). Die klassischen Organisationsstrukturen begünstigen in vielen Fällen die Bildung von sogenannten Fürstentümern. So kann es vorkommen, dass sogar Abteilungen gegeneinander arbeiten. Umgekehrt sind viele Firmen auch bemüht, die aktuellen Infos an die einzelnen Ebenen heranzutragen (n=2). In den meisten Fällen muss man wiederrum über die Hierarichie gehen und aktiv nach Informationen fragen und die relevanten Ansprechpartner suchen.

Die Qualität der Abstimmung Innerhalb des Konzerns ist analog zur Qualität der Abstimmung innerhalb des Unternehmens zu bewerden.

Zu externen Lieferanten sind die Schnittstellen in allen Fällen bekannt (n=12). Die Termine werden in enger Abstimmung definiert und die Schnittstellensystemen haben meistens direkten Kontakt zu ihren Lieferanten.

Die Qualität der Abstimmung mit den Zulieferern ist sehr gut und umgekehrt, zum Kunden wird sie auch auch als sehr gut bezeichnen (n=12). Außerdem hängt die Qualität der Abstimmung von der Qualität des Lieferanten ab und des Preisniveaus. In vielen Projekten gibt es fast tägliche Meetings mit dem Lieferanten, sodass man ständig auf den aktuellsten Stand ist.

## 5.6. Entwicklungsmethodik: Testen

Einige Experten behaupten (n=4), dass die Menge der Fehlertickets für das Fehlermanagement und für die Entwicklung unbeherrschbar ist **(F30)**. Aus Managementsicht kann man dies mit zusätzlicher Ressourcen-Aquirierung während den Testphasen beheben. Oft liegen die Ursachen nicht im Testmanagement, sondern vielleicht im falschen Design oder Belastungen aus vorherigen Phasen. Einige wiederum behaupten (n=6), dass es beherrschbar ist. Viele Fehler, wie das absichtliche Liefern von suboptimalen Lösungen in den Verbundrelease, sind außerdem vermeidbar aus Entwicklersicht (n=2). Vieles wird an Dienstleister aus dem europäischen Ausland delegiert, die dann über Supportverträge die niedrigen Prioritäten abarbeiten. Es wird außerdem angemerkt, dass es mittlerweile viele Wege gibt eine Struktur im Projekt zu schaffen. Jedes Fehlerticket wird von einem Menschen erstellt, somit ist das eine überschaubare Menge an Daten die entsteht, die man dann filtern und aggregieren muss, sodass sie eben beherrschbar bleiben. Dies kann man mit klassischen

Ticketsystemen, die beim Automobilhersteller eingesetzt werden nicht umsetzen. Aufgrund dessen, dass man die Logik dahinter nicht selber weiter verbessert sondern man ein Ticketsystem kauft und das über 10 Jahre behältund irgendwann ist es schon veraltet.

Es findet außerdem eine ständige Priorisierung der Fehlertickets statt. So werden anhand der Kritikalität einer Funktion die Tickets höher priorisiert. Dies führt aus Entwicklersicht zu ständigen Verzögerungen. Denn es gibt während der Entwicklung von neuen Features, laufende Wartungsarbeiten an dem aktuellen Softwarstand, welche den aktuellen Entwicklungsprozess vergögern. Dies geschieht in Form von Ticketanfragen. In Extremfällen werden Fehler erst sehr spät gefunden. Diese reichen von funktionalen bis konzeptionellen Fehler oder Daten die anders als geplant geliefert werden, welche eine Folge von unzureichender Kommunikation ist.

Die gestiegene Anzahl an Fehlertickets führt zu Problemen in der Planung und Umsetzung des nächsten Verbundreleases **(F31)**. Diese Aussage wird von den meisten Befragen bestätigt (n=9). Nur in den automotive ähnlichen Branchen, wo die Releasezyklen länger dauern, ist dieses Phänomen nicht so stark ausgerägt (n=3). Wenn sich eine Menge an Arbeit anstaut aus dem letzten Meilenstein und man das vor sich hinschiebt in den nächsten Meilenstein, entstehen zwangsläufig Verzögerungen. Das zieht sich dann über das gesamte Projekt durch. In Bezug auf das V-Modell, gibt es das Konzept des Front-Loading als Projektmanagement-Maßnahme, womit man die Arbeit über die Projektlaufzeit in Relation setzt. Zu Projektbeginn ist die Anzahl der zu bearbeitenden Anforderungen extrem hoch und gegen Ende flacht es wieder ab , so die Theorie. In der Realität verschiebt sich der Berg an Arbeit immer nach hinten, Aufgrund von Nachintegrtionen erhöhte Anzahl an Fehlertickets etc. So fehlen die Kapazitäten im nächsten Releasezyklus. In verwandten Branchen mit längeren Releasezyklen (ca. 1 Jahr) ist das kein Problem (n=3). Dort gibt es auch kontinuierliche Testphasen, die aber in hochausgestatteten Testumgebungen und -Simulationen durchgeführt werden. So kann ein hoher Grad an Testautomatisierung erreicht werden und dies führt zu einer erhöhten Testabdeckung. Im Steuergeräte-Umfeld und vor allem in der Automobilbranche, sind Simulationstests keine Endtests. So können häufig, Aufgrund fehlender Testsinfrastuktur keine Verbundtests durchgeführt werden. Dies führt zu Verschiebungen und Problemen in der Umsetzung des nächsten Releases. Außerdem muss man zwischen Hardware- und Softwarefehler differenzieren. Bei der Fahrzeug-Hardware ist man an den physischen Entwicklungsprozess gebunden, den man evtl. eingehen muss, um ein Problem zu lösen. Softwarefehler sind i.d.R. einfacher und schneller zu lösen. Beides muss jedoch anschließend im Verbund getestet werden. Vor allem wenn starke Abhängigkeiten zwischen beiden Bereichen bestehen müssen zum Testzeitpunkt auch beide Komponenten vorhanden sein für den integrativen Test. Oft hilft man sich auch gegenseitig, sodass man Verschiebungen und Engpässe im Prozess ausmerzt, um den Projektverlauf wieder zu stabilisieren.

Außerdem wurden die Gründe für die hohe Anzahl an Fehlertickets abgefragt **(F32)**. Ein Experte zählt folgende Gründe auf (n=1). Ein Grund ist, wenn das Prinzip des Front-Loadings scheitert und man die Welle an Arbeit vor sich hinschiebt. Oft startet man ein Projekt und ist Vorbelastet aus den Vorprojekten. Wenn für Testdurchläufe Daten aus den vorherigen Generationen durchgeführt werden und dies zu Problemen in den Schnittstellensystemen führen. Dann schiebt man diese Welle vor sich hin, weil man am Anfang nicht sofort reingekommen ist. Außerdem technische Probleme und Ressourcen. In Automobilprojekten plant man i.d.R. immer Puffer Phasen ein, wo die Welle dann wieder reinsacken kann. Technische und Ressourcen Probleme.

Ein weiterer Experte meint, dass eine fehlende Testabdeckung bzw. eine schlechte Testbarkeit auch zu Problemen führt (n=1). Die Aufmerksamkeit die über den Arbeitstag verteilt sinkt. Ebenso eine niedrige Qualität des anfänglichen Designs, führt dazu dass es später nicht gut getesten werden kann. Verständnisprobleme bzgl. der Anforderung sowie Sprachbarrieren führen zu fehlerhaften Umsetzungen.

Ein weiterer Experte (n=1) betohnt eine späte Absicherung sowie nicht vorhandene Tests. Dadurch werden viele Sachen erst kurz vor dem SOP bekannt. Sie werden im Zweifel einfach geduldet oder nicht mehr gefixt. Das fällt dann dem Kunden auf, die Kunden melden sich und dadurch entstehen die Tickets. Spätes Testing hat auch zur Folge, dass der Endtest beim Kunden stattfindet. Im Zweifel sind es Komponenten die lebensbedrohlich sind, sodass ausgiebig getestet wird. Nichtsdestotrotz kann es immer auch im Zusammenspiel mit anderen Komponenten Abhängigkeiten geben die dann ein Fehlerbild auslösen. Zudem werden häufig nur Testspezifikationen und Anforderungsspezifikation durchgeführt aber nicht umgesetzt, weil die Kapazitäten fehlen.

Auf die Frage, ob zu jedem Verbundrelease alle geplanten Änderungen am Steuergeräteverbund vollumfänglich getestet werden müssen **(F33)** haben einige Befragten (n=7) mit nein beantwortet. Sie sind der Meinung, dass nur die umgesetzten Änderungen vollumfänglich getestet werden müssen. Wenn eine geplante Funktion nicht umgesetzt wird, muss man sie auch nicht testen. Drei Experten (n=3) konnten keine genaue Aussage treffen. Zwei Experten (n=2) behaupten, dass jede Komponente getestet werden und ein Teststatus zurückgeben muss. Edge Cases werden meistens erst spät im Projektverlauf entdeckt und können deshlab erst da getestet werden. Aus Entwicklersicht würde man gerne weniger am Meilensteinprozess gekoppelt sein, sodass unfertige Features nicht zwangsläufig abgegeben werden müssen und somit unnötigerweise vollumfänglich getestet werden. Denn das ist mit einem hohen Ressourcenverbrauch verbunden. Oft entstehen Nachintegrationen aufgrunddessen, dass streng am Meilensteinprozess released wird, obwohl im Voraus schon festteht, dass eine Nachintegration nötig sein wird.

Alle Steuergeräte müssen laut einer vielzahl von Befragten (n=8) vollumfänglich am Ende eines Projekts kurz vor SOP getestet werden **(F34)**. Außerdem in Abhängigkeit zu den Änderungen (n=2). Zwei Experten (n=2) konnten keine genaue Aussage treffen. Dennoch wird das nicht testen von Funktionalität als sinnlos betrachtet. Außerdem wird betont, dass alles was eine Auswirkung auf einen anderen Prozess haben kann vollumfänglich getestet werden muss. Aus Entwicklersicht ist es sinnvoll alles vollumfänglich zu testen und automatisiert wie möglich.

Auf die Frage, ob zu jedem Verbundrelease für alle Steuergeräte alle Arten an Tests durchgeführt werden müssen **(F35)** einige Experten (n=6) mit nein geantwortet. Die Testfälle sollten priorisiert werden, sodass die neu hinzugekommenen Funktionen zuerst getestet werden. Das ist ein sehr hoher Aufwand, den man gerne am Ende des Projektes durchführen kann. Aus Entwicklersicht hat jede Testart seinen eigenen Fokus, somit muss jedesmal vollumfänglich getestet werden (n=3). Regressionstests müssen immer durchgeführt werden. Sie werden in jedem Release auch immer größer, sodass man wiederum priorisieren muss, welche ausgeführt werden. Weitere Experten (n=3) behaupten, dass das davon abhängt wie sicherheitskritisch eine Funktion ist.

Tests die nicht so umfangreich sind können unter Umständen auch in eine Pipeline integriert und automatisiert ausgeführt werden. So wie sich die Anforderungen ändern, ändern sich die finalen Tests, somit kann es vorkommen, dass einigie Tests keinen Sinn mehr machen gegen Ende.

# 6. Interpretation

Die nachfolgenden Informationen wurden direkt aus den Ergebnissen der durchgeführten Studie bei Dr. Ing. h. c. F. Porsche AG entnommen und deshalb auch als Zitate in dieser Arbeit hinterlegt. Die bereits durchgeführte Studie wird voraussichtlich im Jahr 2019 veröffentlicht.

## 6.1. Studienvergleich

Nachfolgend werden die Daten aus der durchgeführten Studie bei Dr. Ing. h. c. F. Porsche AG mit den erhobenen Daten aus der aktuellen Studie verglichen.

### 6.1.1. Kategorie: Context

"Die durchschnittliche Berufserfahrung bei Porsche beträgt etwas weniger als 6 Jahre, mit einem Minimum von einem Jahr und einem Maximum von sechzehn Jahren." [19] Die durchschnittliche Berufserfahrung in der aktuellen Studie liegt bei 4,5 mit einem Minimum von 0,5 Jahren und einem Maximum von 15 Jahren. Somit ist die durchschnittliche Berufserfahrung um 1,5 Jahren höher in der bereits durchgeführten Studie, als bei der aktuellen Studie. "Die meisten Befragten (n = 17; 44%) üben eine Managementrolle aus." [19] Dies lässt sich auch in dieser Studie feststellen, hier arbeiten 50% der Befragten in Managementrollen. "Andere Experten waren verantwortlich für Projekte, Produkte, Funktionen, Integration, Testing, Qualität, Daten Prozesse, oder andere verwandten Disziplinen. Zehn der Teilnehmern (26%) sind stellvertretend für die Oprative Ebene." [19] Die operative Ebene wird mit einem Anteil von 25% in der aktuellen Studie vertreten. "Die restlichen zwölf Teilnehmer (30%) sind in der Qualifikationsphase verwickelt, die auf Meilensteinebene ausgeführt wird." [19] Hier ist exakt nur ein Experte in dieser Tätigkeit verwickelt (8,3%).

"Die meisten Studienteilnehmer bei Porsche gaben an, in Fahrzeugprojekten (n = 24), der Entwicklung von E/E-Komponenten (n = 18), der Entwicklung von Funktionen (n = 14), der Entwicklung von Softwarekomponenten (n = 12) und vernetzten Diensten (n = 14) zu arbeiten."[19] "Andere (n = 7) befassten sich mit IT-Backend, projektübergreifender Integration, verteilten Funktionen oder der Qualität."[19] "14% der Befragten gaben an, dass ihr Projekt sicherheitskritisch ist."[19] Die Bereiche die in dieser Arbeit abgedeckt werden sind

- Motorelektronik (n=1; 8,3%)
- Sicherheitsrelevant (n=1; 8,3%)
- IT-Logistik (n=2; 16,6%)
- Funktionen-/Softwareentwicklung (n=4; 33,3%)
- E/E-Bauteile (n=3; 25%)

- Kabelbaum und Steuergeräteproduktion (n=1; 8,3%)

Die Bereiche die beide Studien Abdecken sind: Motorelektronik, E/E-Bauteile,und die Entwicklung von Softwarekomponenten und von Funktionen. Der Backendbereich konnte mit dem Sampling von einem Experten abgedeckt werden.

“Die meisten Teilnehmer ordneten ihr Projekt dem Bereich Infotainment zu (n = 13), gefolgt von Elektronik für Karosserien (n = 11) und Elektronik für Motoren (n = 7). In Bezug auf die 24 zusätzlichen Klassifikationen gaben zehn Teilnehmer an, an Querschnittsthemen zu arbeiten (F4).”[19] In dieser Arbeit deckt ein Experte den Infotainmentsystem ab. Ebenso wie ein Experte den Bereich von Elektrik für Motoren abdeckt.

“Die meisten Befragten gaben an, traditionelle Entwicklungs- oder Projektmanagement-Ansätze wie das V-Modell oder sequentielle Ansätze zu verwenden (n = 26). Nur sechs Befragte verwendeten angepasste agile Methoden, und sieben Personen verwendeten hybride Ansätze, die als stark angepasste agile Methoden oder als Verwendung nur einzelner agiler Praktiken definiert wurden. Dies zeigte, dass zu diesem Zeitpunkt nur ein Drittel der Studienteilnehmer agile Konzepte einsetzte. Agile Implementierungen basierten auf Scrum oder der Porsche-spezifischen Anpassung der agilen Methode. Eine Person berichtete sogar über skaliertes Agile und Lean auf Einheitenebene in Kombination mit einem angepassten Scaled Agile Framework (SAFe). In traditionellen Projekten wurden einzelne agile Vorgehensweisen wie tägliche Standups, User Stories, Backlogs, Rückblicke oder die Scrum-Master-Rolle verwendet. Einige Befragte gaben an, sowohl agile als auch traditionelle Ansätze auf verschiedenen Projektebenen zu verwenden. In einer Antwort wurde angegeben, dass Agile auf Teamebene zusammen mit dem V-Modell für ganze Projekte verwendet wird, während ein anderer Befragter aufgrund hochdynamischer Änderungen der Anforderungen einen sprintartigen Ansatz innerhalb des V-Modells angab. Ein anderer Befragter gab die Verwendung unterschiedlicher Entwicklungsparadigmen in verschiedenen Lebenszyklusphasen an.”[19]

Ähnliche Projekt- bzw. Prozessmanagement Methodiken werden auch hier verwendet. In dieser Arbeit haben Teil der Befragten an (n=1) das SAF-Framework zu verwendet als Management- bzw. Entwicklungsmethodik. Weitere Befragte (n=2) gaben an, SAF und SCURM für größere Projekte zu verwenden als Managementmethodik, sonst klassisch nach Wasserfall zu arbeiten. Desweiteren verwenden einige Befragten (n=5), ein Teil davon aus der automotive ähnlichen Branche (n=3), ein Mix aus V-Modell und agilen Methoden. Oft ist es aber so, dass obwohl sie angeben agil zu arbeiten, in der Realität klassische bzw. traditionelle Entwicklungsmethodiken nach V-Modell dominant auf den Prozessverlauf einwirken. Ein weiterer (n=1) gab an klassich nach V-Modell zu arbeiten. Ein Experte (n=1) setzt für für die Projektorganisation und Entwicklung Kanban und vereinfachtes SCRUM ein. Ein weiterer Experte gab an reines SCRUM zu verwenden (n=1).

### 6.1.2. Prozesssystematik: Verbundrelease

“Die Experten sollten außerdem die aktuelle Anzahl an Verbundreleases über die gesamte Projektlaufzeit beurteilen. Auch bei Porsche gab die Mehrheit der Teilnehmer (n = 22; 56%) an, dass die aktuelle Anzahl der Releases einschließlich aller zusätzlichen Qualifizierungsphasen und Sonderqualifizierungen, zu hoch ist. Man erkennt auch hier eine große Differenz zwischen den Antworten der Manager und denen der Entwickler. Aus Managementsicht (n = 17) ist die Anzahl der vorhandenen Releases zu hoch (56%), während für die Entwickler, vor allem um nach agilen Prinzipien

arbeiten zu können, die Anzahl zu niedrig ist (25%). Außerdem wird von den Entwicklern mit agilen Background auch bei Porsche bestätigt, dass die absolute Anzahl der Qualifizierungsphasen zu gering ist, um einen agilen Prozess zu etabblieren." [19]

Auch in der aktuellen Studie empfindet das Management die Anzahl der Verbundreleases als zu hoch (n=5; 41,7%). Ein Experte (n=1; 8,3%) mit Managementbackground gab an, dass die Anzahl per se nicht so hoch ist, aber der bürokratische Aufwand für die Vorbereitung der Meilensteine zu hoch ist. Aus Entwicklersicht (n=4; 33,3%) ist die Anzahl deutlich zu kurz, sodass oft etwas nachgeliefert werden muss. Zwei Experten konnten keine genaue Aussage treffen.

Zur Aussage, ob immer abgabereife Software-Lieferleistungen erbracht werden gaben 60% der Umfrageteilnehmer bei Porsche an, dass "das Ergebnis selten in der geforderten Qualität verfügbar ist." [19] Dies wiederspiegelt sich auch in den erhobenen Daten aus dieser Studie. "Im Gegensatz dazu antworteten 40% der Teilnehmer bei Porsche, dass es immer oder zumindest die meiste Zeit möglich ist, für jede angeforderte Version eine Lieferversion zu erstellen."[19]

In der aktuellen Studie gab ein Experte an (n=1), dass alle zwei Monate abgabereife Software-Lieferleistungen erbracht werden können. Zwei weitere Experten behaupten (n=2), dass sie realistisch alle drei Monate releasen können, aber die Tendenz Richtung zwei Monate geht. Im Logistik Bereich finden die Releases maximal zwei Mal im Jahr statt (n=2). Ein weiterer Experte behauptet (n=1), dass sinnvolle Releases alle 4 Monate erbracht werden können, aber die Abgabereife sei projektabhängig, sodass auch kürzere Zyklen erreicht werden können. In den Automobilverwandten Branchen wird ein Mal im Jahr released (n=3). Dies ist durch die hohe Komplexität des Produktes determiniert. Drei Enwickler sagen (n=3), dass es von der zu entwickelnden Software abhängt. Dabei bewegt sich die Dauer der Releasezyklen zwischen zwei Wochen (Applikationssoftware) und zwei bis sechs Monate (Basissoftware und Bootloader).

Es konnten keine überstimmenden Aussagen getroffen werden, sodass die Ergebnisse nicht verglichen werden können.

"74% der Teilnehmer bei Porsche gaben an, dass sie zum größten Teil rechtzeitig Feedback zu Qualifizierungsphasen erhalten." [19]

Dies wiederspiegelt sich nicht in den erhobenen Daten aus dieser Studie. Zwei Experten (n=2; 16,6%) konnten keine genauen Angaben dazu machen, da sie zu wenig im Verbundrelease eingebunden sind. Drei Experten (n=3; 25%) meinen, da vieles nicht vollautomatisiert getestet werden kann, erhält man oft nur Feedback, wenn der Kunde direkte Rückmeldung gibt. Aus Managementsicht (n=5; 33,3%) wird nicht immer rechtzeitig Rückmeldung gegeben. In der automotive verwandten Branche bestehen keine Probleme bzgl. der rechtzeitigen Rückmeldung aus dem Verbundrelease (n=2; 25%).

"46% der Prosche Befragen, fordern zusätzliche Qualifizierungsphasen mit reduzierten Testumfängen." [19] Ein ähnliches Ergebniss wird auch in der aktuellen Studie deutlich. Ein relativ hoher Anteil der Befragten (n=5; 33,3%) sehen das als hilfreich, mit der Begründung, dass man dadurch stabilere Lieferstände generieren kann. Einige Experten behaupten (n=3; 25%), dass das nicht so hinfreich wäre. Weitere Experten (n=4; 33,3%) sehen das als sinnvoll.

"Auf die Frage, wie oft ein Verbundrelease stattfinden sollte, damit man immer einen abgabereifen Software-Lieferstand liefern kann geben zwei Drittel der Teilnehmer bei Porsche an, dass die Qualifizierungsphasen mindestens vierteljährlich stattfinden sollten." [19]

Ähnliche Aussagen werden auch in der aktuellen Studie erkennbar, wobei hier nochmal betont wird, dass die Tendenz Richtung 2 Monate geht.

### 6.1.3. Prozesssystematik: Planung

“Aus den Proscheergebnissen erkennt man eine gespaltene Meinung bezüglich der Sinnhaftigkeit und der Möglichkeit eine Initialaufplanung zu erstellen. Demnach empfanden 50% der Befragten es als möglich und 50% als selten möglich.”[19]

Dies wiederspiegelt sich teilweise in den Daten aus der aktuellen Studie. Für ein Großteil der Experten empfindet das als nicht sinnvoll (n=8; 66,6%). Eine Initialaufplanung wird als sinvoll gesehen, wenn die Bereichsabhängig durchgeführt wird, z.B. bei einer Werkplanung, oder Maschinenbeschaffung. Auch wenn die Aufgaben zu Beginn noch nicht klar sind, ist eine Initialaufplanung sinnvoll (n=4; 33,3%).

“Aus den Porschedaten kommt zudem hervor, dass es schwierig ist eine erste Planung zu erstellen, da zu Beginn eines Projekts noch keine Entscheidungen für oder gegen einen Lieferanten getroffen wurden. Es wird eine allgemeinere Aussage getroffen, dass eben viele Prozessanforderungen erst während des Projektes und nach der Ausführung von Tests detailliert werden können.”[19]

Dies wird auch aus den Daten der aktuellen Studie ersichtlich und bestätigt. Vieles steht im Voraus noch nicht fest und kann somit auch schwer geplant werden. Außerdem ändern sich die Anforderungen ständig im laufenden Betrieb, sodass die Initailaufplanung immer überarbeitet werden muss.

“Eine signifikante Mehrheit (74%) gab an, dass die Planung zu Beginn eines Projekts sinnvoll ist, da sie ein belastbarer Ausgangspunkt für weitere Schritte ist. Die Teilnehmer erwähnten auch den bestehenden Change-Management-Prozess, der jederzeit Aktualisierungen ermöglicht.” [19]

Aus den erhobenen Daten dieser Studie kommt konträrerweise hervor, dass ein großer Teil der Befragten es nicht unbedingt als sinnvoll erachtet (n=8; 66,6%), da der Plan sich ständig im Projektverlauf ändert.

“80% der Prosche Befragen antwortete, dass die geplanten Inhalte in der Praxis häufig nicht umsetzbar sind. Die Mehrheit der Teilnehmer gab an, dass immer noch ein Bewusstsein für hohe Planungsqualität besteht. Planungsaktualisierungen müssen von einem Ausschuss durchgeführt werden. Dies ist ein Grund, warum Änderungsanforderungen nicht in der aktuellen Version implementiert werden. Auch einige Bereiche wie „Connected Car“, sind sehr dynamisch, was ein weiterer Grund für den schlechten und nicht aktuellen Planungsstand ist.” [19]

Dies kommt in großem Maß auch aus den erhobenen Daten in dieser Studie hervor. Der Plan wird ständig umgeworfen und aktualisiert, aber er wird trotzdem gemacht, da dies der einzige Weg ist, um an den Verbundrelease teilnehmen zu können. Standards und Normen fordern zudem auch eine detaillierte Vorausplanung, da man sich in einem sicherheitskritischen Umfeld befindet.

“Informationen über Planungsdetails von den relevanten Ansprechpartnern zu erhalten, wird von 74% der Porsche Befragten als große Herausforderung empfunden. Es ist schwierig, rechtzeitig Informationen zu erhalten, da es weder geregelte Aufgaben noch einen konsistenten Informationsfluss für die Änderung der relevanten Informationen gibt” [19], was auch mit den Daten aus der aktuellen Studie von 50% der Befragten bestätigt wird.

“Ein weiterer Punkt, der erwähnt wurde, ist die Auswirkung von Managemententscheidungen während des Entwicklungszyklus (F16). 90% der Porsche Befragten bewerteten den Einfluss von Managemententscheidungen als stark oder sehr stark und gaben an, dass die Entwicklung neuer Funktionen unter unerwarteten Änderungen leidet, die vom Management gefordert werden. Einige Befragte beklagten sich über Managemententscheidungen, die die Priorität des Rückstands ändern und schwerwiegende Auswirkungen auf die weiteren Verfahren haben.” [19] Dieselben Symptome werden auch aus den erhobenen der aktuellen Studie Daten ersichtlich. Die ständige Umpriorisierung von Anforderungen erschwert eine lineare Entwicklung und führt zu Verzögerungen in den nächsten Releasephasen. Ebenso führen außerplamäßige Managemententscheidungen zu Ressourcen-Engpässe.

### 6.1.4. Prozesssystematik: Integration

Laut 87% der Prosche Befragten beinträchtigt das Bugfixing sehr stark die termingerechte Umsetzung der für den nächsten Release geplanten Funktionalitäten. “Da im Projektplan keine Hold Phase vorhanden ist, kommt es sogar zu Verzögerungen bei den nächsten geplanten Funktionen (F18).” [19] Zu ähnlichen Schlussfolgerungen kommt man auch in der aktuellen Studie. Es wird zusätzlich angemerkt, dass die Bugfixing Phase großzügig eingeplant werden muss. Zusätzlich, sollte man den manuellen Testumfang drastisch reduzieren und einen hohen Grad an Automatisierung einführen. Dazu müssen die Testumgebungen besser ausgestattet sein, sodass Simulationstests oder integrative Tests verlässlicher ausgeführt werden können.

“Die Hauptaktivitäten oder -aufgaben bei Porsche die mit der jeweiligen Rolle verbunden sind, sind: Das Management ist mit der Koordination beschäftigt und stellt den geplanten Umfang in Bezug auf die nächste Version sicher. Auf operativer Ebene stehen die Verfolgung der Testergebnisse und die Analyse anstehender Fehlertickets im Vordergrund. Beide Gruppen müssen die Nachlieferungen abwickeln.” [19] Zu denselben Ergebnissen kommt man auch in der aktuellen Studie (siehe 5.4). Man versucht den Prozess voranzutreiben. Aus Entwicklersicht, wenn man bewusst suboptimale Lösungen geliefert hat, ist man darauf bedacht, die Fehler während dieser Phase so schnell wie möglich zu beheben.

“Fast alle Porsche Befragten (96%) gaben an, dass die Bereitstellung von Softwareversionen mit hoher Qualität unmöglich ist, wenn sie auch die für die nächste Version geplanten Inhalte bereitstellen müssen. Die für die Integration in Betracht gezogenen Ergebnisse befinden sich daher aufgrund des zunehmenden Kosten- und Termindrucks in einem frühen Stadium der Reife. Aus diesem Grund wurden zusätzliche Qualifizierungsphasen nach dem ursprünglichen Stichtag festgelegt.” [19] 66,6% der Experten aus der aktuellen Studie bestätigen das auch. Die Featureaufbauphase des nächsten Releases wird durch die Bugfixes beeinträchtigt, die oft mit einer höheren Priorität abgearbeitet werden müssen.

“Zudem wurde gefragt, ob solche zusätzlichen Qualifizierungsphasen notwendig (F20) und sinnvoll (F21) sind. 65% der Teilnehmer hielten zusätzliche Qualifizierungsphasen für erforderlich und 35% für angemessen.” [19] Eine deutlich geringere Anzahl der Experten aus der aktuellen Studie empfinden das als notwendig (25%). Außerdem empfinden 50% der Experten Nachintegrationen als störend, weil sie eine regelmäßigke Ausführung des Prozesses hindern.

“Als Hauptgründe für die spätere Integration nannten die Teilnehmer schlechte Softwarequalität, mangelnde Termintreue der Lieferanten, schlechte Planung ohne Pufferung und keine vollständige Fehlerbehebung aus der vorangegangenen Qualifizierungsphase.” [19] Ähnliche und weitere Motivationen findet man auch in der aktuellen Studie wieder:

- Durch Krankheit bedingter Ressourcenmangel (n = 3; 25%)
- Fehler die man erst später findet (n = 4; 33,3%)
- Anforderungen die sich ändern (n = 5; 41,6%)
- Technische Probleme (n = 3; 25%)
- Abgabe unfertiger Softwarestände (n = 4; 33,3%)
- Unterschätzen von Problemen oder Zeitpläne (n = 6; 50%)

### 6.1.5. Entwicklungsmethodik: Abstimmung

“Die nächsten Fragen betrafen die Transparenz des Status von Projekten durch relevante Ansprechpartner und relevante Gegenstücke. Hier gaben nur 15% (n = 6) der Porsche Befragten an, dass der Entwicklungsstatus anderer Projekte für sie transparent ist. Die meisten Teilnehmer (n = 19; 49%) eine teilweise Transparenz an, während 36% (n = 14) einen Mangel an Transparenz angaben. Gründe für die mangelnde Transparenz waren fehlende Zeit- und Koordinierungsmechanismen sowie die Verwendung veralteter Inhalte der Freigabepläne.” [19]

Dies kann auch mit der Datenerhebung der Studie untermauert werden. Auf Entwickler- und Teamebene wird der Gesamtentwicklungsstand meist als teilweise transparent wahrgewonnen (n = 3; 25%). Hier wird aber auch betont, dass die Transparenz nur zu direkten Schnittstellensystemen von Interesse ist, da die Komplexität das gesamte Projekt zu überblicken einfach zu groß ist. In Unternehmen mit klarer Trennung bei der Systemverantwortung sehen die Entwickler den Gesamtentwicklungstand eher als intransparent an (n=2; 16,6%). Auf Teamebene werden jedoch klare Randbedingungen gestellt und danach wird entwickelt. Daher wird der Entwicklungsstand auf Teamebene meist als transparent wahrgenommen (n=5; 41,6%). In Unternehmen mit Tools zur Anzeige des Gesamtentwicklungsstands wird der Entwicklungsstand als transparent gesehen, da diese Tools aktiv zwischen Kunden und Entwicklung eingesetzt werden (n=2; 16,6%).

“Die Transparenz des Status Quo eines bestimmten Entwicklungsprojekts ist sehr wichtig und eng mit der Qualität einer Veröffentlichung verbunden. 95% der Befragten befürworteten die Aussage, dass es jederzeit wichtig ist, eine transparente Softwareversion zu haben. Aufgrund der Komplexität, Abhängigkeit und Konnektivität des Software-Engineerings ist dies dringend erforderlich.” [19]

In der Aktuellen Studie legen die Befragten einen hohen Wert auf die Transparenz zu den direkten Ansprechpartnern, jeoch Projektübergreifend ist das oft nicht möglich, da die Komplexität zu hoch ist. Der Meilensteinprozess sollte sie aus dieser Hinsicht entlasten, da im Meileinstein alle Ergebnisse abgefangen werden, sodass die SE-Teams ungestört an ihren Aufgaben weiterarbeiten können.

‘Eine weitere Frage zielt darauf ab, Informationen über die Kommunikationsstrukturen innerhalb des Unternehmens und der beteiligten Personen aus dem Freigabeplanungsprozess zu erhalten (F25). Die Teilnehmer mussten bewerten, ob sie ihre Schnittstellen und relevanten Stakeholder kannten und ob die Qualität der Koordination gut war. Diese Bewertung musste für mehrere Schnittstellen erfolgen:

innerhalb des Teams, zwischen Team und Test, innerhalb der Fallfirma, innerhalb der Firmengruppe sowie gegenüber externen Lieferanten. Die Kommunikation innerhalb eines Projekts wurde als gut empfunden, die Qualität jedoch als abnehmend in der Kommunikation innerhalb des Unternehmens und noch schlechter in der Kommunikation mit Lieferanten ( intern bedeutet Unternehmensgruppe und externe Lieferanten). In ähnlicher Weise wurden die relevanten Stakeholder und Schnittstellen des breiteren Projektkontexts als weniger bekannt gemeldet als diejenigen innerhalb des Teams." [19]

Aus den Daten kommen ähnliche Aussagen hervor. Die Kommunikation zu Ansprechpartner und relevanten Schnittstellen innerhalb des Teams ist qualitativ gut. Je hoher man in die Hierarchie hochwandert, desto weniger sind auch die Ansprechpartner bekannt und die Qualiät der Absimmung nimmt auch ab.

### 6.1.6. Entwicklungsmethodik: Testen

"Die erste Frage zielte darauf ab zu bewerten, ob die Anzahl der Fehlerberichte noch von der Entwicklung oder vom Fehlermanagement beherrscht werden kann. Insgesamt waren sich 56% (n = 22) der Teilnehmer einig (n = 7) bzw. einig (n = 15), dass die Entwicklung die hohe Anzahl an Fehlerberichten beherrschen kann. Die übrigen Befragten waren eher anderer Meinung (n = 9) oder anderer Meinung (n = 8). In Bezug auf das Fehlermanagement stimmten die meisten Teilnehmer (n = 25) nicht zu (n = 8; 21%) oder neigten dazu, nicht zuzustimmen (n = 17; 44%). Die Minderheit der Teilnehmer stimmte zu (n = 5; 12%) bzw. stimmte zu (n = 9; 23%)." [19]

Die Ergebnisse aus der aktuellen Studie sehen folgendermaßen aus: Einige Experten behaupten (n=4: 33,3%), dass die Menge der Fehlertickets für das Fehlermanagement und für die Entwicklung unbeherrschbar. Aus Managementsicht kann man dies mit zusätzlicher Ressourcen-Aquirierung während den Testphasen beheben. Oft liegen die Ursachen nicht im Testmanagement, sondern vielleicht im falschen Design oder Belastungen aus vorherigen Phasen. Einige wiederum behaupten (n=6; 50%, dass es beherrschbar ist. Viele Fehler, wie das absichtliche Liefern von suboptimalen Lösungen in den Verbundrelease, sind außerdem vermeidbar aus Entwicklersicht (n=2; 16,6%). Daraus kann man schließen, dass die Menge an Fehlertickets für die die Entwicklung in beiden Studien beherrschbar ist.

"Darüber hinaus wurden die Teilnehmer nach den Gründen für die hohe Anzahl an Fehlerberichten / Tickets befragt. Die Umfrage ergab, dass die Identifizierung von Fehlern in der Regel nicht vor der bevorstehenden Veröffentlichung erfolgt, da die Entwicklungszeit, die Kosten und der Termindruck nicht ausreichen. Es wurde berichtet, dass die Intensität der Prüfung durch den Lieferanten nicht ausreichend war. Weitere Gründe für die hohe Anzahl von Fehlertickets sind die zunehmende Komplexität des Produkts selbst, die mangelnde Koordination innerhalb des Teams und das unzureichende Requirements Engineering. Generell kann festgestellt werden, dass die Softwarequalität vor einer Qualifizierungsphase unzureichend und fraglich ist, was den Erfolg der Qualifizierungsphase gefährdet." [19]

Die Gründe aus der Porschestudie wiederspiegeln sich auch in der aktuellen Studie. Zu den Gründen aus der laufenden Studie zählen, die absichtliche Abgabe von suboptimalen Lieferständen. So ist die Entwicklung immer durch Bugfixings belastet und kann seine Aufgaben nicht abarbeiten. Dies

führt zu Ressourcen Engpässe, die sich über den gesamten Projektverlauf durchziehen. Mangelnde Kommunikation zwischen Schnittstellensysteme ist auch ein großes Problem. Dies wird oft durch die räumliche Trennung zwischen den Schnittstellensystemen begünstigt.

“Softwareänderungen können schwerwiegende Auswirkungen auf die Schnittstellen haben, weshalb Tests durchgeführt werden müssen. 18% der Befragten (n = 7), die angaben, dies sei nicht erforderlich, sahen nicht die Notwendigkeit, die Software-Änderungen für jede Qualifizierungsphase in vollem Umfang zu testen. Die meisten Befragten (n = 18; 46%) gaben an, dass Änderungen für jede geplante Veröffentlichung in vollem Umfang getestet werden müssen. Die restlichen 36% (n = 14) stimmten teilweise darin überein, dass Tests immer notwendig sind, und spezifizierten in den Kommentaren bestimmte Situationen, in denen mehr Tests notwendig waren oder weniger Tests akzeptabel waren. Einige gaben an, dass der Umfang der Tests von der Anzahl der vorgenommenen Änderungen oder von der Entwicklungsphase abhängt. Ein Befragter gab an, dass es nicht möglich sei, alle Änderungen zu testen. Ein anderer meinte, dass ein vollständiger Test immer notwendig ist, da Abhängigkeiten nur sichtbar werden, wenn innerhalb eines Releases getestet wird. Nur 10% (n = 4) der Befragten stimmten zu, dass alle Arten von Tests in jedem Freigabezyklus durchgeführt werden müssen. 39% (n = 15) stimmten dieser Aussage nicht zu und etwa die Hälfte (n = 20; 51%) stimmte teilweise zu. Die Teilnehmer wiesen ferner darauf hin, dass die Durchführung aller Tests nicht durchführbar ist oder dass die erforderlichen Testarten in der Teststrategie vordefiniert sind und von der Änderung selbst abhängen. Andere berichteten, dass Regressionstests häufig ausreichen oder dass vollständige Releases genauer getestet werden müssen als Teil-Releases.” [19]

In der aktuellen Studie haben einige Experten (n=6; 50%) mit nein geantwortet. Die Testfälle sollten priorisiert werden, sodass die neu hinzugekommenen Funktionen zuerst getestet werden. Das ist ein sehr hoher Aufwand, den man gerne am Ende des Projektes durchführen kann. Aus Entwicklersicht hat jede Testart seinen eigenen Fokus, somit muss jedesmal vollumfänglich getestet werden (n=3; 25%). Regressionstests müssen immer durchgeführt werden. Sie werden in jedem Release auch immer größer, sodass man wiederum priorisieren muss, welche ausgeführt werden. Weitere Experten (n=3; 25%) behaupten, dass das davon abhängt wie sicherheitskritisch eine Funktion ist.

Ein deutlich höherer Anteil der Befragten der aktuellen Studie hat mit nein beantwortet. Tendenziell ist die Meinung, dass man auf Standardtests wie Regressionstests zurückgreifen sollte und die Durchführung aller Testarten nicht als sinnvoll erachtet wird.

“Um Testaufwand zu sparen, ist es wichtig zu wissen, wann umfassende Tests (einschließlich aller Arten von Tests) aller Steuergeräte durchgeführt werden müssen . 85% der Befragten (n = 33) gaben an, dass Tests in Abhängigkeit von Software, Hardware oder Funktionsänderungen durchgeführt werden müssen. Fünf Befragte (13%) gaben an, dass die ECUs zu Beginn oder am Ende einmal pro Qualifizierungsphase getestet werden müssten. 2% (n = 1) gaben an, dass Tests nicht immer notwendig sind. Ein Teilnehmer bemerkte, dass aufgrund der hohen Produktkomplexität und der geringen Softwarequalität alle Steuergeräte als integriertes System mit allen möglichen Tests oder zumindest mit guten Regressionstests getestet werden müssen. Ein anderer behauptete, dass ein umfassender Test nicht für alle Systemteile möglich sei, aber große Teile mit einer guten Teststrategie abgedeckt werden könnten.” [19]

Ähnliche Gründe werden auch in dieser Studie aufgezählt. Aus den Daten kam hervor, dass nicht alle Steuergeräte vollumfänglich getestet werden müssen. Die Tests sind oft viel zu aufwändig und man muss eine Ingenierusmäßige Abschätzung durchführen und dabei die Tests prio risiert ausführen.

## 6.2. Auswertung der Daten mit Hilfe der GTM

Für die Auswertung der Experteninterviews wird von einem idealen Prozessverlauf ausgegangen (Abbildung 6.1).


**Abbildung 6.1.:** Der ideale Prozessverlauf.

Dieser Verlauf hat sich aus den Gesprächen mit den Experten herauskristallisiert. Zu Beginn eines Projektes findet die Phase der Anforderungsumsetzung statt. Bevor die Verbundreleasephase beginnt, geben die Teams ihre Leistungen (SW und HW) ab. Es folgt eine Integration aller erbrachten Leistungen im Automobil mit anschließender Testingphase. Die Komponenten werden anschließend im Verbund getestet. Beide Phasen dauern mehrere Monate. Die Testphase soll potentielle Schwachstellen entdecken, die dann den jeweiligen betroffen Abteilungen weitergeleitet werden. In dieser Phase haben die Teams nochmal die Zeit Fehler zu beseitigen, damit die nächste Phase möglichst unbelastet und fehlerlos beginnen kann. Wenn die Fehlerliste komplett abgearbeitet ist kann die nächste Entwicklungsphase beginnen. Nachfolgend werden mit Hilfe der GTM die wichtigsten Kategorien aus den Daten gefiltert.

## 6.3. Anforderungsumsetzung und Featureaufbau

Insgesamt wurden zwei Konzepte, als Herausforderungen in der Entwicklungsphase formuliert, die einer Reihe an weiteren Folgen und Problemen mit sich ziehen.

1. Räumliche Trennung zwischen Schnittstellensysteme (Entwicklung, Lieferanten etc.)
2. Belastung aus vorherigen Projekten bzw. Projektphasen

Eine räumliche Trennung zwischen Schnittstellensysteme (Entwicklungsabteilungen aus unterschiedlichen Fachbereichen, Lieferanten etc.), führt zu einer unzureichenden Kommunikation. So werden Engpässe in vielen Fällen erst spät erkannt, und es entsteht eine versetzte Leistungsabgabe von abhängigen Steuergeräte und Softwarekomponenten. Das hat zur Folge, dass ein Meilenstein, mitsammt all deinen Qualitäts- und Sicherheitsstandards nicht erfüllt wird und eine gemeinsame Testingphase nicht stattfinden kann. Eine weitere Folge sind Nachintegrationen, um entstandene Lücken zu kompensieren. Die Nachintegrationen haben unmittelbare Auswirkungen auf den Verbundrelease und führen zu einer Verschiebung der Releasezeitpunkte der einzelnen Fachabteilungen.

Oft startet ein neues Projekt bzw. eine unmittelbar nachfolgende Projektphase verzögert mit einer sogenannten technischen Schuld. Dies geschieht oftmals, wenn eine unzureichende Ressourcenplanung durchgeführt wurde, oder die geplanten Ressourcen in weiteren, höher priorisierten Projekten oder Bugfixingphasen eingespannt sind. Auch zusätzliche Anforderungen auf Unternehens- und Kundenseite haben eine Prozessverschiebung zur Folge.

## 6.4. Verbundreleasephase

Für die Verbundreleasephase wurden insgesamt drei Konzepte formuliert.

1. Hybride Projektmanagement- bzw. Entwicklungsmethodiken in den SE-Teams
2. Abhängigkeiten zwischen Steuergeräte- und Softwarekomponenten
3. Hohe Sicherheits- und Qualitätsanforderungen im Meilenstein
4. Inhärente Unterschiede zwischen Hardware und Softwarekomponenten

Hybride Organisationsstrukturen hindern oft eine reibungslose Umsetzung, der im Meilenstein geplanten Funktionalitäten und Tests. Die Releasezyklen haben im agilen und klassischen Umfeld ganz andere Frequenzen. So entstehen Reibungen beim verknüpfen von mechatronischen Anforderungen, insbesondere, wenn Abhängigkeiten zwischen den Steuergeräte- und den Softwarekomponenten vorherrschen. Oft können Softwarelieferleistungen nur in Verbindung mit einem Steuergerätes in den Verbundrelease zugelassen werden. Unterschiedliche Fertigstellungszeitpunkte abhängiger Komponenten, die zwangsläufig im Verbund getestet werden müssen führen dazu, dass eine gemeinsame Testphase nicht stattfinden kann. Der zwangsläufige Verbundtest ist durch das sicherheitskritische Umfeld gefordert, in welches das Automobil zu kategorisieren ist. Dort herrscht eine hohe Anforderung an Sicherheit und Qualität, die über das gesamte Fahrzeugprojekt sichergestellt werden muss. So werden auch hier Nachintegrationen als Maßnahme eingeleitet, die den Prozess verzögern. Die hohen Sicherheitsanforderungen führen zudem zu einem hohen bürokratischen Aufwand für die Planung der Meilensteine. Außerdem gibt es inhärente Unterschiede zwischen Steuergeräte- und Softwarekomponenten die einen reibungslosen Verbundrelease ermöglichen. Software kann hochgradig nach agilen Prinzipien entwickelt werden, währen die Steuergeräteentwicklung noch sehr stark an den Entstehungsprozess und die Meilensteinstruktur gebunden ist. Dies schränkt den Grad der Agilität massiv ein.

## 6.5. Testphase

Für die Testphase wurden folgende Konzepte der Herausforderung formuliert:

1. Fehlende oder unzureichende Testinfrastruktur
2. Eingeschränkte Testabdeckung
3. Räumliche Trennung zwischen Testhaus und Entwicklung (Allgemeiner Begriff finden)
4. Mangel an interdisziplinären Teams
5. Unregelmäßigkeiten im Prozess

Die höchste und sicherste Testinstanz in Fahrzeugprojekten ist die reale Fahrzeugumgebung selbst. Zu Beginn eines Projektes steht das fertige Fahrzeug noch nicht zur Verfügung. So werden iterativ und zu jedem Meilenstein Teilsysteme entwickelt, die dann auch als Testumgebung für die jweiligen Softwarekomponenten verwendet werden kann. Wenn also zu Beginn die Testumgebung noch nicht den gewünschten Reifegrad erreicht hat, greift man oft auf Simulationsmodelle wie Hardware in the Loop (HIL), Software in the Loop (SIL), Prozessor in the Loop (PIL), Model in the Loop

(MIL) zurück (siehe 2 - Simulationsmodelle), die eine Testumgebung mit synthetisch generierten Daten und simulierten Komponenten aufspannen. Simulationsmodelle haben je nach Branche einen unterschiedlichen Verlässlichkeitsgrad. In der Automobilbranche werden sie nicht als Endtests gesehen, da man damit nur eine eingeschränkte Testabdeckung erreicht. So muss die Validierung der Komponente zwangsläufig unter realen Fahrzeugbedingungen und im Verbund mit all seinen Schnittstellen stattfinden. Eine räumliche Trennung zwischen Testhaus und Entwicklung führt zu einer fehlenden Kommunikation und Übersicht in den Testingphasen. Alle Inkonsistenzen und Rückmeldungen bezüglich eines Testergebnisses werden häufig nur sehr verzögert an die verantwortlichen Fachbereiche weitergeleitet oder haben nicht die gewünschte Aussagefähigkeit. Parallele Tätigkeiten, wie sie von der opertiven Ebene ausgeführt werden, verflechte sich und der Prozess verzögert sich. Dies hat zur Folge, dass die Qualität, Kosten und Produktivität negativ belastet werden. Hinzukommt, dass die intrinsische Motivation einer Fehlerbehebung nur in den verantworlichen Fachbereichen herrscht, wo die Komponenten entwickelt werden. Die nötigen und gefragten interdisziplinären Fähigkeiten, wie sie in reinen Softwareentwicklungsbranchen vorzufinden sind, wo die Entwickler auch gleichzeitig Tester sind, sind nur beschränkt verfügbar oder fehlen heutzutage. Der Ausbau solcher Fähigkeiten könnte zu einer drastischen Reduzierung der Fehlertickets führen. Nachintegrationen erschweren zudem eine Regelmäßigkeit im Projekt einzuführen. So können gemeinsame Releasezyklen mit Schnittstellensysteme nur schwer geplant werden.

## 6.6. Fehlerabbau- und Featureaufbauphase

Die Fehlerabbauphase wir oft von einer Featureaufbauphase für den nächsten Releases begleitet. Für die Fehlerabbauphase wurden folgende Konzepte entwickelt:

1. Außerplanmäßige Anforderungsumsetzungen
2. Unzureichend kurzer Zeitraum für die Fehleranalyse und Abbauphase
3. Ressourcenmangel (Kosten, Mitarbeiter, Testumgebung)

Eine erfolgreiche Fehlerabbauphase wird dominant von außerplanmäßigen Anforderungsumsetzungen beeinträchtigt. Anforderungen können unterschiedlich motiviert und dementsprechend auch priorisiert werden (Unternehmen-, Prozess- und Kundenanforderungen). So kommt es vor, dass laufende Arbeiten unterbrochen oder vernachlässigt werden müssen, um wichtigere Anforderungen umsetzen zu können. Unzureichend kurze Zeiträume für die Fehleranalyse und Abbauphase führen zudem zu Ressourcen-Engpässe und Prozessverzögerungen. Die zeitliche Einplanung bedarf einer großen Managementerfahrung, sowie einer effizienten und gut ausgestatteten Testumgebung. Die außerplanmäßige Anforderungsumsetzung führt zu Ressourcenmangel in Form von finanzielln Mitteln, Mitarbeiter und Testumgebungen. Testumgebungen sind je nach Bauteilkomplexität, mit einem erheblichen Kostenaufwand verbunden. Mitarbeiter können nicht über einen überzogenen Zeitraum in Projekten eingespannt werden, da Mitarbeiterregelungen wie Urlaubsrecht etc. eingreifen. In Krankheitsfällen müssen die Prozessanforderungen eingreifen, sodass nur eine umfangreiche Dokumentation diesen Ausfall ausgleichen kann. Mitarbeiterwechsel sind auch oft ein Problem, aufgrund der Einarbeitunsphase der neuen Mitarbeiter.

Für die Featureaufbauphase wurden folgende Drucksituationen entdeckt:

1. Managemententscheidungen
2. Einhaltung von Meilensteinterminen
3. Fehlerbehebung
4. Leistungserbringung

Zu den Drucksituationen, die eine Featureaufbauphase verhindern, gehören Managemententscheidungen, die Einhaltung von Meilensteinterminen, die Fehlerbehebung sowie die zu erbringende Leistung. Das Management selbst ist durch den Kunden, sowie von sozialpolitischen Entscheidungen durch die Geschäftsleitung, gesteuert. Zudem wird die Featureaufbauphase von der Fehlerbehebungsphase gebremmst. Die terminliche Fertigstellung von priorisierten Komponenten führt zur Vernachlässigung von laufenden Anforderungsumsetzungen. Alle Faktoren werde von einem großen Leistungsdruck bzgl der zu erbringenden Leistungen begleitet.

Die Fehlerabbau- und Featureaufbauphase laufen oft ineinander ein, aufgrund auch der Verzögerung im Prozess durch Nachintegrationen und Bugfixing Vorgänge. Die Fehlerabbau- und die Featureaufbauphase sind somit kontiuierliche Prozesse die sich bis Abschluss des Projektes wiederholen. Die Meilensteine werden währenddessen begonnen und abgeschlossen.

# 7. Fazit und Ausblick

## 7.1. Fazit

Ein Produktentstehungsprozess spiegelt den aktuellen Stand der Technik und den gesellschaftlichen Zustand in der Branche wieder. Der Produktentstehungsprozess wird meistens sehr direkt von Phänomenen wie der Globalisierung und der Digitalisierung sowie von technologischen Umschwüngen beeinflusst. Er wird eingeführt, um große Organisationen und komplexe Produkte genaustens zu orchestrieren. Der Faktor Mensch und die technischen sowie prozessualen Gegebenheiten, dominieren und lenken diesen hochkomplexen Prozess. In dieser Arbeit wurden mit Hilfe von Interviews mit Experten aus der Automotilbranche (o.ä.), die Herausorderungen in der Releaseplanung, zur reibungslosen Integration der Kernkomponenten eines Fahrzeugs, untersucht. Diese Arbeit baut auf eine zuvor durchgeführte Studie bei der Dr. Ing. h. c. F. Porsche AG auf. Als Datenerhebungsinstrument wird auch hier aus Vergleichsgründen, derselbe Fragebogen verwendet. Insgesamt wurden 10 Experten aus der automotive oder automotive ähnlichen Branche befragt. Die erhobenen Daten zeigen deutlich, dass die unterschiedlichen Projektebenen (Unternehmensleitung, Projektmanagemen und Operative Ebene) in einem ständigen Konflikt zwischen termingerechter Anforderungsumsetzung und den immer kürzer werdenden Produktzyklen leben. Um mit dem Puls des Marktes mitzuhalten, greifen die Unternehmens- und Projektebene oft zu Maßnahmen, die in Extremfällen die erfolgreiche Umsetzung eines Projektes gefährden. So müssen Ressourcen immer wieder umorganisert und Prioritäten neu definiert werden. Dies führt auf operativer Ebene zu Unzulänglichkeiten, die im Laufe des Prozesses entstehen. Die Projektdurchführung in der Automobilbranche ist sehr klassisch durch eine Meilensteinstruktur definiert. Zu terminlich festgelegten Zeitpunkten müssen Software- und Hardwarelieferleistungen implementiert bzw. fertiggestellt sein. Nur so können problemlos alle Schnittstellensysteme validiert werden. Der Plan läuft in der Realität etwas anders ab. Oft beobachtet man, dass die Prozesse verstarren und sich die Welt um sie herum schneller dreht, sodass eine Adaptierungslücke entsteht. Diese Lücke soll dann möglichst schnell gefüllt werden, sodass der Ablauf reibungslos verläuft, um keine Verluste zu generieren. Verzögerte Abgaben, fehlende Ressourcen und Testumgebungen erschaffen einen ineffizienten Prozess. Aufgrund des guten Rufes in der Softwareentwicklung, haben agile Methoden auch in der Automobilbranche Aufmerksamkeit erhalten. Sie werden als fehlendes Puzzlestück gesehen, um die Lücke in den aktuellen Projektmanagementherausforderungen zu füllen. Heutzutage herrscht aber die Situation, dass die agilen Methoden nur in vereinzelten Bereichen, und überwiegend im Softwarebereich, eine Erfolgsgeschichte sind. Bereiche wie die Steuergeräteentwicklung hinken in der Prozessumwandlung hinterher. Vor allem Hardwarekomponenten die einen bestimmten physischen Entwicklungsprozess fordern, der eine andere Wartbarkeitskomplexität besitzt, sind besser ganz klassisch mit einer Meilensteinstruktur zu beherrschen. In Cross-Funktionalen Bereichen, wo eben mehrere Disziplinen (Machanik, Elektronik und Software) vereint werden müssen ist man noch in der Experimentierphase, sodass das Durchsetzungsvermögen von agilen Methoden noch nicht ausgereift ist. Ein wichtiger Treiber solcher Methoden ist der Faktor Mensch und eine klare

Umstrukturierung der übergeordneten Meilensteinzutruktur hin zu einem agilen Ökosytem. Bevor sich die Prozesse jedoch durchsetzen können, müssen die Menschen bereit sein sie zu adaptieren. Es fehlt ein konsequenter Einsatz an Methoden und somit einer Standardisierung und Vereinheitlichung der Prozessmanagementstrukturen, um die Reibungen die im Verbundrelease entstehen auszumerzen.

## 7.2. Ausblick

Mit dieser Studie wurde, aufgrund der geringen Anzahl an Teilnehmern, keine Generalisierbarkeit erreicht.

Sie kann jedoch als Basis für eine ausgedehnte Prozessanalyse verwendet werden, um gezielt die Diskrepanz zwischen Hardware und Software zu erforschen und gemeinsame Prozessmanagementschnittstellen zu schaffen.

Ein Prozess soll einen reibungslosen Ablauf des Projektes sicherstellen. Die positiven Effekte werden nur durch eine Verwendung über einen längeren Zeitraum sichtbar. Dies kann aber nur stattfinden, wenn die Prozesse auch Rückschläge im Zeuge einer Prozessadaption in kauf nehmen können. Erst dann kann ein Stabilisierungsmechanismus einsetzen.

# Literaturverzeichnis


[19] *Real-Life Challenges in Automotive Release Planning*. 2019 (zitiert auf S. 59–66).

[ADS+10] A. Albers, T. Düser, O. Sander, C. Roth, J. Henning. *X-in-the-Loop-Framework für Fahrzeuge, Steuergeräte und Kommunikationssysteme*. Bd. 5. 5. Springer Science und Business Media LLC, Okt. 2010, S. 60–65. DOI: https://doi.org/10.1007/BF03224034 (zitiert auf S. 17, 31, 32).

[AFFR16] D. Ameller, C. Farré, X. Franch, G. Rufian. *A Survey on Software Release Planning Models*. Springer International Publishing, 2016, S. 48–65. DOI: 10.1007/978-3-319-49094-6_4 (zitiert auf S. 39).

[Agi] S. .-.-. S. Agile. URL: https://www.scaledagileframework.com/ (zitiert auf S. 37).

[BBW+16] H.-H. Braess, T. Breitling, J. Weissinger, N. Grawunder, U. Hackenberg, V. Liskowsky, U. Widmann. *Produktentstehungsprozess*. Springer Fachmedien Wiesbaden, 2016, S. 1257–1369. DOI: https://doi.org/10.1007/978-3-658-09528-4_11 (zitiert auf S. 16, 19, 22–24, 26).

[BH17] C. Brandes, M. Heller. *Eine Geschichte voller Missverständnisse*. Springer Fachmedien Wiesbaden, 2017, S. 1–4. ISBN: 3885796449. DOI: 10.1007/978-3-658-18085-0_1 (zitiert auf S. 33).

[BHR05] U. Bestfleisch, J. Herbst, M. Reichert. „Requirements for the Workflow-based Support of Release Management Processes in the Automotive Sector.“ In: 2005 (zitiert auf S. 39).

[BLM+18] M. Bartonitz, V. Lévesque, T. Michl, W. Steinbrecher, C. Vonhof, L. Wagner, Hrsg. *Agile Verwaltung*. Springer Berlin Heidelberg, 2018. DOI: https://doi.org/10.1007/978-3-662-57699-1 (zitiert auf S. 34, 35).

[Böh05] R. Böhm. *Methoden und Techniken der System-Entwicklung*. Vdf Hochschulverlag Ag, 2005. 424 S. ISBN: 978-3-7281-3690-9. DOI: 10.3218/3690-9. URL: https://www.amazon.com/Methoden-und-Techniken-der-System-Entwicklung/dp/3728129569?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3728129569 (zitiert auf S. 21, 22).

[Bor18] J. M. Boris Gloger. *Das Scrum-Prinzip*. Schäffer-Poeschel Verlag, 14. Feb. 2018. 316 S. ISBN: 9783791039473. URL: https://www.ebook.de/de/product/31055412/boris_gloger_juergen_margetich_das_scrum_prinzip.html (zitiert auf S. 17, 33, 35).

[BS12] H.-H. Braess, U. Seiffert, Hrsg. *Vieweg Handbuch Kraftfahrzeugtechnik*. Vieweg+Teubner Verlag, 2012. DOI: 10.1007/978-3-8348-8298-1 (zitiert auf S. 15).

[BS13] H.-H. Braess, U. Seiffert, Hrsg. *Vieweg Handbuch Kraftfahrzeugtechnik*. Springer Fachmedien Wiesbaden, 2013. DOI: 10.1007/978-3-658-01691-3 (zitiert auf S. 17, 19–21, 23, 24, 26).

[CADR18] J. Chen, M. H. Alalfi, T. R. Dean, S. Ramesh. *Modeling AUTOSAR Implementations in Simulink*. Springer International Publishing, 2018, S. 279–292. DOI: https://doi.org/10.1007/978-3-319-92997-2_18 (zitiert auf S. 31).

[CKS03] M. B. Chrissis, M. Konrad, S. Shrum. *CMMI(R): Guidelines for Process Integration and Product Improvement*. Addison-Wesley Professional, 2003. ISBN: 0321154967. URL: https://www.amazon.com/CMMI-Guidelines-Process-Integration-Improvement/dp/0321154967?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321154967 (zitiert auf S. 26).

[DAST12] A. S. Danesh, R. Ahmad, M. R. Saybani, A. Tahir. *Companies Approaches in Software Release Planning – Based on Multiple Case Studies*. Bd. 7. 2. International Academy Publishing (IAP), Feb. 2012. DOI: 10.4304/jsw.7.2.471-478 (zitiert auf S. 39).

[DK01] R. Durst, D. Kabel. *Virtual Teams*. EMERALD GROUP PUB, 11. Dez. 2001, S. 167–214. 258 S. ISBN: 978-0-76230-843-9. DOI: https://doi.org/10.1016/S1572-0977(01)08024-4. URL: https://www.ebook.de/de/product/4904894/virtual_teams.html (zitiert auf S. 26).

[FG13] J. Feldhusen, K.-H. Grote, Hrsg. *Pahl/Beitz Konstruktionslehre*. Springer Berlin Heidelberg, 2013. DOI: https://doi.org/10.1007/978-3-642-29569-0 (zitiert auf S. 22, 29).

[Gün07] W. A. Günthner, Hrsg. *Neue Wege in der Automobillogistik*. Springer Berlin Heidelberg, 2007. DOI: 10.1007/978-3-540-72556-5 (zitiert auf S. 19).

[Hen10] W.-G. B. Henning Wolf. *Agile Softwareentwicklung*. Dpunkt.Verlag GmbH, 1. Okt. 2010. ISBN: 978-3-89864-701-4. URL: https://www.ebook.de/de/product/11130282/henning_wolf_wolf_gideon_bleek_agile_softwareentwicklung.html (zitiert auf S. 34, 35).

[Her14] M. Herczeg. *Prozessführungs-systeme/ Process Control Systems: Sicherheitskritische Mensch-maschine-systeme Und Interaktive Medien Zur Überwachung Und Steuerung ... Safety-critical Human-mac (German Edition)*. Walter de Gruyter, 2014. ISBN: 9783486584455. URL: https://www.amazon.com/Prozessf%C3%BChrungs-systeme-Process-Control-Systems-Mensch-maschine-systeme/dp/3486584456?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3486584456 (zitiert auf S. 17, 21, 22, 28, 31).

[HPLE13] V. T. Heikkilä, M. Paasivaara, C. Lassenius, C. Engblom. „Continuous Release Planning in a Large-Scale Scrum Development Organization at Ericsson“. In: (2013), S. 195–209. DOI: https://doi.org/10.1007/978-3-642-38314-4_14 (zitiert auf S. 39).

[HRJ10] V. Heikkila, K. Rautiainen, S. Jansen. *A Revelatory Case Study on Scaling Agile Release Planning*. IEEE, Sep. 2010. DOI: 10.1109/SEAA.2010.37 (zitiert auf S. 39).

[JK16] H. H. Jung, P. Kraft. *Digital vernetzt. Transformation der Wertschöpfung*. Hrsg. von H. H. Jung, P. Kraft. Hanser Fachbuchverlag, 5. Dez. 2016. 432 S. ISBN: 978-3-446-44780-6. DOI: https://doi.org/10.3139/9783446449466. URL: https://www.ebook.de/de/product/26183754/digital_vernetzt_transformation_der_wertschoepfung.html (zitiert auf S. 22).

[KBOK17] T. Karvonen, W. Behutiye, M. Oivo, P. Kuvaja. *Systematic literature review on the impacts of agile release engineering practices*. Bd. 86. Elsevier BV, Juni 2017, S. 87–100. DOI: 10.1016/j.infsof.2017.01.009 (zitiert auf S. 39).

[KDM+19] M. Kuhrmann, P. Diebold, J. Munch, P. Tell, K. Trektere, F. McCaffery, V. Garousi, M. Felderer, O. Linssen, E. Hanser, C. R. Prause. *Hybrid Software Development Approaches in Practice: A European Perspective*. Bd. 36. 4. Institute of Electrical und Electronics Engineers (IEEE), Juli 2019, S. 20–31. DOI: 10.1109/MS.2018.110161245 (zitiert auf S. 39).

[KHF+17] J. Klünder, P. Hohl, M. Fazal-Baqaie, S. Krusche, S. Küpper, O. Linssen, C. R. Prause. *HELENA Study: Reasons for Combining Agile and Traditional Software Development Approaches in German Companies*. Springer International Publishing, 2017, S. 428–434. DOI: 10.1007/978-3-319-69926-4_32 (zitiert auf S. 39).

[KLP03] S. Köhl, D. Lemp, M. Plöger. *Steuergeräte-Verbundtests mittels Hardware-in-the-Loop-Simulation*. Bd. 105. 10. Springer Science und Business Media LLC, Okt. 2003, S. 948–955. DOI: https://doi.org/10.1007/BF03221590 (zitiert auf S. 15).

[KNO12] P. Kruchten, R. L. Nord, I. Ozkaya. *Technical Debt: From Metaphor to Theory and Practice*. Bd. 29. 6. Institute of Electrical und Electronics Engineers (IEEE), Nov. 2012, S. 18–21. DOI: 10.1109/MS.2012.167 (zitiert auf S. 26, 27).

[LLNW08] M. Lindgren, R. Land, C. Norstr, A. Wall. *Key Aspects of Software Release Planning in Industry*. IEEE, März 2008. DOI: 10.1109/ASWEC.2008.4483220 (zitiert auf S. 39).

[LR09] M. Lloyd, P. Reeve. „IEC 61508 and IEC 61511 assessments - some lessons learned“. In: *4th IET International Conference on Systems Safety 2009. Incorporating the SaRS Annual Conference*. IET, 2009. DOI: 10.1049/cp.2009.1540 (zitiert auf S. 28).

[Luí18] B. S. Luís Ferreira Pires Slimane Hammoudi. *Model-Driven Engineering and Software Development*. Hrsg. von L. F. Pires, S. Hammoudi, B. Selic. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-94764-8 (zitiert auf S. 31).

[MHDZ08] M. Mueller, K. Hoermann, L. Dittmann, J. Zimmer. *Automotive SPICE in Practice: Surviving Implementation and Assessment (Rockynook Computing)*. Rocky Nook, 2008. ISBN: 9781933952291. URL: https://www.amazon.com/Automotive-SPICE-Practice-Implementation-Assessment-ebook/dp/B00VB467GU?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B00VB467GU (zitiert auf S. 26).

[MHHR06] D. Müller, J. Herbst, M. Hammori, M. Reichert. *IT Support for Release Management Processes in the Automotive Industry*. Springer Berlin Heidelberg, 2006, S. 368–377. DOI: https://doi.org/10.1007/11841760_26 (zitiert auf S. 39).

[MM11a] G. Mey, K. Mruck, Hrsg. *Grounded Theory Reader*. VS Verlag für Sozialwissenschaften, 2011. DOI: 10.1007/978-3-531-93318-4 (zitiert auf S. 41, 42).

[MM11b] G. Mey, K. Mruck. *Grounded-Theory-Methodologie: Entwicklung, Stand, Perspektiven*. VS Verlag für Sozialwissenschaften, 2011, S. 11–48. DOI: https://doi.org/10.1007/978-3-531-93318-4_1 (zitiert auf S. 41).

[MSG07] H. E. Mößmer, M. Schedlbauer, W. A. Günthner. *Die automobile Welt im Umbruch*. Springer Berlin Heidelberg, 25. Okt. 2007, S. 3–15. ISBN: 978-3-540-72556-5. URL: https://www.ebook.de/de/product/11430111/neue_wege_in_der_automobillogistik.html (zitiert auf S. 20).

[PCCW93] M. Paulk, B. Curtis, M. Chrissis, C. Weber. *Capability maturity model, version 1.1*. Bd. 10. 4. Institute of Electrical und Electronics Engineers (IEEE), Juli 1993, S. 18–27. DOI: 10.1109/52.219617 (zitiert auf S. 25).

[PL11] J. Ponn, U. Lindemann. *Konzeptentwicklung und Gestaltung technischer Produkte*. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-20580-4 (zitiert auf S. 29).

[Rei14] K. Reif. *Automobilelektronik*. Springer Fachmedien Wiesbaden, 2014. DOI: 10.1007/978-3-658-05048-1 (zitiert auf S. 22).

[Roy87] W. W. Royce. *Managing the development of large software systems: concepts and techniques*. IEEE Computer Society, 1987. ISBN: 0-89791-216-0. URL: https://www.amazon.com/International-Engineering-INTERNATIONAL-ENGINEERING-PROCEEDINGS/dp/0897912160?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0897912160 (zitiert auf S. 21).

[Sch05] P. Scholz. *Softwareentwicklung eingebetteter Systeme*. Springer Berlin Heidelberg, 2005. ISBN: 9783899583885. DOI: 10.1007/3-540-27522-3 (zitiert auf S. 17, 22, 28).

[Sch14] M. D. Schulz. *Der Produktentstehungsprozess in der Automobilindustrie*. Springer Fachmedien Wiesbaden, 2014. DOI: https://doi.org/10.1007/978-3-658-06464-8 (zitiert auf S. 17, 20).

[SRGK17] E. Sax, R. Reussner, H. Guissouma, H. Klare. *A Survey on the State and Future of Automotive Software Release and Configuration Management*. en. Karlsruhe, 2017. DOI: 10.5445/IR/1000075673 (zitiert auf S. 39).

[Sto14] R. Stobbe. *Frühe Vorgehensmodelle*. GRIN Publishing, 27. Juni 2014. 20 S. ISBN: 978-3656675655. URL: https://www.ebook.de/de/product/22567394/rochus_stobbe_fruehe_vorgehensmodelle.html (zitiert auf S. 22).

[SZ16] J. Schäuffele, T. Zurawka. *Automotive Software Engineering*. Springer Fachmedien Wiesbaden, 2016. DOI: https://doi.org/10.1007/978-3-658-11815-0 (zitiert auf S. 15, 29).

[Tie03] O. Tietze. *Strategische Positionierung in der Automobilbranche*. Deutscher Universitätsverlag, 2003. DOI: 10.1007/978-3-322-81642-9 (zitiert auf S. 20, 21).

[Tie14] Tiemeyer. *IT-Projektmanagement*. Carl Hanser Verlag GmbH & Co, 2014. ISBN: 3446440747. URL: https://www.amazon.com/-Projektmanagement-2-Tiemeyer/dp/3446440747?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3446440747 (zitiert auf S. 34).

[WD08] K. W. Wagner, W. Dürr. *Reifegrad nach ISO/IEC 15504 (SPiCE) ermitteln*. Carl Hanser Verlag, Nov. 2008. ISBN: 978-3-446-40721-3. DOI: https://doi.org/10.3139/9783446418998. URL: https://www.amazon.com/Nach-ISO-IEC-15504-erm/dp/3446407219?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3446407219 (zitiert auf S. 26).

[Wei06] N. Weiss. *Systematische Prozessoptimierung*. Wissenschaft & Praxis, 1. Nov. 2006. 60 S. ISBN: 9783896442659. URL: https://www.ebook.de/de/product/7764802/norbert_weiss_systematische_prozessoptimierung.html (zitiert auf S. 21).

[Wes03] E. Westkaemper. *Wandlungsfähige Unternehmens-strukturen für variantenreiche Serienproduktion*. Springer Berlin Heidelberg, 2003, S. 95–108. DOI: https://doi.org/10.1007/978-3-642-55495-7_9 (zitiert auf S. 21).

[WEW15] U. Wilhelm, S. Ebel, A. Weitzel. *Funktionale Sicherheit und ISO 26262*. Springer Fachmedien Wiesbaden, 2015, S. 85–103. DOI: https://doi.org/10.1007/978-3-658-05734-3_6 (zitiert auf S. 28, 29).

[Wol18] F. Wolf. *Fahrzeuginformatik*. Springer Fachmedien Wiesbaden, 2018. DOI: https://doi.org/10.1007/978-3-658-21224-7 (zitiert auf S. 17, 21, 23–25, 28, 30, 34).

[Zan09] J. Zander-Nowicka. *Model-based Testing of Real-Time Embedded Systems in the Automotive Domain*. Fraunhofer IRB Verlag, 2009. ISBN: 978-3-8167-7974-2. URL: https://www.amazon.com/Model-based-Testing-Real-Time-Embedded-Automotive/dp/3816779743?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3816779743 (zitiert auf S. 17, 23).

Alle URLs wurden zuletzt am 20. 08. 2019 geprüft.

# A. Fragebogen

In diesem Kapitel wird das Datenerhebungsinstrument vorgestellt.

Context

1. In welcher Rolle sind sie aktuell tätig?
2. Wie lange sind sie in dieser Rolle tätig?
3. An was arbeiten Sie in Ihrem Projekt?
4. In welchen (Produktions-)Bereich würden sie Ihr Projekt einordnen?
5. Welche Entwicklungs- bzw. Projektmanagement Methodik verwenden Sie?

Prozesssystematik: Verbundrelease

1. Wie beurteilen Sie die aktuelle Anzahl an Verbundreleases über die gesamte Laufzeit des Fahrzeugprojektes?
2. Die aktuelle Dauer des Verbundreleases ist
3. Wie oft können abgabereife Software-Lieferleistungen für einen Verbundrelease erbracht werden
4. Erhalten sie rechtzeitig Rückmeldungen aus dem Verbundrelease?
5. Wie oft sollte ein Verbundrelease stattfinden, damit Sie immer einen abgabereifen Softwarestand liefern können?
6. Wären zusätzliche Verbundreleases in Form von Teilverbünden mit reduziertem Testumfang zur Absicherung abhängiger Steuergeräte/Komponenten hilfreich?

Prozesssystematik: Planung

1. Ist eine Initialaufplanung der Inhalte zu Projektstart für die gesamte Laufzeit sinnvoll und möglich?
2. Wie oft ist der Inhalt der initialen Planung zu Beginn eines Verbundreleases noch aktuell?
3. Eine einheitliche Releaseplanung (Funktionen und Komponenten) mit dem Lieferanten/der umzusetzenden Einheit wird erstellt und auch gelebt?
4. Wie schwer ist es Planungsinformationen von den relevanten Ansprechpartnern zu bekommen?
5. Meine Rolle und meine Aufgaben hinsichtlich des Planungsprozesses für mein Projekt sind mir klar bzw. einheitlich dokumentiert

6. Wie stark haben Managemententscheidungen äußere Einflussfaktoren bzw. fremdbestimmte Entscheidungen Einfluss auf Ihren Entwicklungsverlauf?

Prozesssystematik: Integration

1. Wie stark beeinflusst das Bugfixing nach einem Verbundrelease die termingerechte Umsetzung der für den nächsten Verbundrelease geplanten Funktionalitäten?
2. Wie oft werden notgedrungen qualitativ oder inhaltlich suboptimale Softwarestände in den Verbundrelease abgegeben. (wie oft wird müll abgegeben)
3. Welche Aktivitäten dominieren Ihren Arbeitsalltag während eines Verbundreleases?
4. Nachintegrationen sind oft notwendig, da der ursprünglich geplante VR oft nicht ausreicht
5. Nachintegrationen sind sinnvoll, da der ursprünglich geplante Verbundrelease ost nicht ausreicht
6. Was sind Gründe und Ursachen für Nachintegrationen?

Entwicklungsmethodik: Abstimmung

1. Ist der aktuelle Entwicklungsstand für Sie zu jedem Zeitpunkt transparent? (Soll-Ist-Vergleich)
2. Ist der Entwicklungsstand anderer Beteiligter für Sie zu jedem Zeitpunkt ausreichend transparent?
3. Wie wichtig ist die Transparenz des Entwicklungsstands anderer Beteiligter für Sie?
4. Inwiefern kann eine fehlende Transparenz des Entwicklungsstandes von Prozessbeteiligten das Entwicklungsvorhaben eines anderen Beteiligten beeinflussen.
5. Wie stark beeinflusst Sie eine fehlende Transparenz des Entwicklungsstandes anderer Beteiligter bei Ihrem Entwicklugsvorhaben?
6. Wie bewerten Sie die Kommunikation und Abstimmung in den folgend genannten Bereichen?
7. Ansprechpartner/Schnittstellen bekannt / Qualität der Abstimmung ist gut
   a) Innerhalb des Teams
   b) Zwischen Entwicklung und Testhaus
   c) Innerhalb ihres Unternehmens Zu externen Lieferanten

Entwicklungsmethodik: Testen

1. Aussagen bewerten: trifft nicht zu, trifft eher nicht zu, trifft eher zu, trifft zu
2. Menge der Fehler Tickets ist für Fehlermanagement unbeherrschbar/beherrschbar
3. Menge der Fehlertickets ist für die Entwicklung nicht mehr beherrschbar
4. Gestiegene Anzahl an Fehlertickets führt zu Problemen in der Planung und Umsetzung des nächsten Verbundreleases
5. Was sind Gründe für die hohe Anzahl an Fehlertickets?

6. Müssen zu jedem Verbundrelease alle geplanten Änderungen am Steuergeräteverbund vollumfänglich getestet werden?
7. Wann müssen alle Steuergeräte vollumfänglich getestet werden?
8. Müssen zu jedem Verbundrelease für alle Steuergeräte alle Arten an Tests durchgeführt werden?

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift