Die Evolution des SiL – Ansätze zur gesamtheitlichen Absicherung

Felix Strauß, Alexander Merkel, Christian Menzel

Electronics and Virtual Testing Solutions
Bertrandt Group
Birkensee 1
71139 Ehningen
felix.strauss@bertrandt.com
alexander.merkel@bertrandt.com
christian.menzel@bertrandt.com

Abstract: Software-in-the-Loop (SiL) ist Stand der Technik bei der Absicherung von kompiliertem Funktionscode von Steuergeräten in der frühen Entwicklungsphase. Die Ergebnisse der SiL-Tests dienen bisweilen vor allem den entwickelnden Softwareabteilungen der Teilfunktionen, werden jedoch nicht immer konsequent in die gesamtheitlich zu entwickelnden Kundenfunktion eingebracht. Dieser Beitrag thematisiert mögliche Gründe hierfür und stellt in Aussicht, wie SiL in Zukunft einen Beitrag zu einer durchgängigen (End-to-End) Absicherung von Fahrzeugfunktionen leisten kann. Hierfür werden technologische Fortschritte der jüngsten Vergangenheit beleuchtet sowie eine erste Umsetzung gezeigt. Ebenso sollen grundlegende Prozesse der Absicherung besprochen und kritisch bewertet werden.

1 Motivation

Seit einigen Jahren befindet sich die Automobilindustrie in einem Umbruch:

- Der Trend zu autonomen Fahrfunktionen macht das Fahrzeug hinsichtlich der Softwarefunktionalität zu einem völlig anderen Produkt als noch vor fünf bis zehn Jahren.
- Kundenwünsche, vor allem getrieben aus der Smartphone-Welt, führen zum Einzug der Digitalisierung in die Fahrzeugentwicklung und erfordern neue Ansätze für Fahrzeugarchitekturen.
- Der Einstieg in die Elektromobilität erfordert in der Regel komplette Neuentwicklungen der antriebsstrangseitigen Soft- und Hardware.

All diese Trends und Entwicklungen zahlen letzten Endes in eine Zunahme der Komplexität auf allen Ebenen ein: Die Steuergeräte werden leistungsstärker und vernetzter, die Software wird umfangreicher, verteilter und lässt die Zeilen an notwendigem Code sprichwörtlich explodieren.

Es ist immer häufiger zu beobachten, dass die Absicherung der hieraus resultierenden Geschwindigkeit und Quantität nicht Stand zu halten scheint. Ebenso entsteht der Eindruck, dass parallel zu dem beschriebenen Umbruch ein schrittweises Überdenken etablierter Prozesse stattfindet. Dieser Vortrag soll einen Beitrag zu zukünftigen Absicherungsstrategien leisten, indem er den Fokus auf das SiL-Framework lenkt und versucht anhand seiner Evolution zukünftige Potenziale abzuleiten und Lösungswege aufzuzeigen.

2 Kurze Historie des SiLs

Vor der Millenniumwende war die Automobilentwicklung noch stark getrieben von Sensor-/Aktuator-Funktionen mit geringer Komplexität entlang der Software. In den frühen 2000er-Jahren fing die Anzahl der Steuergeräte pro Fahrzeug sowie die Komplexität der Funktionen (vor allem im Bereich Antriebsstrang) deutlich zu steigen, wie Abbildung 1 zeigt. Für Funktionsentwickler wird es zunehmend herausfordernder Code effizient zu integrieren und zu testen. Mit Firmengründungen wie der von QTronic (2006) [PEG19] und PikeTech (2007) [PIK22] treten erste kommerzielle Produkte auf den Markt, welche den Fokus auf die Absicherung der Software legen. SiL etabliert sich vor allem im Softwaremodultest, direkt in den Entwicklungsbereichen. Durch die Etablierung von Entwicklungsstandards, wie dem ersten grundlegenden AUTOSAR-Standard im Jahr 2006 [AUT22] und dem ersten FMI-Standard im Jahr 2011 [FMI22] gewinnt die Technologie weiter an Bedeutung. Die Firma dSPACE bringt die erste Version des Tools VEOS im Jahr 2012 [DSP22] auf den Markt. Über die letzten Jahre wird SiL nun auch vermehrt für den Softwareintegrations- und Softwarequalifikationstest eingesetzt.

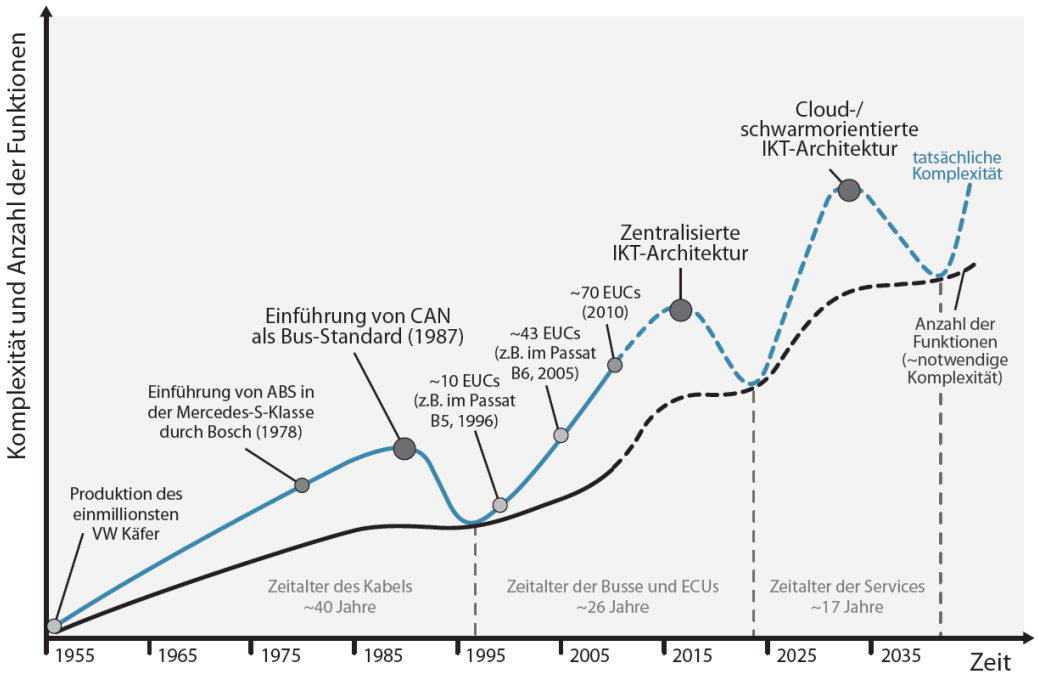

Abbildung 1: Komplexität und Anzahl Funktionen im Fahrzeug [BUC11]

3 Stand der Technik

3.1 Das SiL-Framework

Software-in-the-Loop wird grundsätzlich eingesetzt um sogenannten „Target-code“, in aller Regel auf der Basis von C, zu testen. Der Code wird auf Windows- oder Linux-PCs ausgeführt, nicht jedoch in für die Zielhardware (Mikrocontroller) kompilierter Form. Das „System under Test“ (SuT) wird häufig als virtuelles Steuergerät (vECU) bezeichnet. Dieses wird in eine Simulation eingebracht, welche eine geschlossene Regelschleife abbildet, sowie Verhaltensmodelle der weiteren Komponenten simuliert. Je nach Ausprägung kann in einem SiL nur eine einzelne Softwarekomponente abgebildet werden, jedoch auch verteile Fahrzeugfunktionen, welche über mehrere Steuergeräte verteilt sind.

Zur Einordnung des Reifegrads hilft die folgende Klassifizierung von virtuellen Steuergeräten aus [PRO20], siehe auch Abbildung 2:

- Level-0: Regler Modell
- Level-1: Einfache vECU
- Level-2: Erweiterte vECU
- Level-3: MCAL-vECU
- Level-4: Binary Target vECU

Die Ansätze für Level-0 und Level-1 vECUs sind seit den frühen 2000ern weit verbreitet und finden vor allem im Modultest Anwendung. Marktreife Verbreitung von Level-2 vECUs wurde vermutlich erst rund um die 2010er Jahre erreicht. Level-3 vECUs wurden bis dato eher selten eingesetzt. Über die letzten Jahre wurde jedoch vermehrt daran gearbeitet, den Aufwand für die Generierung von Level-2 und Level-3 vECUs zu optimieren. Maßgeblichen Einfluss hierauf hat die breitere Akzeptanz des AUTOSAR-Standards in der Steuergerätentwicklung, welche die Generierung von vECUs deutlich vereinfacht. Ebenso wurden Herausforderungen hinsichtlich Performance in PC-Umgebungen gelöst. Nichtsdestotrotz lässt der breite Einsatz von Level-3 vECUs noch auf sich warten. Erst über die letzten zwei Jahre scheint es hierfür vermehrt Bestrebungen zu geben. Level-4 vECUs haben, bedingt durch die notwendige Emulation der Zielplattform, einen sehr hohen Ressourcenbedarf und finden noch eher selten Anwendung.

Diverse Toolhersteller haben sich in den letzten Jahren darauf spezialisiert die Erstellung von vECUs zu ermöglichen. Hierzu gehören beispielsweise Synopsys (ehem. QTronic), dSPACE, Vector, ETAS und einige weitere. Der Aufwand für den Erstellungsprozess variiert zwischen den einzelnen Tools, je nach Anwendungsfall und vorhandenen Artefakten, teilweise sehr. Darüber hinaus wächst der Aufwand für die Erstellung mit dem Level der vECU zusätzlich deutlich.

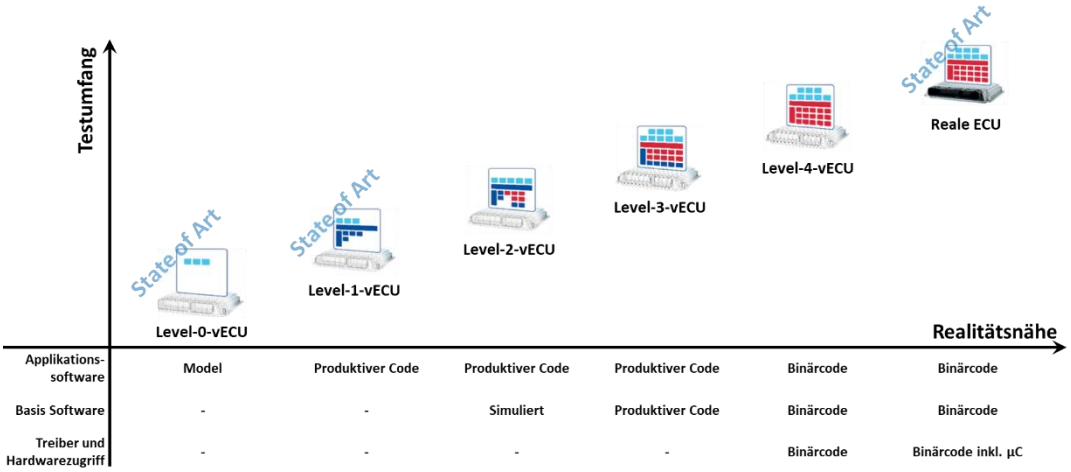

Abbildung 2: Klassifizierung von vECUs [PRO20]

3.2 Der Entwicklungsprozess gemäß V-Modell

Das V-Modell hat sich als gängiges Vorgehensmodell in der Automobilentwicklung durchgesetzt. Es stellt eine Erweiterung zum Wasserfallmodell dar, welches die Softwareentwicklung über die zweite Hälfte des 20. Jahrhunderts geprägt hat. Im Bewertungsmodell „Automotive SPICE“ (ASPICE), welches seit 2006 vom VDA als Entwicklungsgrundlage empfohlen wird, findet es grundlegende Anwendung. Das V- sowie das Wasserfallmodell haben über die Zeit bei Softwareentwicklern für einige Diskussionen und Kritik gesorgt. Einige der Aspekte werden im Folgenden aufgegriffen und in den heutigen Kontext gesetzt.

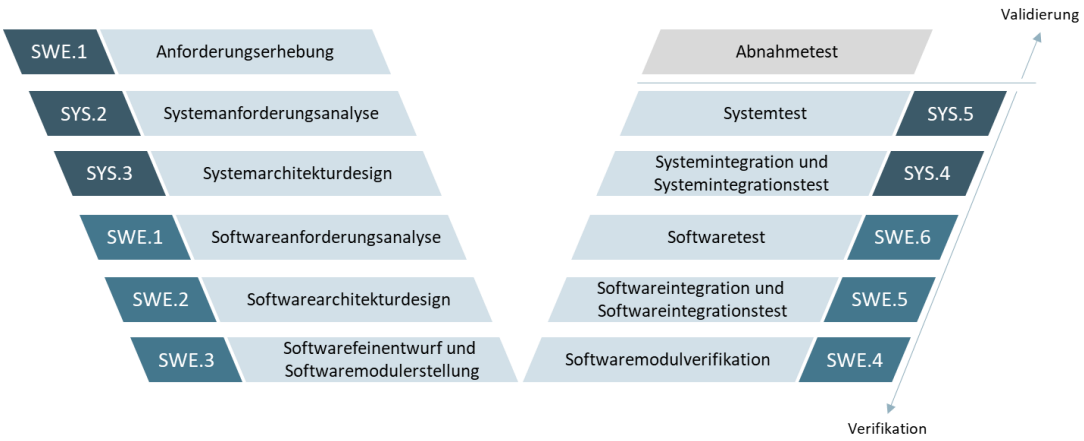

Abbildung 3: V-Modell / ASPICE

Das V-Modell schafft Zuständig- und Verbindlichkeiten: Durch seine Unterteilung in Spezifikations-, Realisierungs- und Testphase sind klare Bereiche, mit definierten Meilensteinen und Artefakten abgesteckt. Die Aufgliederung während der Spezifikations- und Testphase, nach dem Top-Down-Prinzip auf der rechten und nach dem Bottom-Up-Prinzip auf der linken Seite sind sehr eingängig und erlauben es in der Praxis jedem Schritt die notwendige Gründlichkeit beizumessen. Die 1:1-Gegenüberstellung der einzelnen Stufen sollen zu einer möglichst vollständigen Testabdeckung führen.

Durch seine eindeutige Trennung in einzelne Phasen inklusive mehrerer Detaillierungen schafft das V-Modell zwar erfolgreich Zuständigkeiten, jedoch bringt es damit auch unweigerlich Trennlinien und Abhängigkeiten in den Entwicklungsprozess, welche ihn mitunter verlangsamen und unflexibel machen können. Es birgt das Risiko, dass einzelne Bereiche Zuständigkeiten von sich weisen und nicht immer im Sinne des Endprodukts gearbeitet wird. Teilweise wird auf fehlende Ergebnisse der Vorgängerstufe gewartet und der Gesamtprozess verlangsamt. In Analogie zum Wasserfall kann außerdem das unten angekommene Wasser nicht einfach oben wieder in den Wasserfall zurückfließen. Somit leuchtet ein, dass das Modell nicht sonderlich gut für Iterationen in den Anforderungen geeignet ist. Durch seine Linearität ist es zunächst auch nicht vorgesehen aus einer fortgeschrittenen Stufe in eine vorangegangene zurückzuspringen, oder ggf. auch eine Stufe zu überspringen. Fehler werden dadurch teilweise zu spät oder mehrfach gefunden. Manchmal können Fehler in höheren Stufen auch deutlich schlechter nachvollzogen werden als in den unteren, da tiefen Einblicken ins System nicht mehr möglich sind. Aufgrund der zuvor genannten Nachteile werden in der modernen Softwareentwicklung in der Regel agilere Ansätze wie DevOps eingesetzt.

3.3 Aufgliederung des Testprozesses

Wie beschrieben, definieren sowohl das V-Modell als auch ASPICE einen mehrstufigen Testprozess auf der rechten Modellseite, wobei jede Stufe eine 1:1-Beziehung zur linken Seite hat. Im Folgenden soll nun der Testprozess vereinfacht aufgegliedert und mit den häufig genutzten Test-Methoden verknüpft werden:

Die unteren Stufen werden durch den Softwaremodultest (SWE.4), den Softwareintegrationstest (SWE.5) und den Softwarequalifikationstest (SWE.6) beschrieben. Hier findet vor allem der Komponenten-SiL Anwendung. Beim Softwareintegrationstest werden teilweise aber auch Processor-in-the-Loop (PiL) und Hardware-in-the-Loop (HiL) -Systeme eingesetzt. Ebenso findet HiL Anwendung beim Softwarequalifikationstest. Während der anschließenden Systemintegration (SYS.4) und dem zugehörigen Test werden bei Systemlieferanten in der Regel Komponenten-HiLs eingesetzt, bei OEMs auch schon vernetze HiL-Systeme. Der Systemtest (SYS.5) wird dann im Allgemeinen auf Fahrzeug-HiLs oder vernetzen HiLs durchgeführt. Auch Entwicklungsfahrzeuge oder Fahrzeugnachbauten finden beim Systemtest verbreitet Anwendung.

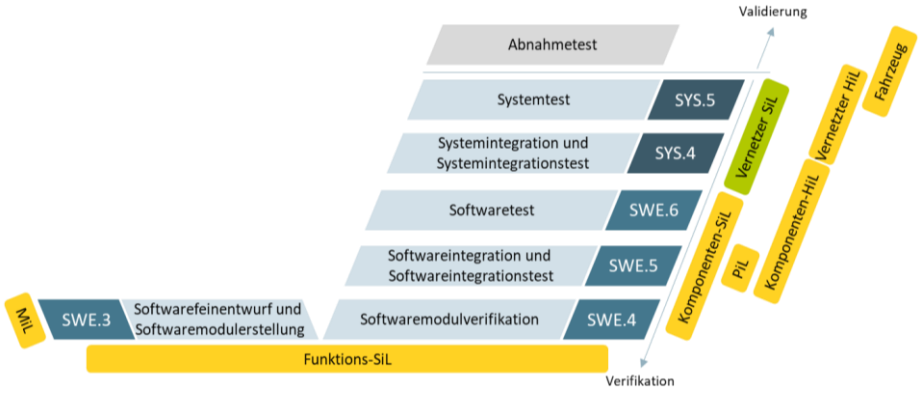

Abbildung 4: Testplattformen im V-Modell/ASPICE

4 SiL im heutigen Testprozess

Als Fazit aus dem vorherigen Kapitel kann gezogen werden, dass der SiL ausschließlich in SWE.4 bis SWE.6 Einsatz findet. Häufig wird hier eine SiL-Umgebung aufgebaut, welche nur ein Steuergerät im Fokus hat. Die Systemintegration beginnt, bedingt durch die Verteilung der vielen Services auf unterschiedliche Steuergeräte, somit erst, sobald eine Vernetzung der Steuergeräte mit Hardware möglich ist. In aller Regel kann damit erst gestartet werden, sobald genügend Hardware in ausreichender Güte vorhanden ist. Dies führt häufig zu Verzögerungen. Im folgenden Abschnitt soll nun skizziert werden, wie die SiL-Technologie durch den Aufbau eines vernetzten SiLs sowie eine flexiblere Ausgestaltung der Testprozesse zu einer Beschleunigung führen kann.

5 Zukünftige Potentiale

5.1 Der vernetze SiL

Wie schon in Abschnitt 3.1 diskutiert, wurden in den letzten Jahren einige technologische Fortschritte bei der Virtualisierung von Basissoftware und damit bei den Level-2 und Level-3 vECUs gemacht. Ebenso ist die erforderliche Performance sowohl im Desktop-PC-Bereich als auch bei diversen Cloud-Anbietern gestiegen und erschwinglicher geworden. Die Erweiterung des FMI-Standards sowie die Entwicklung von leistungsstarken und hochfunktionalen Middlewares ermöglicht es nun zudem sinnvolle Umgebungen mittels Toolkopplung, bzw. Co-Simulation aufzubauen. Dieser Ansatz hat den Vorteil, dass ein großer Teil der funktionalen Fahrzeugwirkkette inkl. aller Vernetzungen virtuell abgebildet werden kann. Somit beschränkt sich der SiL nicht mehr nur auf die Abbildung einzelner Teilsysteme oder virtueller Bauteile, sondern ein virtuelles Fahrzeug bzw. ein digitaler Zwilling kommt in Reichweite.

Bertrandt arbeitet mit einem deutschen Premium-OEM an einer SiL-Plattform, welche diese Aspekte nutzt und dadurch eine virtuelle End-to-End Absicherung bereits auf den Stufen SYS.4 und SYS.5 ermöglicht. Durch den im Vergleich zur Hardware stark wachsenden Software-Anteil kann ein beachtlicher Teil der Funktion bereits mit einer solchen Plattform abgesichert werden. Selbstverständlich muss diese aber nach wie vor durch Tests an echter Hardware validiert werden.

5.2 Toolkopplung durch Middleware

In einem ersten Schritt wurde das Konzept für eine hochskalierbare Co-Simulationsplattform erarbeitet. Dieses wurde im Rahmen eines Proof-of-Concept umgesetzt und wird im Laufe des Jahreswechsels 2022/23 in eine Cloudumgebung für den produktive Einsatz überführt.

Kernbestandteile der Proof-of-Concepts sind

- die kommerziellen Simulationstools, welche als Laufzeitumgebung für virtuelle Steuergeräte von Level-1 bis Level-3 dienen,
- die Open-Source Middleware „SiL-Kit“ der Firma Vector, welche zur Verbindung der Simulatoren auf Bus- und/oder Signalebene dient sowie die Synchronisierung der Simulatoren übernimmt,
- eine kommerzielle Testautomatisierung zur Ausführung von Testfällen sowie
- eine von Bertrandt implementierte Steuerinstanz, welche die Orchestrierung der Simulation übernimmt und eine API für das Testmanagement abstrahiert.

Die SiL-Architektur dieser Co-Simulation ist in Abbildung 5 dargestellt.

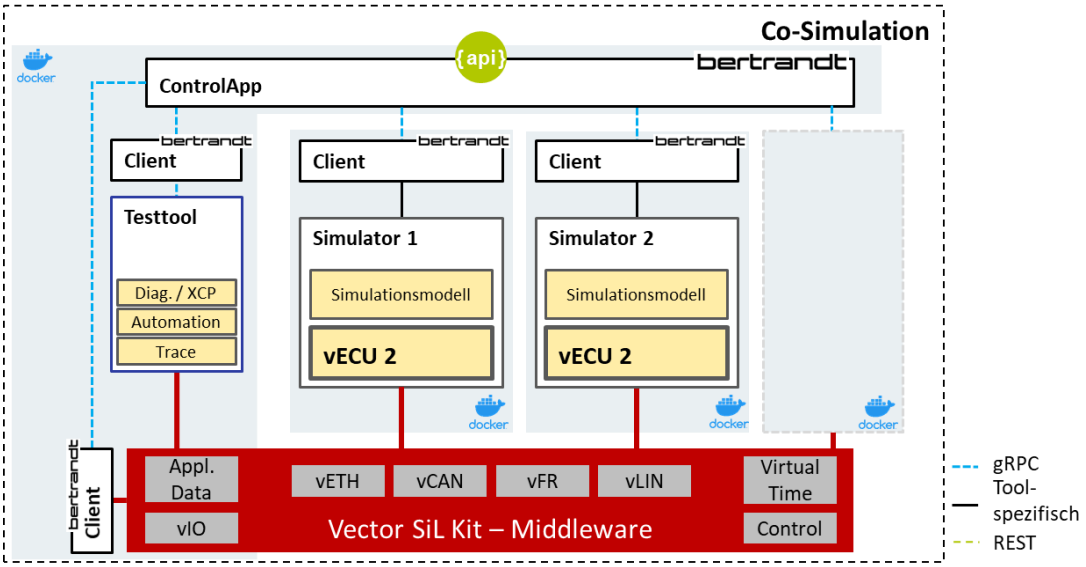

Abbildung 5: Architektur der Co-Simulation

Alle Komponenten sind auf gängigen Desktop-PCs benutzbar. Außerdem sind alle Komponenten als containerisierte Version umgesetzt und die Kommunikation von Middleware und Orchestrierung ist über TCP/IP, bzw. UDP abgebildet. Durch diesen Ansatz wird die Plattform beliebig in die Tiefe (d.h. Hinzufügen weiterer Simulatoren) als auch in die Breite (d.h. Replizieren der Plattform durch Container-Technologie) in der Cloud skalierbar.

5.3 SiL kann mehr als Front-Loading

Wie bereits thematisiert, wurde SiL in der Vergangenheit häufig mit „Front-Loading“ motiviert. Durch die neuen beschriebenen Möglichkeiten des vernetzten SiLs kann dieser nun jedoch als zusätzliche Ergänzung in allen Phasen der Entwicklung gesehen werden. Zwei Szenarien sollen hier näher betrachtet werden:

- **Worksplit HiL – SiL**: Ein vernetzter SiL mit Level-3 vECUs ist in der Lage sehr ähnliche funktionale Aussagen wie ein vernetzter HiL zu treffen, da bereits die gesamte Software in produktiver Form vorhanden ist. Es kann Sinn machen, dass die beiden Methoden HiL und SiL gemeinsam zur Erbringung von Labor- bzw. Simulationsergebnissen verwendet werden. Beispielsweise können Variantentests durch geschicktes Verteilen auf einen HiL sowie einen SiL parallelisiert werden. Der HiL testet hier die maximale Fahrzeugausstattung. Der SiL bestätigt die Ergebnisse noch für die kleineren Varianten. Nur bei Abweichungen muss mit der Zielhardware nachgetestet werden. Des Weiteren könnte an einem HiL-System die Testtiefe eher in Richtung von Performance und elektrischen Fehlern gesetzt werden, an einem ergänzenden SiL eher in die Richtung der Funktionalität.

- **Kopplung HiL – SiL**: Ein weiteres Szenario stellt die Kopplung der beiden Systeme dar. Nicht immer sind alle virtuellen, aber auch realen Steuergeräte zu jedem Zeitpunkt im Entwicklungsprozess verfügbar. Hierzu können Level-3 vECUs mit den realen Bussystemen eines HiL-Prüfstands gekoppelt werden. Somit kann der SiL um eine reale Komponente, bspw. ein Display, ergänzt werden. Alternativ kann an einem HiL eine (noch) nicht verfügbare reale Komponente durch ein virtuelles Steuergerät ersetzt werden, statt sie durch eine Restbussimulation abzubilden. Ein Aufbau, der diesen Anwendungsfall darstellt, wurde bereits im Rahmen der Co-Simulationsplattform von Bertrandt umgesetzt und erprobt. Zu beachten ist hierbei, dass alle Komponenten echtzeitfähig sein müssen.

5.4 Die Rolle des Testmanagements

Eine zentrale Rolle für den Erfolg des diskutierten Ansatzes wird das Testmanagement übernehmen.

Essenziell ist hier vor allem, zu entscheiden, welche Testfälle an einem vernetzten SiL zu aussagekräftigen Ergebnissen führen und welche Testfälle auf jeden Fall an einer höheren Instanz wie einem HiL-Prüfstand durchgeführt werden müssen. Wird heute noch häufig das Testmanagement nur separat auf jeder Integrationsstufe durchgeführt, sollte es in Zukunft übergreifender eingeführt und gelebt werden. Hierdurch kann sichergestellt werden, dass Fehler nicht mehrfach oder zu spät gefunden werden. Des Weiteren kann die Auslastung der hardwarebasierten Instanzen durch geschickte Auswahl und Priorisierung von Testfällen gesenkt und somit eine größere Testtiefe für die kritischen Funktionen erreicht werden.

5.5 CI/CD/CT

Abschließend sollen noch die häufig unter Continuous-X zusammengefassten Aspekte thematisiert werden: Continuous Integration (CI), Continuous Delivery (CD) und Continuous Testing (CT). Hierbei geht es grundlegend darum, automatisiert und kontinuierlich auf Änderungen in der Softwareentwicklung zu reagieren. Ändert ein Entwickler etwas im Code und lädt diesen in ein Repository, so wird sichergestellt, dass diese Änderungen automatisiert in einen neuen Build der entsprechenden vECU resultieren. Die vECU könnte, sofern alle automatisierten Konformitätsprüfungen bestanden sind, der vernetzen SiL-Plattform geliefert und dort einem umfassenden Test unterzogen. Ob ein so getesteter Softwarestand weiter auf Entwicklungsfahrzeuge ausgerollt werden darf, müssen die Regeln im Test- und Releasemanagement definieren. Es liegt jedoch auf der Hand, dass eine vernetzte SiL-Plattform durch ihre softwareseitige Umsetzung optimal für Continuous-X-Ansätze geeignet ist. Die Zeitersparnis bei einem solchen Durchlauf ist teils enorm: Dauert es an einem HiL-Prüfstand inkl. Flashen der neuen Software durchaus Stunden bis Tage, kann eine parallelisierte SiL-Plattform schon nach einigen Minuten die Testergebnisse liefern.

6 Zusammenfassung und Ausblick

In diesem Beitrag wurde die Entwicklung der SiL-Technologie über die letzten Jahre grob skizziert sowie der geläufige Einsatz von SiL im Rahmen des V-Modells thematisiert und bewertet. Im Hinblick auf die zunehmende Geschwindigkeit bei Softwareänderungen, aber auch die massive Zunahme an Softwarefunktionen im Fahrzeug, wurde ein Ansatz vorgestellt, mit dem der gesamte Entwicklungsprozess optimiert werden kann, um mit dieser Entwicklung Schritt zu halten. Ebenso wurde mit dem Ansatz des vernetzten SiLs eine Möglichkeit vorgeschlagen die Absicherung im Rahmen der Systemintegration und des Systemtests deutlich früher zu beginnen und in allen Phasen des entwicklungsbegleitenden Tests neue Testkapazitäten zu ergänzen.

7 Literaturverzeichnis

[PEG19] „Pegasusprojekt,“ Bundesministerium fürWirtschaft und Energie, 2019. [Online]. Available: https://www.pegasusprojekt.de/de/qtronic-gmbh. [Zugriff am 09.09.2022].

[PIK22] PikeTech GmbH, „PikeTech.com,“ PikeTech GmbH, 2022. [Online]. Available: https://piketec.com/de/ueber-uns/. [Zugriff am 09.09.2022].

[AUT22] AUTOSAR, „autosar.org,“ AUTOSAR, 2022. [Online]. Available: https://www.autosar.org/about/history/. [Zugriff am 09.09.2022].

[FMI22] „fmi-standard.org,“ Modelica Association, 2022. [Online]. Available: https://fmi-standard.org/about/. [Zugriff am 09.09.2022].

[DSP22] dSPACE GmbH, „dSPACE.com,“ dSPACE GmbH, 2022. [Online]. Available: https://www.dspace.com/de/gmb/home/company/companyhistory.cfm. [Zugriff am 09.09.2022].

[BUC11] C. Buckl, „Mehr Software (im) Wagen: eCar-IKT-Systemarchitektur für Elektromobilität, Teilvorhaben: IKT-Architekturszenarien und Auswirkungen auf Deutschland,“ 2011.

[PRO20] prostep IVIP, „Requirements for the Standardization of Virtual Electronic Control Units (V-ECUs),“ *Smart Systems Engineering*, 2020.