SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Erstellen einer einheitlichen Plattform für die Entwicklung von Embedded-Anwendungen für Automotive

Kurzdarstellung

Die Komplexität von Embedded Software für Automotive steigt, da Softwareanwendungen bei der Bereitstellung der erforderlichen Merkmale und Funktionen für moderne Fahrzeuge eine immer größere Rolle spielen. Durch die steigende Bedeutung der Software wird auch das Zusammenspiel von Hardware, Software und physischen Systemen zunehmend komplexer. Aktuelle Softwareentwicklungsmethoden sind außerstande, die vielen und komplexen cyber-physischen Schnittstellen der heutigen Automobile zu bewältigen. Um wettbewerbsfähig zu sein, müssen Unternehmen ihren Softwareentwicklungsprozess weiterentwickeln und einen digitalen roten Faden einrichten, der die Entwicklung von Software mit der von physischen Systemen verbindet.

Piyush Karkare
Director – Global Automotive Industry Solutions

Inhalt

Das moderne Automobil ist ein komplexes System aus leistungsfähigen Rechengeräten, Sensoren, Aktoren sowie mechanischen und mechatronischen Systemen. Heutzutage werden immer mehr Fahrzeugmerkmale und -funktionen auf elektronische Weise erzeugt. Die Anwender erwarten bei der Nutzung der ausgeklügelten Merkmale moderner Fahrzeuge eine nahtlose und gleichzeitig intuitive Nutzererfahrung (Abbildung 1). Durch diesen Trend bekommt die Software, mit der diese Fahrzeugsysteme gesteuert werden, eine viel größere Bedeutung. Zudem ist die Software entscheidend für Unternehmen, die von dem Versprechen vernetzter, elektrifizierter und autonomer Fahrzeuge profitieren möchten. Dies führt zu einem deutlichen Anstieg der Raffinesse und der Komplexität der fahrzeuginternen Software und damit verbunden zu der Herausforderung, diese Software entwickeln, verifizieren und validieren zu müssen.

Produkt- und Softwareentwicklung folgen grundsätzlich verschiedenen Entwicklungslebenszyklen. Konstrukteure handhaben die Softwareentwicklung separat und validieren häufig die Software-Hardware-Schnittstelle nur an zuvor festgelegten Prüfpunkten. Aufgrund der zunehmend mit Intelligenz und Konnektivität ausgestatteten Fahrzeuge wird das Zusammenspiel von Software und physischen Systemen erschwert und die Defizite aktueller Prozesse, Werkzeuge und Methoden stärker sichtbar. Softwareentwicklung, die außerhalb des ganzheitlichen Fahrzeugsystems durchgeführt wird, verstärkt die Probleme mit der Hardwarekompatibilität, Softwarequalität und Programmverantwortung.

Unternehmen müssen ihre Softwareentwicklungsprozesse heute weiterentwickeln, wenn sie beim technologischen Wettlauf um die Zukunft der Mobilität eine Rolle spielen wollen. Ein gemeinsamer digitaler roter Faden, mit dem Software- und physische Systeme verbunden werden, ist die einzige Möglichkeit, die steigende Komplexität von intelligenten und vernetzten Produkten in den Griff zu bekommen. Dadurch wird eine als geschlossene Schleife aufgebaute Darstellung von Software- und Hardwaresystemen eines Fahrzeugs zur ständigen Validierung während des gesamten Produktlebenszyklus ermöglicht. Ein robuster digitaler roter Faden unterstützt Konstrukteure dabei, die Kompatibilität der Softwarefeatures mit dem Fahrzeug, in das sie eingesetzt werden, sicherzustellen und nachzuweisen, dass alle verbundenen Aufgaben und Ergebnisse rechtzeitig verfügbar sind. Mit dem digitalen roten Faden kann nicht nur nachverfolgt werden, wer Änderungen vorgenommen hat, sondern auch wie und warum.

Abbildung 1: Moderne Verbraucher erwarten eine nahtlose Nutzerführung, unabhängig davon, ob es sich um selbstfahrende Fahrzeuge handelt oder nicht

Herausforderungen und der funktionszentrierte Ansatz

Die fahrzeuginterne Softwareentwicklung ist eine komplexe Umgebung, an der verschiedene Domänen unternehmensübergreifend beteiligt sind, um gemeinsame Fahrzeugmerkmale zu implementieren (Abbildung 2). Diese Softwarefeatures entstehen aus Anforderungen, die durch Interaktionen auf Fahrzeug- oder Plattformebene bestimmt werden. Diese werden dann mit der erforderlichen elektrischen Konnektivität und Netzwerkkommunikation weiterentwickelt.

Beim Entwickeln und Erstellen von Anwendungen wird leicht vergessen, dass jede einzelne Anwendung Teil eines größeren Systems ist. Anwendungen nutzen einzelne Fahrzeugfunktionen zur Durchführung einer Aufgabe, die dann eingesetzt wird, um eine oder mehrere Fahrzeugmerkmale zu aktivieren. Genauso wie eine Anwendung die Aktivierung mehrerer Fahrzeugmerkmale unterstützen kann, kann ein einzelnes Fahrzeugmerkmal verschiedene Anwendungen unterschiedlicher elektrischer Steuergeräte (Electronic Control Units, ECUs) nutzen.

In diesem Fall muss das Softwarefeature für die Spurhalte-Assistenz mit Sensoren, mehreren Prozessoren und Lenkaktoren interagieren und den Fahrer über die ausgeführten Handlungen informieren. Die Anforderungen für jede einzelne Embedded Software-Anwendung werden vom System-Design und dem Entwicklungsprozess auf Fahrzeugebene abgeleitet. Das ist der Systemkontext für die Embedded Software-Anwendung. Eine wesentliche Voraussetzung stellt hierbei die Berücksichtigung des Systemkontexts entlang des gesamten Embedded Software-Entwicklungsprozesses dar.

In der Regel übernehmen Originalgerätehersteller (OEMs) die gesamte Entwicklung der Softwarefeatures auf Fahrzeugebene. Der größte Teil der detaillierten Steuerungsalgorithmen und der Entwicklung der Embedded Software-Anwendung erfolgt auf Lieferantenebene. Allerdings beginnen OEMs damit, entscheidende Funktionen wie Infotainment, ADAS und Antriebsfunktionen intern zu entwickeln, da der Wettbewerb in diesen Bereichen sehr hoch ist.

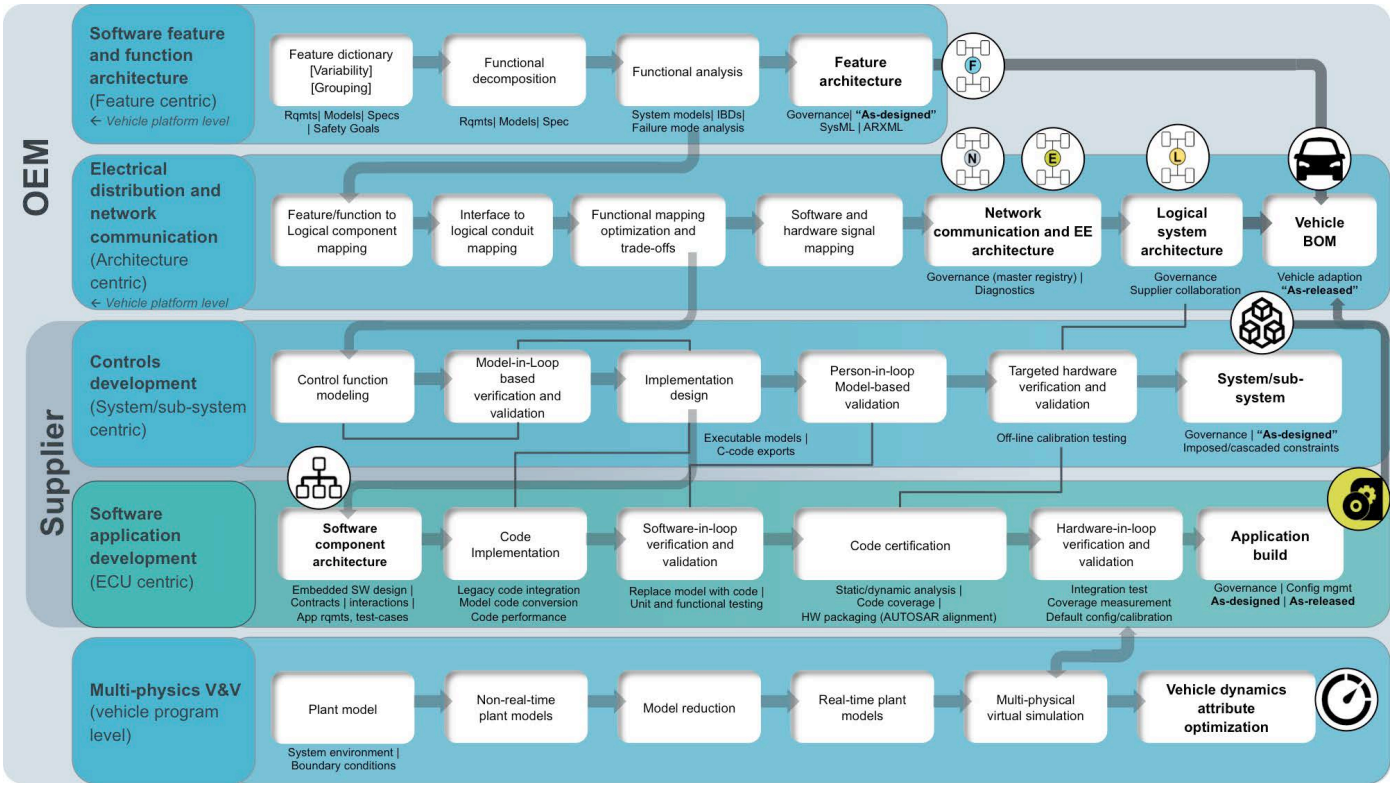

Abbildung 2: Embedded Software-Entwicklung für Automotive ist eine sehr komplexe Aufgabe, die viele Unternehmen aus unterschiedlichen Konstruktionsbereichen involviert

Für die Verifizierung und Validierung der gewünschten Funktionalität benötigen diese Softwarefeatures Steuerungsalgorithmen. Diese lassen sich als das spezifische Verhalten und die Interaktionen der in elektronische Steuergeräte (ECUs) implementierten Softwarekomponenten zusammenfassen. Diese Komponenten werden in einer Softwareanwendung zusammengefasst, die am cyber-physischen System, das die Fahrzeugfunktion implementiert, mitwirkt. In der Regel sind Softwareanwendungen mit allen Fahrzeugen, die eine gemeinsame Plattform teilen, kompatibel.

Jeder Lieferant ist nur für die eigenen Inhalte und nicht für übergreifende Merkmale oder für Merkmale verantwortlich, die seine Steuergeräte voraussichtlich unterstützen werden. Entwicklungsunternehmen auf Lieferanten- und OEM-Seite tendieren dazu, einen ECU-(Hardware-)zentrierten Ansatz zu wählen – sich also weniger auf die zu implementierende Funktion zu konzentrieren. Die kontinuierliche Verifizierung und Validierung des Softwarefeatures zu gewährleisten, nachdem die verschiedenen Software- und Hardwarekomponenten ausgereift sind, stellt dabei eine große Herausforderung dar und ist mit einem hohen Zeitaufwand verbunden. Findet die Verifizierung und Validierung erst verspätet statt, entdecken Konstrukteure Fehler auf Funktionsebene häufig erst zu einem späten Zeitpunkt. Das hat kostspielige Änderungen in der Spätphase, verkürzte Testzeiten und verspätete Produkt- oder Funktionsfreigaben zur Folge. Mit steigender Komplexität und einem Anstieg verteilter Merkmale steigt die Wahrscheinlichkeit, dass Fehler nicht oder zu spät entdeckt werden.

Eine funktionszentrierte Softwaredefinition kann die Unklarheiten bei der Embedded Software-Definition für Automotive beseitigen. Mit einem funktionszentrierten, architekturbasierten Ansatz können sich Systemkonstrukteure auf die Definition und Ausführung der funktionsorientierten Zerlegung der Softwarefeatures (Softwarearchitektur) konzentrieren und logischen elektrischen Komponenten wie ECUs, Schaltern, Aktoren und anderen (logische Architektur) Softwarekomponenten zuweisen und wiederverwenden. Elektrische Konnektivität, Stromverteilung, Erdung und Sicherungen können jetzt basierend auf den Anforderungen des Softwarefeatures (elektrische Architektur) optimiert werden. Auch die Netzwerkkommunikation kann optimiert und geschützt werden (Netzwerkarchitektur).

Wenn die Fahrzeugprogramme anlaufen, identifizieren bestehende Hardware-Software-Beziehungen zwischen Softwarefunktionen und logischen elektrischen Komponenten alle elektrischen Komponenten, die für die einzelnen Fahrzeugfunktionen erforderlich sind. Logische Komponenten können dann physischen Teilenummern zugewiesen werden, die innerhalb des jeweiligen Fahrzeugprogramms nachverfolgt werden können. So entsteht eine End-to-End-Nachverfolgbarkeit über die Entwicklung der Softwarefeatures und ihre Verwendung durch die einzelnen Fahrzeugprogramme (physikalische Architektur). Als Lieferant oder OEM den Überblick über alle Aktivitäten zu behalten, wird zu einer andauernden und herausfordernden Aufgabe. Eine effektive, alle Aktivitäten umfassende Zusammenarbeit unterstützt Konstrukteure bei der Bereitstellung kompatibler, vollständiger und verlässlicher Softwarefeatures für das Fahrzeug.

Eine einheitliche Plattform für die Embedded Software-Entwicklung für Automotive

Die wirksame Umsetzung eines funktionszentrierten Ansatzes erfordert eine robuste, sichere und allgemein zugängliche Methode. So können komplexe Softwarefeatures entwickelt, erstellt, nachverfolgt und verbessert werden, die über eine Vielzahl fahrzeuginterner ECUs verteilt sind und häufig von vielen Lieferanten weltweit bezogen werden.

Diese Methode erfordert eine einheitliche Plattform für die Embedded Software-Entwicklung, um alle Aktivitäten über ein vielseitiges Tool-Set zu koordinieren und einen vollständig verifizierten und validierten Build unter Hardware- und Systemkonfigurationseinschränkungen (Abbildung 3) bereitstellen zu können. Diese Plattform ermöglicht es OEMs und Zulieferern, für einen konsolidierten und einheitlichen Datenfluss entlang der Werkzeugumgebung zu sorgen, um die Prozess-, Methoden- und Werkzeugintegration zu optimieren.

Eine einheitliche Plattform, die alle Aktivitäten entlang der Anwendungsdefinition, Planung und des Entwicklungs- und Bereitstellungs-Lebenszyklus der Embedded Software koordiniert und sich gleichzeitig mit verschiedenen Toolsets verbindet, ermöglicht eine organische Zusammenarbeit zwischen den Konstrukteuren, sichert Rückverfolgbarkeit und propagiert die Wiederverwendung verfügbarer Daten.

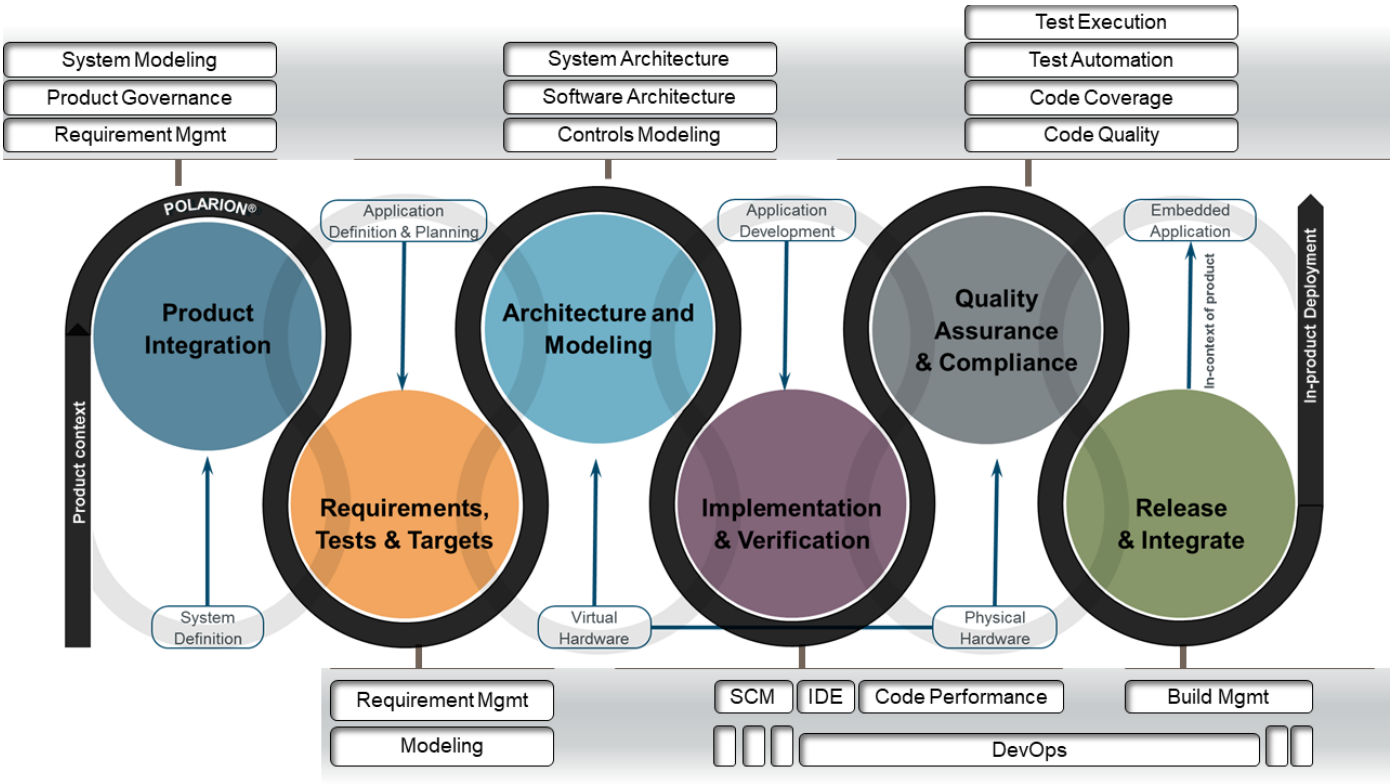

Abbildung 3: Eine einheitliche Plattform für die Embedded Software-Entwicklung für Automotive ist erforderlich, um verifizierte und validierte Anwendungs-Builds auf Basis von Hardware- und Systemeinschränkungen bereitzustellen

Die drei Prozesse der Entwicklung von Embedded Software

Die Steuerung der Entwicklung von Embedded-Anwendungen für Automotive kann in drei Segmente unterteilt werden (Abbildung 4):

- Definition und Planung von Anwendungen
- Anwendungsentwicklung und Qualitätssicherung
- Bereitstellen und Überwachen von Anwendungen

Der Schlüssel zum Erfolg ist die kontinuierliche Gewährleistung konsistenter und kompatibler Arbeitsaufgaben, die von den Beteiligten über eine Vielzahl von Plattformen und Organisationen hinweg ausgeführt werden. Eine einheitliche Softwareentwicklungsplattform unterstützt Unternehmen bei der Koordination dieser Prozesse und der damit verbundenen Aktivitäten mit einem einheitlichen digitalen roten Faden. Einzelne Unternehmen können dann agile oder andere Methoden nutzen, um robuste Embedded Software-Anwendungen im Kosten- und Zeitrahmen bereitzustellen.

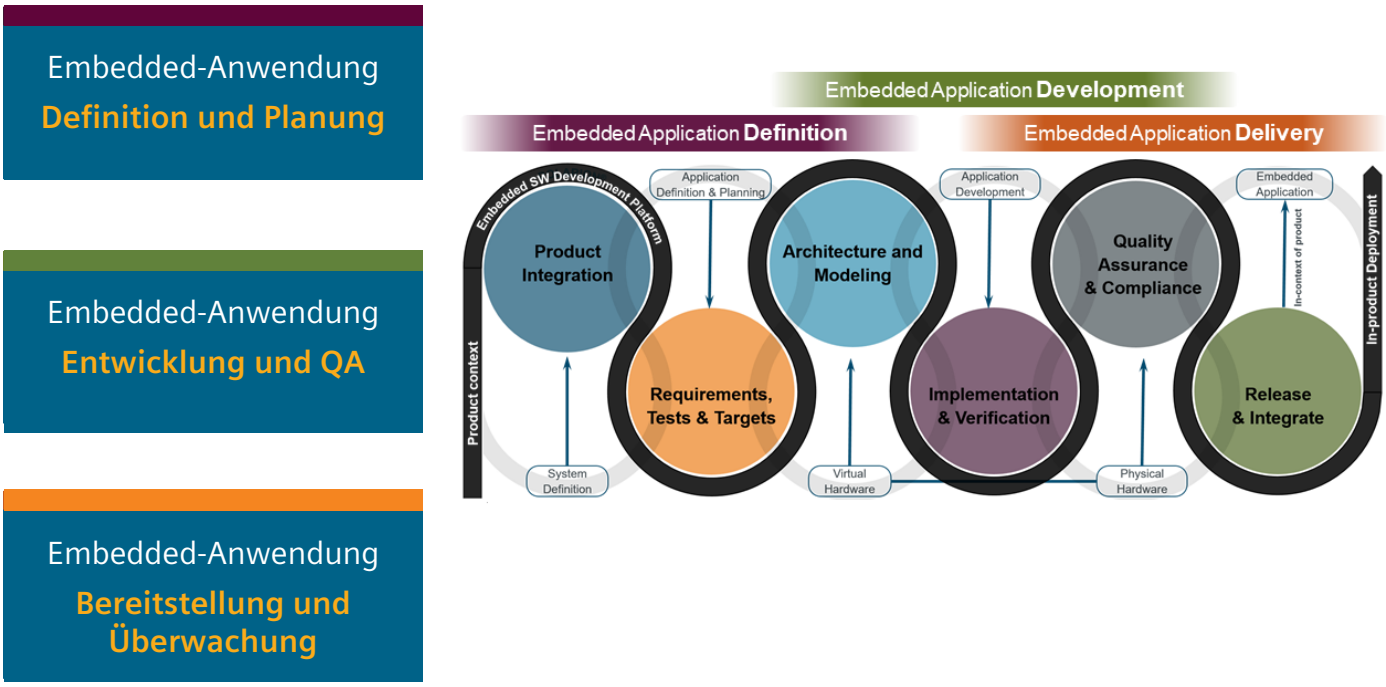

Abbildung 4: Die drei Segmente der funktionszentrierten Entwicklung von Embedded-Anwendungen für Automotive

Definition und Planung von Anwendungen

Polarion, das erweiterte Werkzeuge für die Steuerung der Softwareentwicklung, kann Produktdefinitionen auf Systemebene nutzen und nachverfolgen, um eine direkte Verknüpfung zwischen Änderungen auf Systemebene und Anwendungsentwicklung zu erstellen. So kann gewährleistet werden, dass Anwendung und allgemeine Systementwicklung jederzeit synchron laufen. In diesem Fall stellt das System eine Verallgemeinerung der Abstraktion des Merkmals auf Fahrzeugebene dar.

Die Definitionen auf Systemebene und die Hardwarebeschränkungen werden in Anforderungen und Spezifikationen auf Systemebene festgeschrieben. Softwareingenieure können dann die erforderliche Kombination aus Hardware- und Softwareanforderungen, Störfaktoren, FMEA-Prozessen, Testfällen und Zielen auf die Ebene der eingebetteten Anwendung herunterbrechen und mit der Definition und Planung der Embedded Software auf Anwendungsebene beginnen. Softwareingenieure können während der Planung Aufgaben für die Modellierung, Programmierung, Testausführung, Build-Erstellung und vieles mehr in den erforderlichen Werkzeugen zuweisen. Sie haben jederzeit Einblick in die Definitionen und Randbedingungen auf Systemebene und können mit anderen Systembenutzern zusammenarbeiten. Zudem erhalten sie Benachrichtigungen, wenn Änderungen auf Systemebene stattgefunden haben, um so die Auswirkungen auf Softwareebene evaluieren zu können.

Die Erstellung von detaillierten Softwareanforderungen und alle nachfolgenden Änderungen ziehen Änderungen an der Architektur und Modellierung der Software nach sich. Mit den Anforderungen können Softwareingenieure diese Modelle aktualisieren, sodass sie mit Steuerungsalgorithmen in Einklang stehen. Sie können jetzt detaillierte MiL-Tests (Model-in-the-Loop) durchführen, um zu überprüfen und zu verifizieren, dass auf Anwendungs- und Systemebene die gewünschten Ergebnisse erzielt werden, bevor Änderungen auf Codeebene ausgelöst werden. Dies ermöglicht es Konstrukteuren, entscheidende Risiken, Probleme und Defekte zu erkennen, während sie bereits frühzeitig an den Vorgaben auf Systemebene ausgerichtet sind.

Anwendungsentwicklung und Qualitätssicherung

Die Softwarekomponentenarchitektur und Modellierungsaufgaben verifizieren und validieren, dass mit der Komponenteninteraktion die gewünschte Funktion erzeugt wird. Da die Modelle im Zuge der Verifizierung und Validierung zuverlässiger und vollständiger werden, können Codeänderungen und Aktualisierungen abgeschlossen und mit Software-in-the-Loop- (SiL) und Hardware-in-the-Loop-Tests (HiL) geprüft werden. Konstrukteure können anschließend Aktualisierungen auf der Ebene der Fahrzeugmodellierung und erneute Tests durchführen. So wird die Übereinstimmung, Kompatibilität und Gesamtverantwortung mit den erforderlichen Berichten und Audit-Protokollen auf der Ebene der Fahrzeugfunktionen gewährleistet.

Dieser modellbasierte Ansatz beschleunigt nicht nur den Prozess, sondern kann auch mit Methoden wie SOTIF (Safety of the Intended Functionality) gewährleisten, dass die Software wie vorgesehen arbeitet und dass Risiken per Design ausgeschlossen werden. Die Einbindung von SOTIF-Methoden ergänzt standardmäßige Ansätze für die funktionale Sicherheit und Ansätze der systemtheoretischen Prozessanalyse (STPA), die Risiken durch die Anwendung von Sicherheitszielen minimieren, die vom Auftreten von Fehlern ausgehen. Diese Kombination führt zu außergewöhnlich robusten Embedded Software-Anwendungen für Automotive. Auf diese Weise können Konstrukteure auch Standard-Prozessbeschreibungen und erforderliche Arbeitsprodukte in die einheitliche Softwareumgebung einbinden. Dies ermöglicht es Konstrukteuren, sich auf konstruktionsspezifische Aufgaben zu konzentrieren, während das Tool die Prozesskonformität, Nachverfolgbarkeit, Konsistenz, Zusammenarbeit und den Lebenszyklus der erzeugten Daten bewältigt.

Bereitstellen und Überwachen von Anwendungen

Schließlich muss die Embedded-Anwendung für die Softwarebaugruppe, die in die fertige Fahrzeug-Stückliste (BOM) integriert werden wird, bereitgestellt werden. Aber zunächst müssen die Konstrukteure die Anwendung für die Bereitstellung vorbereiten und eine Infrastruktur für die anschließende Überwachung der Anwendung einrichten. Dazu zählt die permanente Kontrolle, wie und wo die Anwendung in der Fahrzeugarchitektur verwendet wird.

Die Lösung für die Softwareentwicklungsplattform kann sich während der gesamten Anwendungsentwicklung, Bereitstellung und Überwachung mit verschiedenen Werkzeugen verbinden, um Code-Performance und Testabdeckungsdaten zu bieten und die Übereinstimmung mit Methoden und Codierstandards wie

MISRA zu gewährleisten. Zudem unterstützt die einheitliche Softwareentwicklungsumgebung die Zusammenarbeit und die Koordination, die für die richtige Aktualisierung des Codes und die notwendigen Qualitätssicherungsprozesse erforderlich sind.

Schließlich muss die Anwendung mit virtueller und physischer Hardware und anderen Peripheriekomponenten verifiziert und validiert werden, um die gewünschte Funktionalität und das sichere Funktionieren des Systems zu gewährleisten. Damit wird sichergestellt, dass die Embedded Software-Anwendung vollständig und kompatibel ist und den Anforderungen auf der Ebene des Fahrzeugsystems gerecht wird.

Die Softwareingenieure sind für die Bereitstellung des jeweiligen Software-Builds für die richtige Fahrzeugvariante verantwortlich. Fahrzeugplattformen bringen Dutzende unterschiedlicher Fahrzeugausführungen mit einem Mix aus gemeinsamen und spezifischen Merkmalen, Komponenten und Embedded Hardware hervor. Die Softwaredomäne muss die Konfiguration der Builds an jede dieser Ausführungen anpassen, was schnell sehr kompliziert werden kann. Softwareingenieure müssen sicherstellen, dass die für das jeweilige Fahrzeug bereitgestellte Software vollständig mit der Hardware in dieser Fahrzeugvariante kompatibel ist. Dies wird im Allgemeinen erzielt, indem Softwarekonfiguration, Kalibrierung und Bootladeprogramme, die mit der jeweiligen ECU in der Fahrzeugstruktur übereinstimmen, für die Bereitstellung als eine Softwarebaugruppe zusammengestellt werden. Das ist besonders wichtig, wenn Kundenfahrzeuge im Einsatz aktualisiert werden (Abbildung 5).

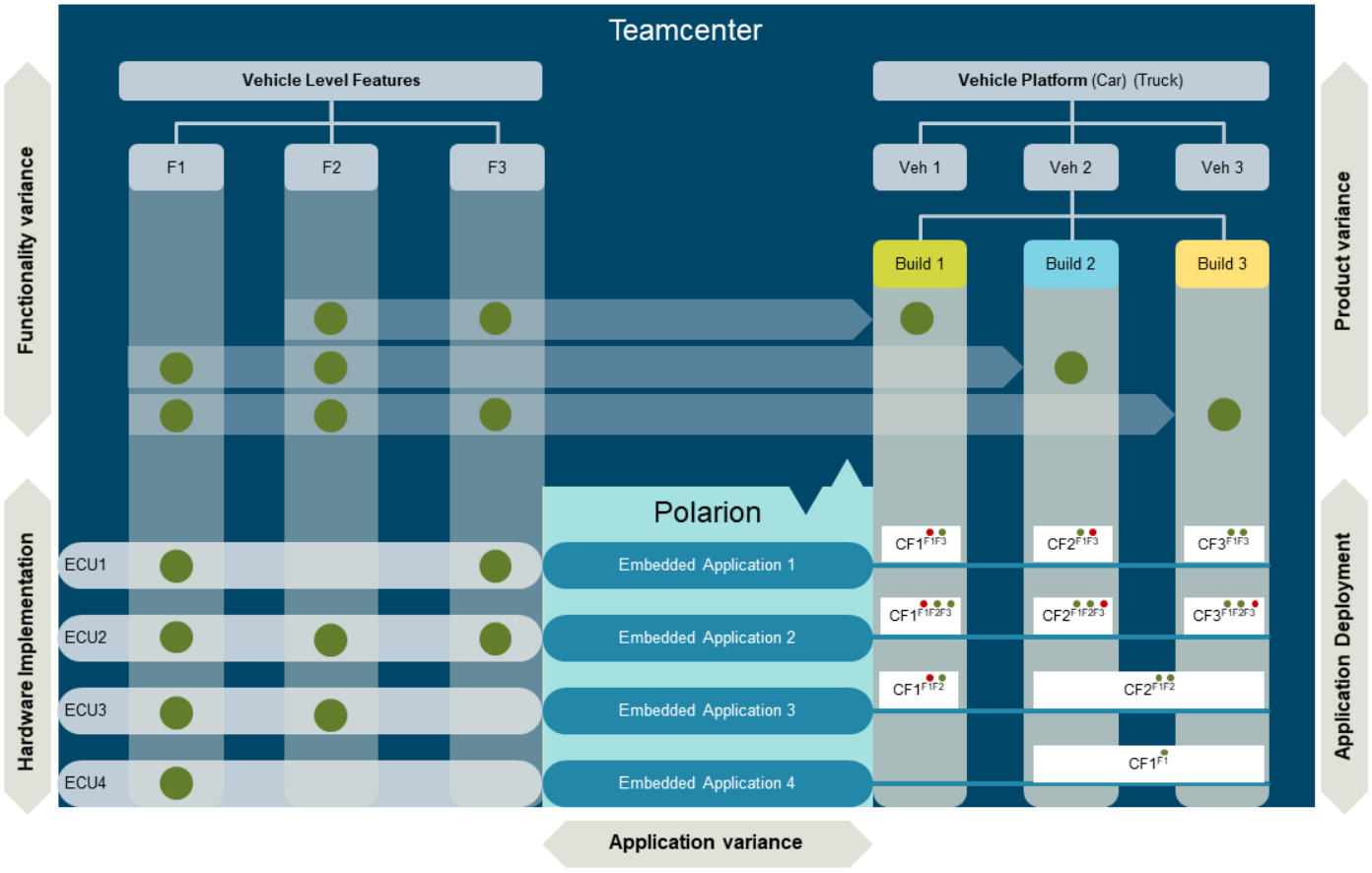

Abbildung 5: Fahrzeugausstattung, abweichende Fahrzeugausführung und Verwaltung der Anwendungs-Builds mit Konfigurationen

Ein weiterer wichtiger Bestandteil von Embedded Software für Automotive ist das Anwendungskonfigurationsmanagement. Die Softwareanwendungsentwicklung hängt zunehmend weniger von der Hardware ab, auf der die Anwendung läuft, wenn Unternehmen mit der Entwicklung von Embedded-Betriebssystemen beginnen. Das Gleiche gilt für Smartphone-Anwendungen und Betriebssysteme. Die Anwendungskompatibilität basiert primär auf dem Betriebssystem und nicht auf der Hardware. Daher ist es eine große Herausforderung, die Kompatibilität zwischen den Embedded Software-Anwendungen für Automotive und der Embedded Software auf Systemebene (Echtzeit-Betriebssystem, Basissoftware usw.) sowie zugrundeliegenden Hardware- und Fahrzeugvarianten herzustellen (Abbildung 6).

Eine zuverlässige Kombination aus Application Lifecycle Management (ALM) und Product Lifecycle Management (PLM) ist eine Grundvoraussetzung im Umgang mit dieser Komplexität. Eine einheitliche Plattform ermöglicht es dem OEM, die grundlegenden Prozesse und die Infrastruktur für "As-Designed" (für die Entwicklung), "As-Released" (für Engineering), "As-Built" (für Fahrzeugmontagewerke) und "As-Serviced" (für Over-the-air oder Händlerupdates) Software-Builds zu erstellen und zu unterstützen.

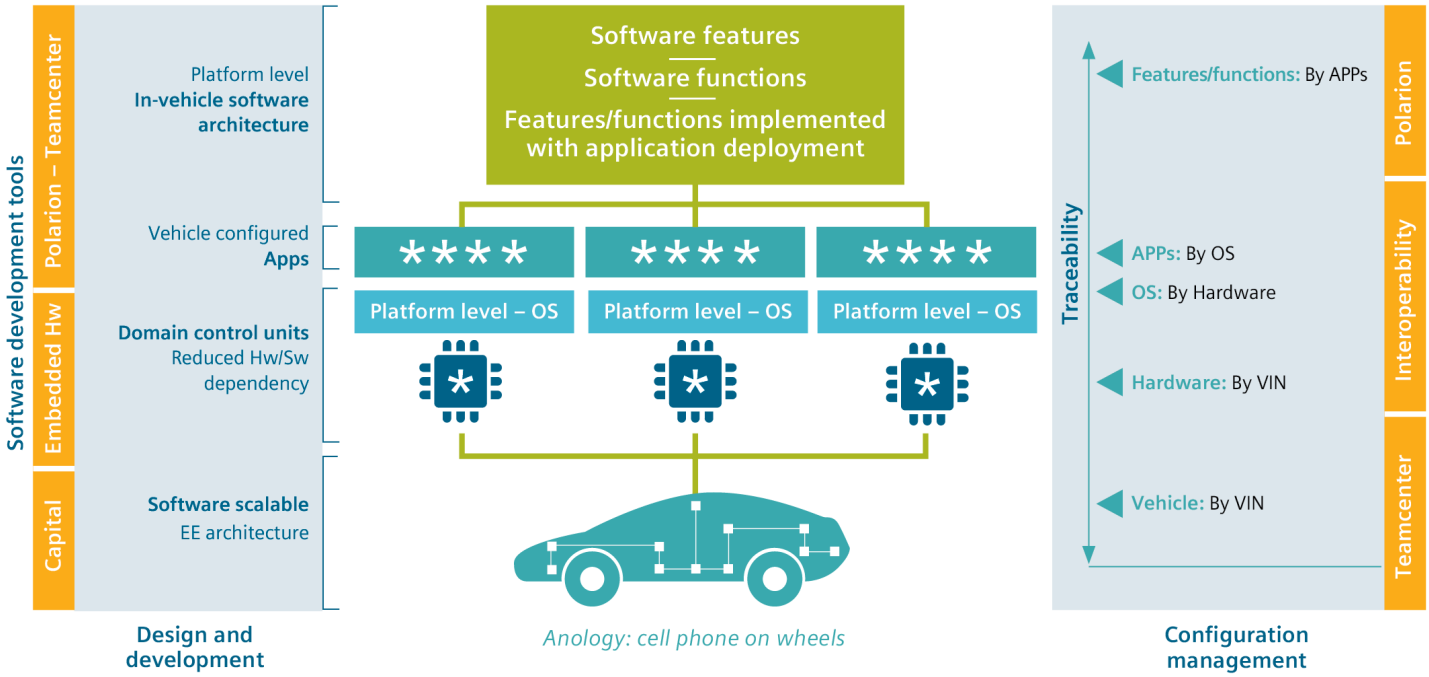

Abbildung 6: Das Konfigurationsmanagement von Embedded-Anwendungen für Automotive stellt im Zuge dessen, dass Software immer weniger von der Hardware abhängt, eine zunehmende Herausforderung dar; ähnlich wie bei der Anwendungsumgebung von Smartphones.

Beispiel für die Anwendungsentwicklung

Dieses Beispiel zeigt die Entwicklung einer neuen Anwendung für ein spezifisches Steuergerät, in diesem Fall dem Modul für den Abstandsregeltempomat (Adaptive Cruise Control Module, ACCM) unter Verwendung einer auf Polarion basierenden Plattform für die Softwareentwicklung (Abbildung 7). In der Abstraktion der Systemebene wurde eine Änderung durch die Kombination von zwei Funktionalitäten erzeugt – den automatischen Bremssystemen (AEB) und dem Abstandsregeltempomat (ACC). Als Erstes müssen diese neuen Systemanforderungen und die Systemspezifikation zum ACCM und allen betroffenen Subsystemen kaskadiert werden.

Dafür sind umfangreiche Softwareänderungen am ACCM erforderlich, die von Änderungen außerhalb des ACCM, die nicht softwareseitig sind, ergänzt werden. Zu den Änderungen am ACCM können neue Systemanforderungen, Hardwarespezifikationen, FMEAs, Sicherheitsanforderungen und mehr zählen. Softwareingenieure im Bereich des ACCM müssen über eine kontinuierliche Verbindung zum Systemkontext verfügen, da sie die erforderlichen Änderungen und Aktualisierungen an der ACCM-Software vornehmen. Die Verbindung des Koordinierungswerkzeugs für die Softwareentwicklung mit PLM-Lösungen (Product Lifecycle Management) ermöglicht diese kontinuierliche Verbindung zwischen Teams und Abstraktionen, insbesondere mit dem Änderungsmanagement der Softwareplattformsteuerung und den PLM-Aktivitäten auf Funktionsebene.

Konstrukteure im Bereich des ACCM können auf Grundlage der Änderungen auf Systemebene die notwendigen Änderungen und Aktualisierungen an softwarespezifischen Anforderungen, Testfällen, Testmethoden und mehr implementieren. Arbeitselemente, die basierend auf den bestehenden Datenartefakten im Softwareverwaltungstool von diesen Änderungen betroffen sein könnten, lassen sich vor der Durchführung von Änderungen für die weitere Prüfung kennzeichnen. Die meisten dieser Arbeitselemente wären dann als Teil der Konstruktionsänderungsanforderungen (ECRs) für die AEB+ACC-Aktualisierung bereits im Vorfeld für diese Änderung vermerkt. Auf diese Weise verwalten die PLM-Lösungen und die Koordinierungslösungen für die Softwareentwicklung konsistente Daten für ein kontinuierliches Änderungsmanagement.

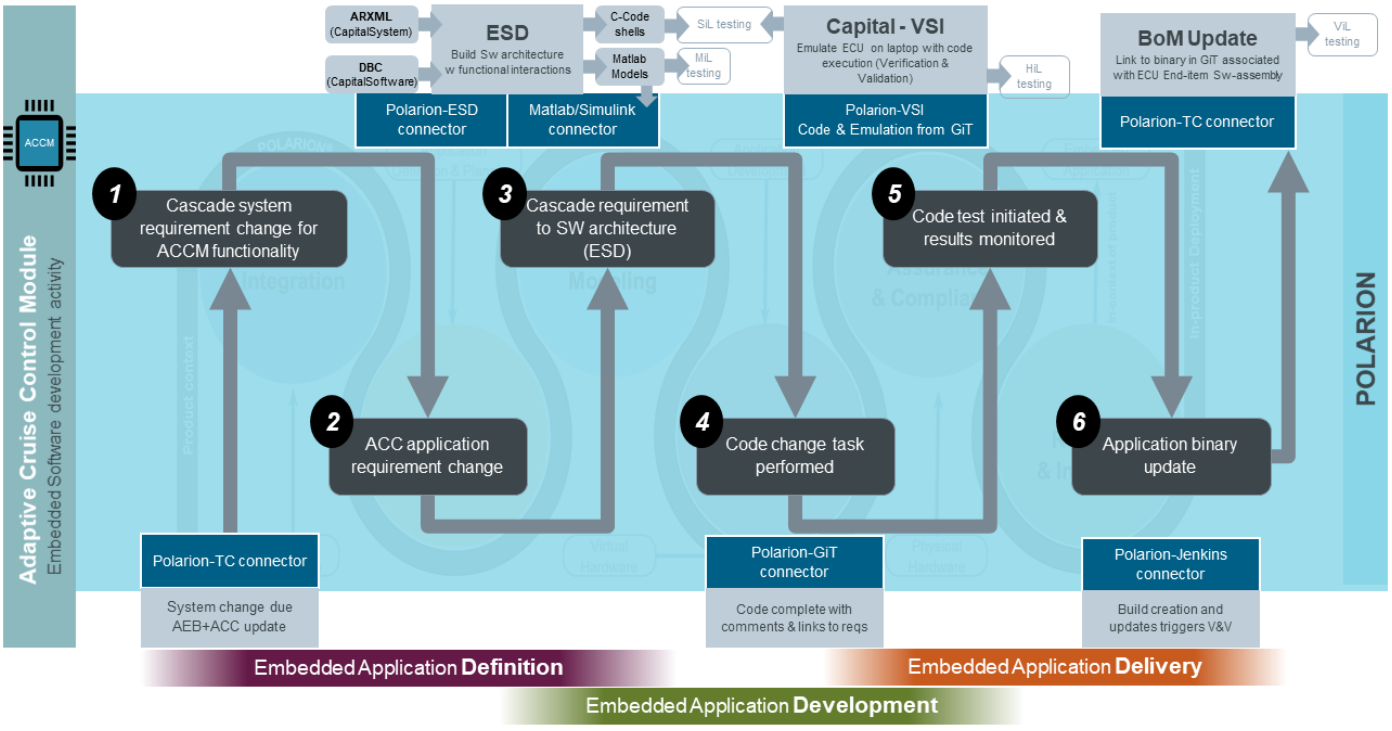

Abbildung 7: Das Beispiel für die Anwendungsentwicklung zeigt die Vorteile einer einheitlichen Softwareentwicklungsplattform

Agile Prozesse können die Änderungen auf Systemebene ausführen. Die darauffolgenden Aktualisierungen an Softwareanforderungen stoßen potenzielle Aktualisierungen der Softwarekomponenten-Architektur und der Modelle im Embedded Software Designer oder MATLAB/Simulink an. Die einheitliche Softwareentwicklungsplattform kann die Modelle, an denen Änderungen wahrscheinlich sind, koordinieren und kennzeichnen, da diese Änderungen kaskadieren und sich auf der Systemebene untergliedern. Da diese Änderungen der Softwarekomponenten auf Architektur- und Modellebene ausgeführt werden (zusammen mit elektrischen Systemen und CAN-Bus Netzwerkkommunikationsänderungen, die aus dem elektrischen Systemtool importiert wurden), kann ESD die Modelle für Matlab und die Programmiersprache C für die MiL-Testausführung (Model-in-the-Loop) exportieren. Abhängig von der Ebene und der Anzahl der Änderungen, können auch vorausgehende Software-in-the-Loop-Tests initiiert werden, um die Funktionsschnittstellen basierend auf der aktualisierten Architektur zu validieren, bevor überhaupt Änderungen am Code vorgenommen werden.

Nachdem die Funktionsschnittstellen verifiziert und validiert wurden, können die Softwareingenieure mit Sprints für die Implementierung von Codeänderungen beginnen. Diese Codeänderungen können wieder mit den Quellanforderungen aus Schritt zwei und mit Änderungen an der Architektur verknüpft werden. Dadurch wird gewährleistet, dass die Änderungen korrekt sind und sich bis zu den Anforderungen auf Systemebene und darüber hinaus vollständig rückverfolgen lassen. Dies kann beispielsweise eine Anweisung vom Marketing oder eine Änderung aufgrund einer Gewährleistung aus der Praxis sein, um ein immer wieder auftretendes Problem mit einem Kundenfahrzeug zu beseitigen. Diese aufwärtsgerichtete Rückverfolgbarkeit hat viele Vorteile, da weitreichende, fahrzeugübergreifende Programmänderungen geprüft werden können. Zudem lassen sich Funktionen über eine einzelne statt mehrerer Aktualisierungen ändern oder reparieren, was zu einer erheblichen Kosteneinsparung führt.

Als Teil von Agile-Sprints kann aktualisierter Code durch SiL auf virtueller Hardware getestet werden oder über HiL mithilfe physischer Hardware. Diese durch Agile-Methoden ausgelöste Tests ermöglichen eine kontinuierliche Integration in Software-Builds und Hardware-Randbedingungen oder Randbedingungen auf Systemebene. Mit der Softwareentwicklungsplattform lassen sich auch Hardware-/Software-Tests auf Systemebene ausführen und verwalten, für die diese Codeänderungen bereitgestellt wurden. Zudem helfen die Gesamtergebnisse der Tests und Projekt-Dashboards den Konstrukteuren dabei, zu überprüfen, dass eine robuste, verifizierte und validierte Softwareanwendung für den Einsatz zur Verfügung steht.

Anschließend kann die fertige Softwarebaugruppe für die „As-Released"-Stückliste mit Änderungsbenachrichtigungen (Engineering Change Notification, ECN), die vom PLM-Tool angestoßen wurden, aktualisiert werden. Die ECN richtet sich an die spezifische Teilenummer des Steuergeräts in der konfigurierten Fahrzeugstruktur. So wird sichergestellt, dass eine aktuelle Softwarebaugruppe konfiguriert werden kann, um in die abschließende Fahrzeugmontage integriert zu werden. Diese wird dann für das Flashen der Steuergeräte an die Fahrzeugmontagewerke geleitet.

Vereinheitlichen der Embedded Software-Entwicklung für Automotive

Die Integration der Software- und Produktentwicklungsumgebungen erzeugt durch Verbinden der relevanten Architekturdefinition, Modellierung und Testwerkzeuge eine einheitliche Plattform für Embedded Software-Engineering für Automotive. Diese Plattform agiert als ein kollaboratives Arbeitsumfeld, das Verfolgbarkeit bietet und die Wiederverwendung geistigen Eigentums fördert. Dies führt zu kürzeren Entwicklungszeiten und einer Kostenreduzierung, indem Inkompatibilitäten frühzeitig im Entwicklungsprozess entdeckt werden. Infolgedessen können Konstrukteure ein robustes, sicheres und zuverlässiges Produkt ohne übermäßige Iterationen erstellen.

Eine einheitliche Embedded Software-Plattform für Automotive ist entscheidend für die Koordination einer komplexen Softwareanwendungsentwicklung, da sie Echtzeit- und On-Demand-Zugriffe auf Konstruktionsdaten, Anforderungen, Testergebnisse und viel mehr direkt in der Konstruktionsansicht bietet. Eine derartige Plattform leistet ihren Beitrag zu einem robusten digitalen roten Faden und ermöglicht eine organische und intuitive Zusammenarbeit zwischen örtlich getrennt arbeitenden Teams und Unternehmen und steigert so die Produktivität aller Beteiligten.

Zudem ermöglichen Lösungen wie Polarion die schnelle Anwendungsbereitstellung durch das Angebot von Vorlagen für Enterprise AGILE, SAFe und andere umfangreiche Implementierungen zusammen mit Vorlagen für Standardkonformität wie ISO 26262, A-SPICE, CMMI und anderen. Diese Vorlagen lassen sich an kundenspezifische Prozesse anpassen, wodurch die Anwenderakzeptanz vereinfacht und die Rendite beschleunigt wird (Abbildung 8).

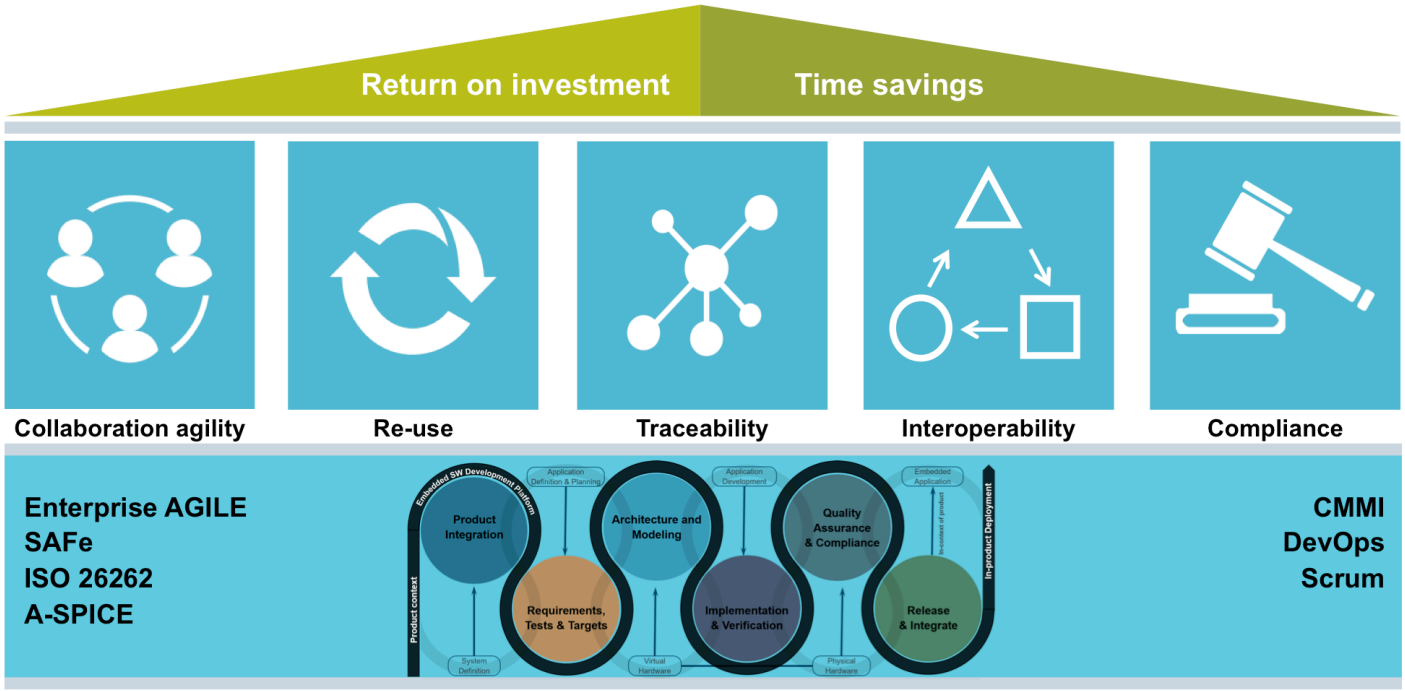

Abbildung 8: Eine einheitliche Entwicklungsplattform für Embedded Software ermöglicht Zusammenarbeit und Wiederverwendung, während gleichzeitig Rückverfolgbarkeit, Interoperabilität und Konformität gewährleistet werden. Standardvorlagen tragen dazu bei, Rendite und Entwicklungszeiten zu beschleunigen.

OEMs und Zulieferer erwarten von diesen komplexen Engineering-Umgebungen, in denen sie Konstruktionskosten reduzieren und die Effizienz steigern können, eine schnelle Rendite. Eine einheitliche Plattform für die Koordinierung der Softwareentwicklung bietet Flexibilität bei der Zusammenarbeit in einer konfigurierbaren Umgebung. Dadurch wird es einfacher und rentabler, agile Workflows zu ändern. Die Umgebung fördert die Wiederverwendung, da Unternehmen geprüfte Komponenten nutzen, um Varianten schneller zu produzieren, differenziertere Produkte und eine höhere Rendite zu ermöglichen. Zudem bieten diese Werkzeuge die dringend benötigte Nachverfolgbarkeit für Audits, Recherchen und Berichte, während Ressourcen, Zeit und verbundene Kosten gespart werden. Konstrukteure sind so in der Lage, sich auf Konstruktionsaufgaben zu konzentrieren, während mit den Tools die Arbeitsergebnisse nachverfolgt werden.

Fahrzeuginterne Software gewinnt für die Wettbewerbsfähigkeit von Fahrzeugen in einem überladenen und hart umkämpften Markt immer mehr an Bedeutung. Interessante und nützliche Funktionen wie ADAS und erweiterte Infotainmentsysteme verwenden in zunehmendem Maße hoch entwickelte Software, um die Funktionen und Merkmale zu bieten, die modernen Verbrauchern wichtig sind. Um sich gegenüber Wettbewerbern durchzusetzen, streben Hersteller danach, immer enger gesteckte Zeitvorgaben einzuhalten. Gleichzeitig steigen die Erwartungen an die Qualität und die Zuverlässigkeit der Software. Systemverzögerungen, Störungen und eine unzureichende Anwenderschnittstelle fallen umso stärker ins Gewicht, je wichtiger die Interaktion des Anwenders mit dem Fahrzeug wird. Eine einheitliche Umgebung für die Zusammenarbeit bei der Softwareentwicklung mit integrierter Rückverfolgbarkeit und Wiederverwendung geistigen Eigentums ist in dieser neuen Umgebung von unschätzbarem Wert.

Siemens Digital Industries Software

Hauptsitz
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Nord-, Mittel- und Südamerika
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europa
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asien-Pazifik
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

Über Siemens Digital Industries Software

Siemens Digital Industries Software, eine Business Unit von Siemens Digital Industries, ist ein weltweit führender Anbieter von Softwarelösungen, die den digitalen Wandel der Industrie vorantreiben und neue Möglichkeiten für Hersteller schaffen, Innovationen umzusetzen. Mit Hauptsitz in Plano, Texas, und mehr als 140.000 Kunden in aller Welt arbeiten wir mit Unternehmen jeder Größe zusammen, um die Art und Weise zu verändern, wie Ideen realisiert, Produkte und Anlagen entwickelt und sinnvoll eingesetzt werden. Weitere Informationen über unsere Produkte und Leistungen finden Sie unter www.siemens.com/plm.

siemens.com/plm

78394-82773-C6-DE 10/20 LOC