Automobile flexibel entwerfen

Automobile Steuerungssoftware in einem Rahmenwerk zur kontinuierlichen Integration. Die Komplexität von Automobilsteuerungssoftware steigt durch eine wachsende Anzahl von Funktionen und Varianten dieser Funktionen stetig an. Internationale Standards stellen zusätzliche Anforderungen an den Prozess der Softwareentwicklung.

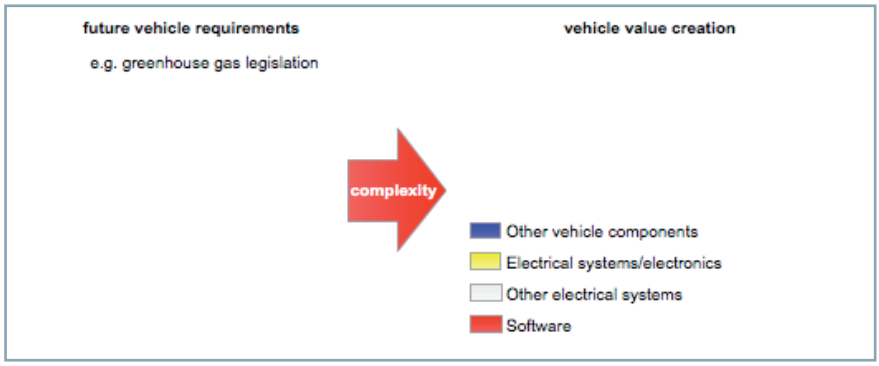


Bild 1: Anstieg der Relevanz von Software, bedingt durch zukünftige Anforderungen in der Automobiltechnik, Bild aus [Wyman, 2007, AutoIndustry Collaboration]

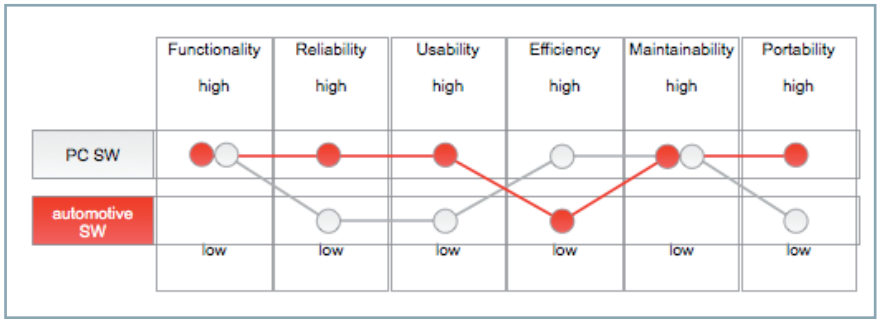


Bild 2: Qualitätseigenschaften von Embedded-Software von PC und Automobil gemäß ISO 25000

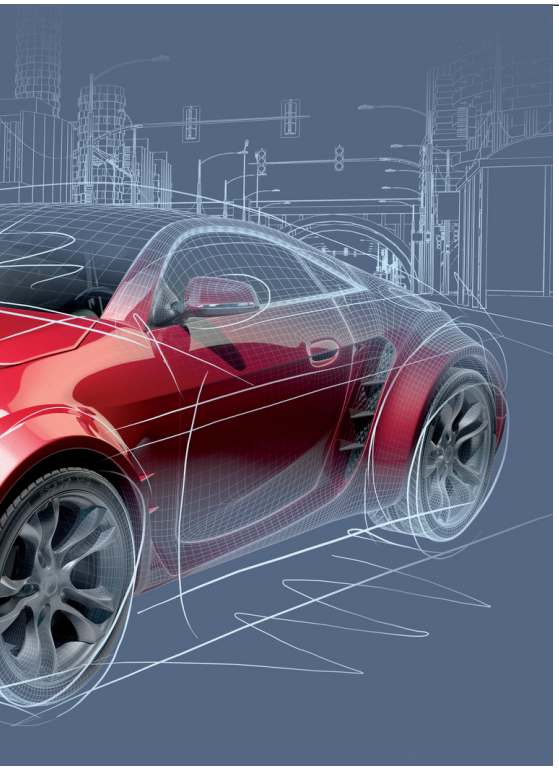
Dr. Philipp Orth, Dr. Axel Schlosser, Johannes Richenhagen

■ Im Bereich der Automobilsteuergeräte steht die Softwareentwicklung in vielerlei Hinsicht vor Herausforderungen. Aus Sicht der Märkte benötigen eine steigende Zahl von Produktvarianten und kurze Entwicklungszyklen eine schnelle Entwicklung neuer Funktionen, die durch den Kunden individuell zusammengestellt werden können [1]. Zusätzlich machen Gesetzesänderungen hinsichtlich der zukünftigen automobilen Fortbewegung die Entwicklung von Antrieben mit immer geringeren Verbrauchs- und Abgasausstoßwerten notwendig. Um sich diesen Bedürfnissen anzupassen, wurden neue Antriebstechnologien entwickelt. Auf der einen Seite werden alternative Antriebstechnologien durch Industrie und Forschungsinstitute erforscht. Dazu gehören verschiedene elektrifizierte Konzepte, innovative Konstruktionsstrategien für konventionelle Verbrennungsmotoren und die Erforschung alternativer Brennstoffe, die aus Biomasse gewonnen werden und auf den Verwendungszweck abgestimmte Verbrennungseigenschaften besitzen. Auf der anderen Seite wird die Entwicklung verschiedener technischer Lösungen durch die Gesetzgebung und staatliche Aktivitäten vorangetrieben. Unter der Annahme, dass die chemischen Eigenschaften von Biomasse bei der Verbrennung bestimmte Vorteile zur Folge haben, wird der Marktanteil von Biomasse im Jahr 2020 zehn Prozent in der Europäischen Union und 13 Prozent in den USA betragen. Die Europäische Kommission hat sich auf einen Plan geeinigt, mit dem die Anzahl der Elektrofahrzeuge schrittweise auf fünf Millionen im Jahr 2020 angehoben werden soll. Die Regierung der USA plant die Investition von zwei Milliarden

KONTAKT

MathWorks
Adalperostraße 45
85737 Ismaning
Tel.: +49-89-45235-6700
Fax: +49-89-45235-6710
www.mathworks.de

© IGT Verlag, München



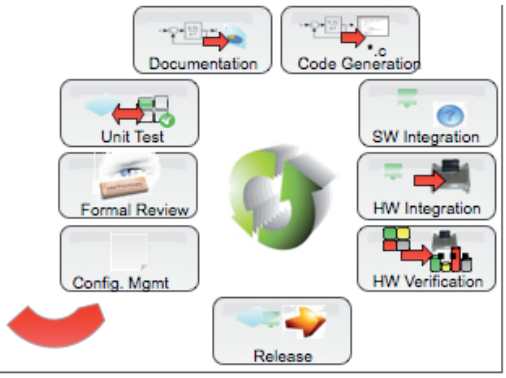


Bild 3: Systemarchitektur des Integrationsrahmenwerks „Nightly Build“

US-Dollar, um die Entwicklung von wieder aufladbaren Autobatterien und Elektrofahrzeugen voranzutreiben. Mit der Einführung elektrischer Komponenten mit Hochspannungsversorgung müssen bei der Entwicklung von Steuergeräten für elektrische Automobilantriebe auch die entsprechenden Sicherheitsmaßnahmen beachtet werden.

Daraus ergeben sich verschiedene Herausforderungen für die Softwareentwicklung. Die Vielfalt von Antriebskonfigurationen und die Anzahl der Anforderungen hinsichtlich der Sicherheitsfunktionen steigen stetig an. Systeminterne Komplexität entsteht durch anspruchsvollere Systemkonzepte (zum Beispiel für Motoren, Antrieb, Batterien). Die systemübergreifende Komplexität wird durch eine stärkere Interaktion ehemals getrennter Systeme erzeugt (zum Beispiel Antriebselektrifizierung, Fahrzeugdynamik-Sicherheitsfunktionen, Komfortausstattung). Zusätzlich wird davon ausgegangen, dass Software zu einem immer wichtigeren Wettbewerbsfaktor auf dem Automobilmarkt wird. Beispielsweise erwarten Experten bei Daimler, dass 80 Prozent der zukünftigen Automobilinnovationen durch Software vorangetrieben werden. Die effiziente Entwicklung von Automobilsoftware wird mehr und mehr zu einem Hauptumsatzfaktor (Bild 1).

Um Funktionsumfang und Sicherheitsanforderungen hinsichtlich der Seriensoftware zu erfüllen, ist der Entwicklungsprozess durch Referenzmodelle wie Automotive SPICE, CMMI oder den Prozessstandard ISO 26262 standardisiert. Besonders die Norm ISO 26262 definiert umfassende Qualitätssicherungsmaßnahmen, entsprechend etablieren sich richtlinienkonforme und zertifizierte Arbeitsprozesse. Die Projektkosten der Entwicklung von Steuersoftware werden anstelle von funktionalen Herausforderungen durch effiziente Prozess-abwicklung getrieben. Der Aufwand für die Softwareintegration, die aus dem Design der Softwarearchitektur, V&V-Aktivitäten (Verifikation und Validierung) und der Codeintegration auf der Zielhardware besteht, steigt ständig an. Daher wird diese nur für wenige Meilensteine durchgeführt, weshalb Fehler und problematische Sachverhalte erst spät erkannt werden. Zusätzlich führt dies zu steigendem Integrationsaufwand und abnehmender Softwarequalität. Ausfallstatistiken belegen, dass das Versagen von Software ein erhebliches Risiko bleibt. Rund 27 Prozent aller Fahrzeugrückrufe sind auf elektronische Fehler zurückzuführen. 20 Prozent aller Funktionsstörungen in Autos werden durch Softwarefehler verursacht

Um entsprechende Gegenmaßnahmen einzuleiten, ist ein integrierter Ansatz bei der Qualitätssicherung notwendig. Die möglichen Methoden zur Verifikation und Validierung müssen in eine Teststrategie integriert werden. Außerdem muss diese Strategie in ein agiles Rahmenwerk integriert sein, das einen minimalen manuellen Testaufwand erfordert sowie fortlaufende Softwareüberprüfung und Nachvollziehbarkeit von Testergebnissen über das gesamte Projekt ermöglicht, um die frühzeitige Behebung von Softwarefehlern zu realisieren.

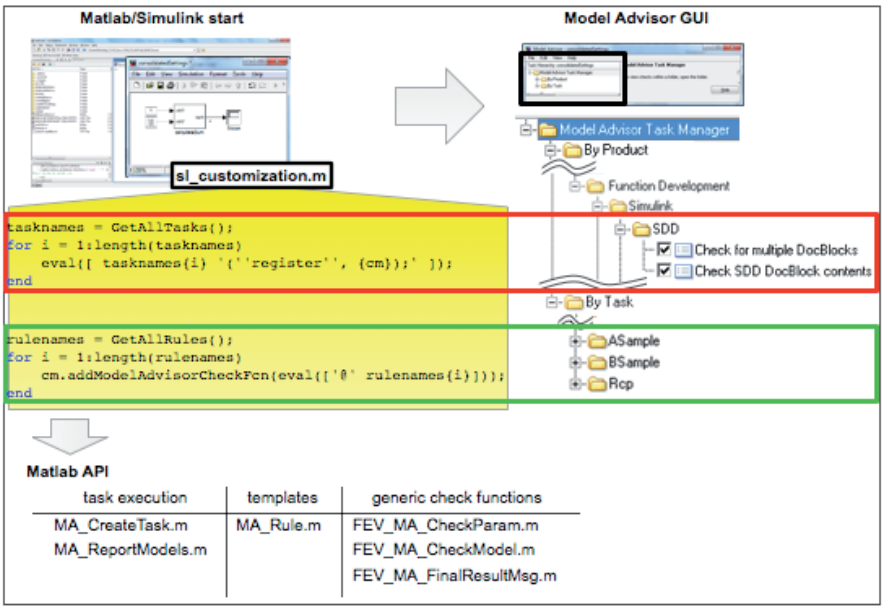


Bild 4: Rahmenwerk der Richtlinienüberprüfung für anwenderdefinierte Verifikation mit Simulink-Model-Advisor

Agile Softwareentwicklung in der Automobilbranche

Der Begriff agile Softwareentwicklung wurde im Bereich der Softwareentwicklung von Anwendungen für Personal Computer (PC) von Fowler geprägt. Diese Definition dient als Basis zur Definition von Anforderungen für ein agiles Integrationsrahmenwerk im Automobilbereich. Nach Fowler setzt sich die agile Softwareentwicklung aus vier Priorisierungsprinzipien zusammen:

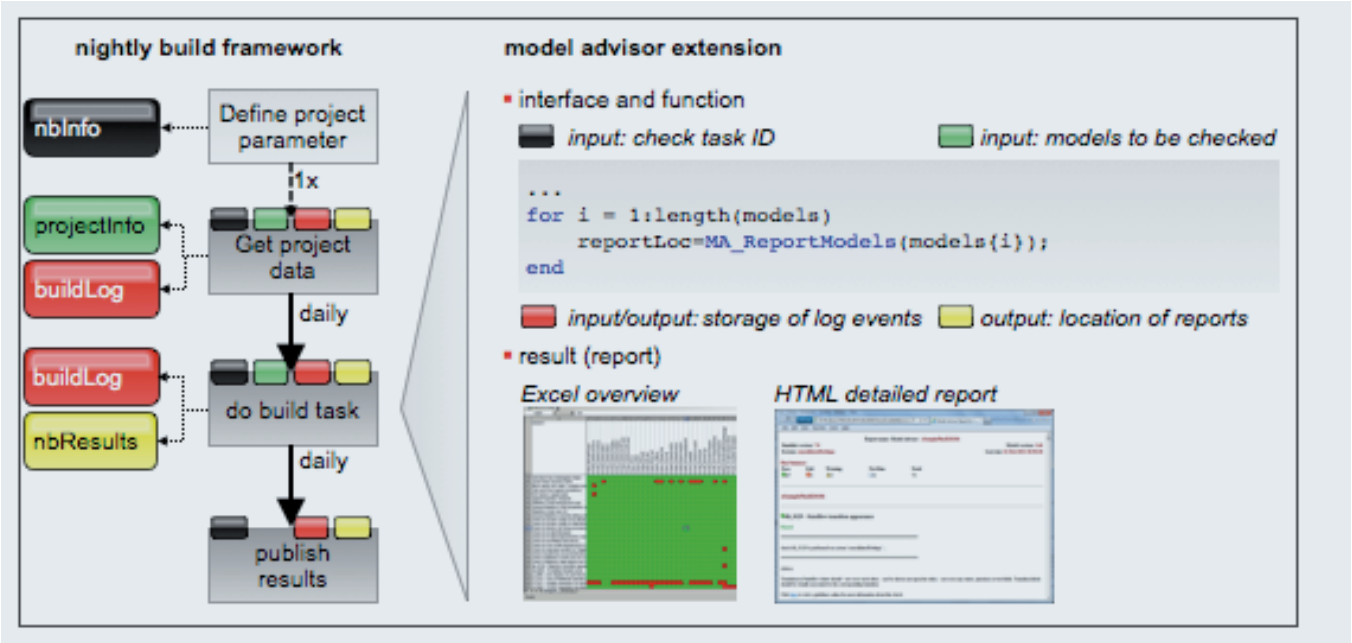


Bild 5: Anpassung des Nightly-Build-Rahmenwerks für Richtlinienüberprüfungen

- Menschen und Interaktionen sind wichtiger als Prozesse und Werkzeuge,
- funktionierende Software ist wichtiger als umfassende Dokumentation,
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen und
- Eingehen auf Veränderungen ist wichtiger als Festhalten an einem Plan.

Diese Prinzipien stärken solche Strategien, die es dem Entwickler und Projektmanager ermöglichen, sich auf technische Lösungen zu konzentrieren, mit dem Kunden zusammenzuarbeiten und diese Aufgaben zu priorisieren, anstatt vorwiegend Prozesskonformität zu fokussieren. Die dargestellten Prozessanforderungen der Automobilbranche scheinen diesem Prinzip zu widersprechen. Dieser Beitrag stellt ein agiles Integrationsrahmenwerk vor, das darauf abzielt, dieses Dilemma zu lösen, indem der Prozess durch Steigerung des Automatisierungsgrades gestützt und alle Entwicklungsschritte integriert sowie die Operationen in regelmäßigen Abständen ausgeführt werden. Kontinuierliche Integration (Continuous Integration, CI) wird in diesem Rahmen als adäquate Methode für die Automobilentwicklung angewandt. Sie wird als „ein Vorgehen in der Softwareentwicklung“ definiert, „in der die Mitglieder eines Teams ihre Arbeit regelmäßig integrieren [...], was zu mehreren Integrationen am Tag führt. Jede Integration wird durch einen automatisierten Build bestätigt (einschließlich eines Tests), um Integrationsfehler so früh wie möglich zu ermitteln“. Die Ziele der Risikominderung dieses Konzepts passen zu den Herausforderungen, die sich in der Automobilindustrie stellen:

- Vermeiden einer späten Erkennung von Softwarefehlern und daraus folgenden Änderungen im Zeitplan,
- Vermeiden von Unsicherheiten bezüglich des Reifegrads der Software und
- Vermeiden der Bereitstellung fehlerhafter Software.

Die Umsetzung von CI wird zum einen durch einen hohen Automatisierungsgrad (Softwaretests, Integrationsmaßnahmen) erreicht, wodurch eine hohe Anzahl an Ausführungen (zum Beispiel wöchentlich, täglich) möglich ist. Zum anderen wird hierzu die Definition aussagekräftiger Softwaremetriken benötigt, welche die Qualitätsmerkmale des Codes erfassen.

Da die Automatisierung bei der Anwendung von CI eine große Rolle spielt, werden entsprechende Rahmenwerke – auch als Dashboard-Toolkit bekannt – für die Softwareentwicklung mit textbasierten Sprachen geschaffen. Beispiele dafür sind die Axivion-Bauhaus-Suite oder das Open-Source-Projekt QALab. Diese Dashboards ermöglichen Integrationsschritte wie Unit-Test, Softwaredokumentation, Beurteilung der Softwaremetrik, Build und Bericht der Integrationsergebnisse. Eine Übertragung dieser Dashboards auf den Entwicklungsprozess von Automobilsoftware ist nicht direkt möglich, weil die Qualitätsansprüche an Echtzeitsysteme in der Automobilindustrie andere sind als an PC-Software. Um diesen Unterschied genauer untersuchen zu können, wird das Qualitätsmodell angewandt, das durch die Qualitätseigenschaften aus der Norm ISO 25010 definiert ist. Die Eigenschaften der Dashboard-Funktionen

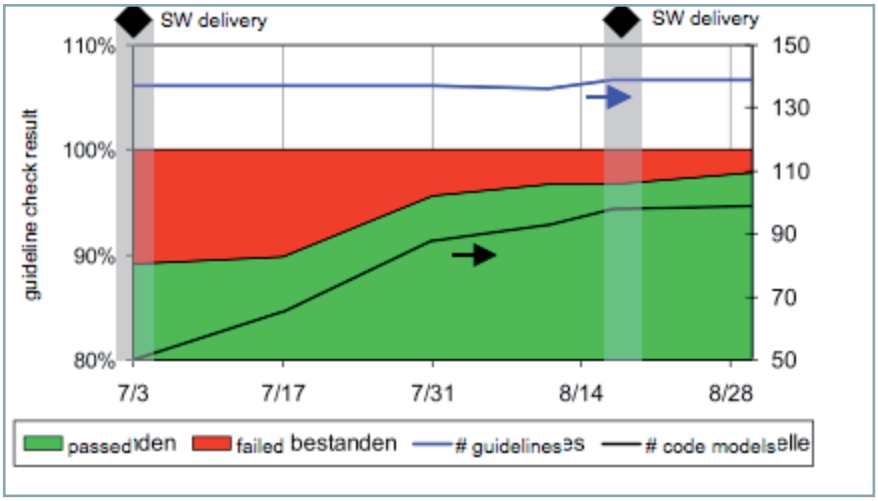


Bild 6: Tägliches Nachverfolgen der Ergebnisse der Richtlinienüberprüfung für ein Seriensoftware-Entwicklungsprojekt

© IGT Verlag, München

sind im Vergleich zu bestimmten PC-Anwendungen ähnlich, aber doch andersartig (beispielsweise Sicherheitslösungen gegen das Eindringen bösartiger Programme (Malware) im Vergleich mit der Genauigkeit von Steuerungsgrößen). Hohe Verlässlichkeitsanforderungen ergeben sich aus komplexen Fahrzeugnetzwerken. Sicherheitskritische Systeme müssen fehlertolerant entwickelt werden. Die Brauchbarkeit der Anwendbarkeit ist sehr beschränkt, da die Antriebssteuerung mit nur sehr wenigen Nutzereingaben auskommt, wie beispielsweise Gaspedal oder Schalthebel, obwohl elektronische Systeme dies zukünftig ändern können. Aufgrund des Echtzeitbetriebs und der potenziell hohen Hardwarekosten muss automobile Steuerungssoftware eine rechtzeitige Antwort auf Ereignisse (Verhalten in Echtzeit) mit minimalem Speicherbedarf und minimaler Prozessorleistung sicherstellen. Die Anforderungen an die Wartbarkeit ergeben sich aus der Verwendung ähnlicher Softwareversionen in verschiedenen Produktlinien verschiedener Fahrzeuggenerationen. Weil Steuerungssoftware auf eine spezifische Hardware zugeschnitten wurde, kann die Portierbarkeit nicht mit PC-Anwendungen verglichen werden. Außerdem wird Software zunehmend durch Model-Based-Design (MBD) entwickelt. Das physikalische Systemverhalten und die Steuerungsalgorithmen werden durch domänenspezifische Modellsprachen dargestellt. Der C-Code, der für den Embedded-Target-Prozessor kompiliert wird, wird aus Modellen automatisch generiert, weshalb sich die Menge an handgeschriebenem Code verringert. V&V-Messungen werden, soweit möglich, auf Modellebene und nicht direkt auf Programmcode-Ebene durchgeführt. In Bild 2 werden diese Unterschiede qualitativ und gemäß den in der ISO-Norm 25010 definierten Qualitätskategorien zusammengefasst.

Für Automobil- und PC-Anwendungen unterscheiden sich Qualitätsanforderungen und Entwicklungsmethoden. Die Qualität wird durch verschiedene Softwaremetriken und Integrationsmessungen sichergestellt. Hinsichtlich der Entwicklung werden Softwareinformationen durch verschiedene Maßnahmen bewertet. Die Verifikations- und Validierungsmessungen müssen auf verschiedenen Abstraktionsebenen durchgeführt werden. Daher müssen das CI-Rahmenwerk und die Integrationsmessungen auf die Automobilbranche zugeschnitten werden. Eine direkte Anwendung von Tools, die auf PC-Software zugeschnitten sind, ist nicht möglich. Daher deckt keines der bereits existierenden Integrationstools alle Anforderungen im Automobilbereich ab. Auf der einen Seite werden Funktionen oder passende Schnittstellen für die durchzuführenden Integrationstasks nicht angeboten. Auf der anderen Seite sind existierende Tools durch eine abgeschlossene Werkzeugumgebung und heterogene Schnittstellen zur Anwendungsprogrammierung (API) gekennzeichnet, die die Anwendung für automobile Softwareentwicklung erschweren.

Framework für die kontinuierliche modellgestützte Integration

Die Toolkits für die CI bei der PC-Softwareentwicklung definieren Softwareintegration als eine Kombination verschiedener formeller und funktionaler Qualitätsprüfungen und des Softwarebuild. Bei der Entwicklung von Automobilsoftware verstehen man Integration auf ähnliche Weise als einen Prozess, der Maßnahmen zur Verifikation und Validierung (V&V), Integration und den Softwarebuild des gewünschten Targets beinhaltet. Die Anforderungen bezüglich des Integrationsrahmenwerks werden von den Vorgaben der Automobilentwicklung abgeleitet. Für die Durchführung von MBD müssen architektonische Konformitätstests, syntaktische Softwareverifikation (zum Beispiel Modellrichtlinien), funktionale Modultests, Integrationstests, Konfiguration von Integrationselementen, Testplanung über den gesamten Lebenszyklus der Software und Dokumentation und Nachvollziehbarkeit aller Testergebnisse auf Modellebene durchgeführt werden. Für eine bessere Agilität der Entwicklung hinsichtlich stetiger Überprüfung und des Build sind die Anforderungen an die Integrationsfunktion dieselben wie für die bereits existierenden CI-Tools im Bereich für PC-Software. Alle Überprüfungen müssen automatisiert werden, Modellklone können erkannt werden und die Qualität und der Reifegrad der Software werden anhand von generierten Metriken gemessen. Prozessziele werden von etablierten industriellen Verfahren abgeleitet: Codegenerierung aus dem Modell, Kompilierung und Bereitstellung des Maschinencodes auf der gewünschten Zielhardware sowie das Erstellen eines auslieferbaren Gesamtpaketes (ausführbare Software, Dokumentation, Installationsprogramm).

Für die Funktionsentwicklung bei FEV und VKA sind sowohl eine flexible Anpassung auf verschiedene Tools und Entwicklungsumgebungen als auch verschiedene Softwarearchitekturen und Programmierrichtlinien nötig. Beides hängt von der Entwicklungsumgebung des Kunden, dem gewünschten Softwarereifegrad und dem Projektumfang ab. Dies beinhaltet auch die Anforderung, dieselben Tools mit dem CI-Rahmenwerk anzuwenden, die von Ingenieuren für Funktionsspezifikation und Softwareentwicklung eingesetzt werden. Dadurch werden parallele Toolketten für Integration und Entwicklung vermieden und ein nahtloses Hinzufügen neuer Integrationsfunktionen auf Basis der Bedürfnisse des Entwicklers sichergestellt. Das Rahmenwerk für CI in einer Modell-basierten Softwareentwicklungsumgebung wurde anhand dieser Anforderungen entworfen. Auf der höchsten Abstraktionsstufe ähnelt die Systemarchitektur einem CI-Rahmenwerk für textbasierte Softwareentwicklung. Entwickler an verschiedenen Standorten speichern ihre Arbeitsergebnisse in einer versionierten Datenbank ab. Ein Integrationsserver ruft fortlaufend den aktuellen Softwarestand aus dieser Datenbank ab und führt die Integrationsoperationen mit einer Build-Routine aus. Aufgrund der kontinuierlichen Ausführung dieser Schritte über Nacht wird das Rahmenwerk auch als „Nightly Build“ bezeichnet (Bild 5).

Im Gegensatz zu CI-Rahmenwerken für PC-Software, die beispielsweise auf Programmiersprachen C oder C# basieren, werden die Integrationsoperationen mit MATLAB-Skripts durchgeführt. Diese Technologie wurde aus verschiedenen Gründen ausgewählt:

- Die Integrationsoperationen können mit denselben Werkzeugen durchgeführt werden, die schon von jedem einzelnen Entwickler verwendet wurden. Damit lassen sich beispielsweise die Modellparametrierung, Richtlinienprüfung, Data-Dictionary-Verwaltung, Codegenerierung und Kompilierung durchführen. In umgekehrter Richtung können Entwickler Build-Operationen ausführen, um zu überprüfen, ob die Modelle mit den Implementierungsrichtlinien konform sind. Redundante Toolimplementierung wird vermieden.
- Matlab umfasst eine Befehlszeile und eine COM/.NET-Schnittstelle für eine bequeme Integration von Drittanbietersoftware, wie zum Beispiele Werkzeuge für das Anforderungsmanagement, die Fehlerverfolgung oder die Versionsverwaltung.
- Der Integrationsprozess kann über die Kommandozeile von Matlab aufgerufen werden, beispielsweise durch Task-Planungsprogramme.

Aus dieser Entscheidung resultieren auch Nachteile. Im Vergleich zu Überprüfungen, die auf dem Parsen der Simulink-Modelldatei als Textdatei basieren erhöhen sich die Ausführungszeit und die Inanspruchnahme der Hardwareressourcen. Dies kann durch paralleles Rechnen und einen Hochleistungs-Buildserver abgeschwächt werden. Zusätzlich ist das Integrationsrahmenwerk nur mit der Werkzeugumgebung Matlab/Simulink einsatzfähig. Der erste Nachteil kann durch die Parallel-Computing-Toolbox gelöst werden. Der zweite Punkt resultiert daraus, dass die Produkte von MathWorks sehr häufig zur modellbasierten Entwicklung in der Automobilsoftwareentwicklung verwendet werden.

Kontinuierliche Richtlinienüberprüfung in der Seriensoftwareentwicklung

Um eine hohe Flexibilität bei der Konfiguration der Integration für verschiedene Projekte, Kunden, Implementierungsrichtlinien und Tools zu erreichen, wurde die Integrationssoftwarearchitektur modular strukturiert. Um eine gesteigerte Flexibilität der Toolkette bei der Funktionsentwicklung und eine Trennung der Belange zu ermöglichen, sind das Rahmenwerk für Richtlinienüberprüfung (GCF, guideline check framework) und das CI-Rahmenwerk verschiedene Entitäten, die miteinander interagieren. Das GCF basiert auf dem Simulink-Model-Advisor von MathWorks (Bild 5). Beim ersten Starten von Simulink oder bei der Registrierung von Add-Ons für Simulink (zum Beispiel Rapid-Control-Prototyping-Tools von Drittanbietern) während des Matlab-Programmstarts wird die Funktion sl_customization.m des GCF ausgeführt. Diese Funktion nutzt die Schnittstelle von Simulink für Plugin-ähnliches Customizing sowohl um die Regeln zur Richtlinienüberprüfung zu registrieren, als auch für Tasks, die die Checklisten für verschiedene Softwareentwicklungsprojekte und Reifegradebenen definieren. Ein Vorteil dieser Schnittstelle besteht darin, dass andere Tools, die ebenfalls Richtlinienprüfungen anbieten, ihre Richtlinien ohne Beeinträchtigung für eine der beiden Seiten ebenfalls registrieren können.

Wenn während der Modellierung über die Benutzeroberfläche auf Model-Advisor zugegriffen wird, werden die hinzugefügten Tasks und Regeln angezeigt und können mittels der Reportroutine des Model-Advisors ausgeführt werden. Für den programmatischen Zugriff auf Richtlinienüberprüfungen bietet das Matlab-API der GCF-Funktionen, um die Tasks zu definieren und auszuführen. Außerdem können anhand von gespeicherten Vorlagen und generischen Funktionen neue Regeln programmiert werden, beispielsweise Parameterprüfungen von Modellen oder deren Blöcken. Die dargestellten Funktionen zur Richtlinienüberprüfung sind über das entsprechende Matlab-API in das CI-Rahmenwerk integriert. Dieses besteht aus vier Elementen. Zu Beginn werden Projektparameter wie Namenskonventionen, Release-Plan oder Richtlinienchecklisten jeweils einmal für das Entwicklungsprojekt definiert. Basierend auf diesen Parametern werden die Projektdaten (Modelle, Code, m-Dateien und so weiter) identifiziert. Abhängig von den Projektanforderungen können verschiedene Build- oder Integrations-Tasks mit den üblichen Schnittstellen erstellt werden. Eine generische Berichtsroutineveröffentlicht die generierten Berichte (z.B. zu einer Subversion-Datenbank, in eine Datei oder auf einen Webserver).

Richtlinienüberprüfungen sind als CI-Routine implementiert (Bild 5). Ihre Eingangswerte sind die Kennungen der Richtlinienliste sowie eine Liste von Modellen und ihren Parametern, die überprüft werden sollen. Danach wird die Berichtsfunktion der Routine ausgeführt. Im Vergleich zu bereits existierenden Lösungen von Drittanbietertools oder eingebauten Routinen zum Ausführen von Überprüfungen besteht der Hauptnutzen dieser Funktion in der Erstellung eines Übersichtsberichts für alle geprüften Modelle, indem die Modelle und ausgeführten Prüfungen als Matrix anzeigt werden. Systematische Fehlermeldungen, die aufgrund fehlerhafter Modellierungskonventionen oder Prüfregeln mit Fehlern aufgetreten sind, können durch Spalten mit einer hohen Anzahl nicht bestandener Prüfungen identifiziert werden. Fehlerhafte Modelle können durch Spalten mit einer hohen Anzahl nicht bestandener Prüfungen identifiziert werden. Die eingebaute Routine für die Erzeugung eines HTML-Berichts wird weiterhin ausgeführt und enthält detaillierte Informationen bezüglich aller Prüfergebnisse.

Um das Potenzial kontinuierlicher Integration in der Serienentwicklung einschätzen zu können, wurde das Integrationsrahmenwerk in der Regelungsentwicklung für ein Abgasnachbehandlungssystem angewandt. Sicherheitsrelevante und für die Borddiagnose wichtige Funktionen wurden in eine konfigurierbare Steuerungssoftware für unterschiedliche Zubehörmärkte implementiert. Nach der Definition von Kundenanforderungen wurden die abgeleiteten Softwareanforderungen zusammen mit den festen Implementierungsrichtlinien in einen Freigabeplan überführt, der bestimmt, welche Prüfungen von welchen Modulen zu welchem Zeitpunkt während des Entwicklungszyklus durchgeführt werden. Das Softwareteam und der Projektmanager können den Reifegrad der Software täglich überprüfen. Bild 6 zeigt einen Ausschnitt aus dieser historischen Nachverfolgung. Es ist zu erkennen, dass die Richtlinienkonformität direkt mit der Qualität der Modelle hinsichtlich ihrer Verwendbarkeit zur Codegenerierung korreliert. Die dargestellte Abbildung ermöglicht es dem Projektmanager, die Produktkonformität zu den zugehörigen Qualitätsanforderungen hinsichtlich einer ordnungsgemäßen Implementierung nachzuverfolgen. Wenn die Erfüllung der Anforderungen der Projektmeilensteine überprüft wird, können mögliche Risiken mit einer maximalen Verzögerung von einem Tag entdeckt werden. Außerdem wird der erhebliche Arbeitsumfang in Summe zeitaufwendiger und dadurch kostspieliger Richtlinienüberprüfungen automatisch vom Integrationsserver durchgeführt.

In diesem Artikel wurde auf die Möglichkeit des CI-Rahmenwerks, weitere Integrationsanforderungen zu erfüllen (zum Beispiel Testen von Modulen, Modelldokumentation), nicht eingegangen. Jedoch können Aspekte wie die Konformität von Funktionsschnittstellen, zeitliches Verhalten oder Ressourcennutzung im Falle eines Anschlusses an die Hardware noch nicht gemessen werden. Zu diesem Zweck werden aktuell weitere CI-Routinen entwickelt, um den Nutzen für die Softwareentwicklung von Steuergeräten zu vergrößern. Außerdem ist die Leistung des Systems für Projekte mit vielen Modelldaten ausschlaggebend. Eine Optimierung des Ressourcenbedarfs für die Ausführung der Funktionen zur kontinuierlichen Integration ist ein weiteres Feld für zukünftige Optimierungen. *(wp)* ■

Autoren

Johannes Richenhagen ist am Institute for Combustion Engines an der RWTH Aachen University beschäftigt, Dr. Philipp Orth und Dr. Axel Schloßer sind Mitarbeiter der FEV GmbH in Aachen

© IGT Verlag, München