Agil und modellbasiert

Automobilbauinnovation durch Software. Eine agile und modellbasierte Entwicklung der Automobilsoftware verlangt kontinuierliche Integration. Ein Rahmenwerk unterstützt durch aktuelle Informationen zum Reifegrad der Software die Entwicklung für den gesamten Lebenszyklus der Software.

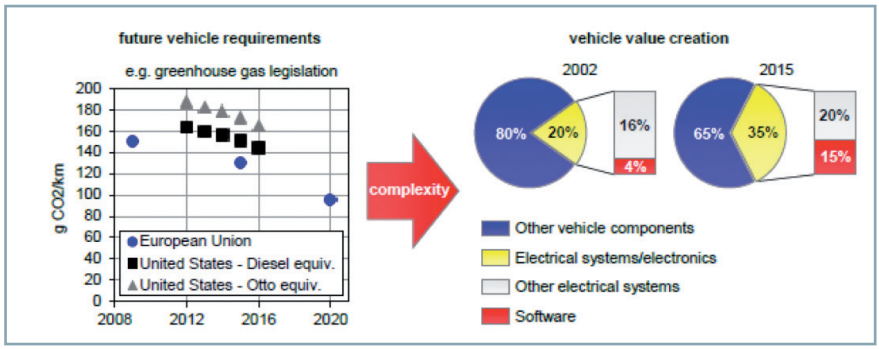

Bild 1: Anstieg der Relevanz von Software, bedingt durch zukünftige Anforderungen in der Automobiltechnik (Wyman 2007, Auto Industry Collaboration und [2], [,3])

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 [2], [3].

KONTAKT

Institut für Verbrennungskraftmaschinen
RWTH Aachen
Schinkelstr. 8
52062 Aachen
Tel. +49 241 80 95350
richenhagen@vka.rwth-aachen.de
www.rwth-aachen.de

Neue Antriebstechnologien

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 [4], [5], innovative Konstruktionsstrategien für konventionelle Verbrennungsmotoren [1] und die Erforschung alternativer Brennstoffe, die aus Biomasse gewonnen werden und auf den Verwendungszweck abgestimmte Verbrennungseigenschaften besitzen [6]. 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 [7], [8], wird der Marktanteil von Biomasse im Jahr 2020 10 Prozent in der Europäischen Union und 13 Prozent in den USA betragen [3], [5], [7]. Die Europäische Kommission hat sich auf einen Plan geeinigt, mit dem die Anzahl der Elektrofahrzeuge schrittweise auf fünf Millionen im Jahr 2020 [10] angehoben werden soll. Die Regierung der USA plant die Investition von zwei Milliarden US-Dollar, um die Entwicklung von wiederaufladbaren Autobatterien und Elektrofahrzeugen voranzutreiben [11]. Mit der Einführung elektrischer Komponenten mit Hochspannungsversorgung müssen bei der Entwicklung von Steuergeräten für elektrische Automobilantriebe die entsprechenden Sicherheitsmaßnahmen beachtet werden [12].

Herausforderungen für die Softwareentwicklung

Die Vielfalt von Antriebskonfigurationen und die Anzahl der Anforderungen hinsichtlich Sicherheitsfunktionen steigen. Systeminterne Komplexität entsteht durch anspruchsvollere Systemkonzepte – beispielsweise für Motoren, Antrieb, Batterien. Die systemübergreifende Komplexität wird durch eine stärkere Interaktion ehemals getrennter Systeme erzeugt – zum Beispiel Antriebselektrifizierung, Fahrzeugdynamik-Sicherheitsfunktionen, Komfortausstattung [14]. 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 [15]. Die effiziente Entwicklung von Automobilsoftware wird mehr und mehr zu einem Hauptumsatzfaktor (Bild1).

Um Funktionalität und Sicherheitsanforderungen hinsichtlich der Seriensoftware zu erfüllen, ist der Entwicklungsprozess durch Referenzmodelle wie Automotive SPICE, CMMI oder den Prozessstandard ISO 26262 standardisiert [15 bis 17]. Besonders die Norm ISO 26262 definiert umfassende Qualitätssicherungsmaßnahmen, entsprechend etablieren sich richtlinienkonforme und zertifizierte Arbeitsprozesse [18], [19]. Die Projektkosten der Entwicklung von Steuersoftware werden anstelle von funktionalen Herausforderungen durch effiziente Prozessabwicklung

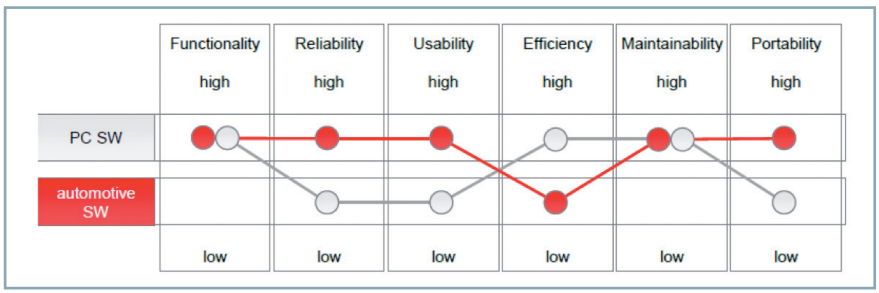

Bild 2: Qualitätseigenschaften von Embedded Software von PC und Automobil gemäß ISO 25000

getrieben [1], [21], [22]. 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 [23]. 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 SW-Qualität [24], [25]. Ausfallstatistiken belegen, dass das Versagen von Software ein erhebliches Risiko bleibt. Etwa 27 Prozent aller Fahrzeugrückrufe sind auf elektronische Fehler zurückzuführen [26]. 20 Prozent aller Funktionsstörungen in Autos werden durch Softwarefehler verursacht [27].

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 [22]. 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 [25].

Agile Softwareentwicklung für Automotive

Der Begriff Agile Softwareentwicklung wurde im Bereich der Softwareentwicklung von Anwendungen für Personal Computer (PC) von Fowler geprägt [28]. 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:

- Menschen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen
- 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.

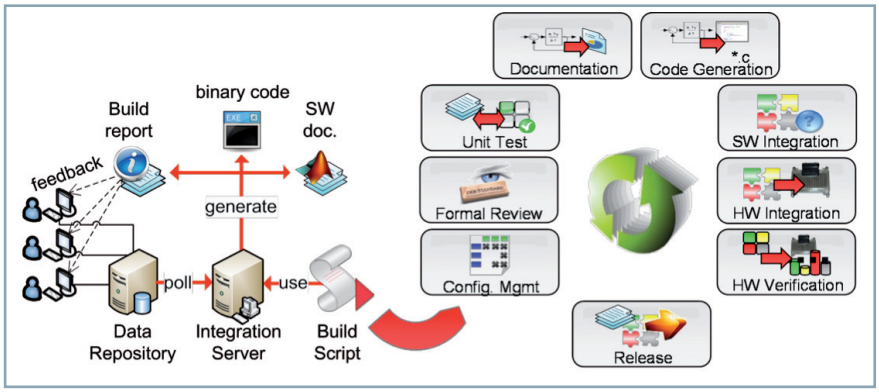

Bild 3: Systemarchitektur des Integrationsrahmenwerks „Nightly Build“ [34]

Kontinuierliche Integration (Continuous Integration, abgekürzt 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“ [29]. 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
- 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 (z. B. wöchentlich, täglich) möglich ist. Zum anderen wird hierzu die Definition aussagekräftiger Softwaremetriken benötigt, welche die Qualitätsmerkmale des Codes erfassen [30].

Softwareentwicklungs-Rahmenwerke

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 [25]. Beispiele dafür sind die Axivion Bauhaus Suite [31] oder das Open-Source-Projekt QALab [32]. 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

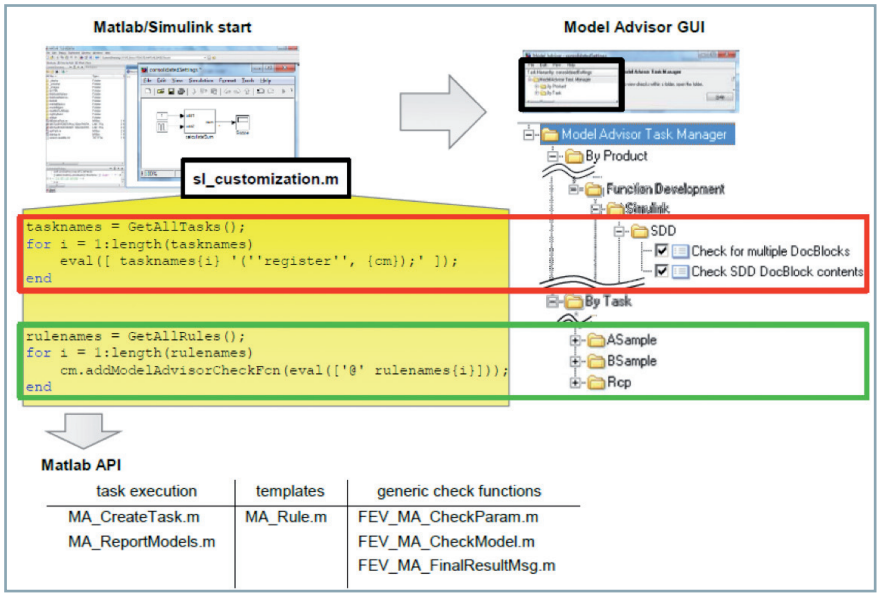

Bild 4: Rahmenwerk der Richtlinienüberprüfung für anwenderdefinierte Verifikation mit Simulink Model Advisor

PC-Software. Um diesen Unterschied genauer untersuchen zu können, wird das Qualitätsmodell angewandt, das durch die Qualitätseigenschaften aus der Norm ISO 25010 definiert ist [33]. Die Eigenschaften der Dashboard-Funktionalität ist 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 [1]. Die Brauchbarkeit der Anwendbarkeit ist sehr beschränkt, da die Antriebssteuerung mit nur sehr wenigen Nutzereingaben auskommt, wie z.B. 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 [1]. 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.

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) [34] gekennzeichnet, die die Anwendung für automobile SW-Entwicklung erschweren.

Framework für die 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 wir 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 (z. B. 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 (Abbildung 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 Buildoperationen 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 Werkzeuge für das Anforderungsmanagement, die Fehlerverfolgung oder die Versionsverwaltung. Der Integrationsprozess kann über die Kommandozeile von Matlab aufgerufen werden, zum Beispiel durch Taskplanungsprogramme.

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. Das folgende Kapitel beschreibt beispielhaft die Einbindung der Modellierungsrichtlinienprüfungen in das CI-Rahmenwerk.

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 (Abbildung 5). Beim ersten Starten von Simulink oder bei der Registrierung von Add-Ons für Simulink (beispielsweise 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 Richtlinienchecklist jeweils einmal für das Entwicklungsprojekt definiert. Basierend auf diesen Parametern werden die Projektdaten (Modelle, Code, m-Dateien etc.) identifiziert. Abhängig von den Projektanforderungen können verschiedene Build- oder Integrationstasks mit den üblichen Schnittstellen erstellt werden. Eine generische Berichtsroutine veröffentlicht die generierten Berichte (zum Beispiel zu einer Subversion-Datenbank, in eine Datei oder auf einen Webserver).

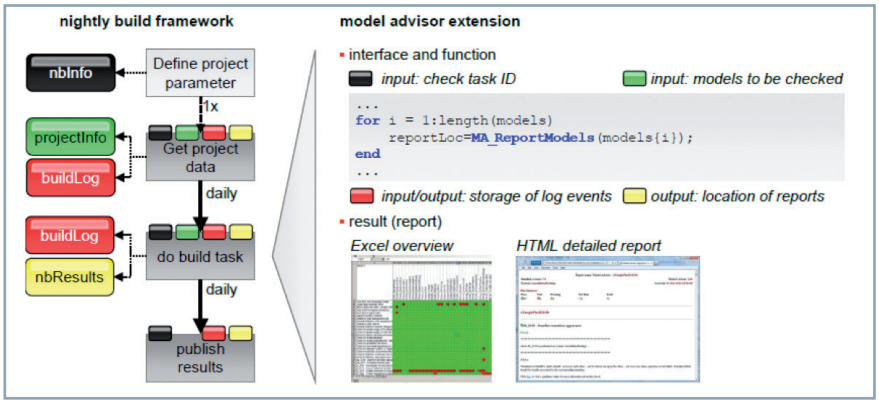

Bild 5: Anpassung des Nightly-Build-Rahmenwerks für Richtlinienüberprüfungen

Richtlinienüberprüfungen sind als CI-Routine implementiert (Abbildung 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. Abbildung 7 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

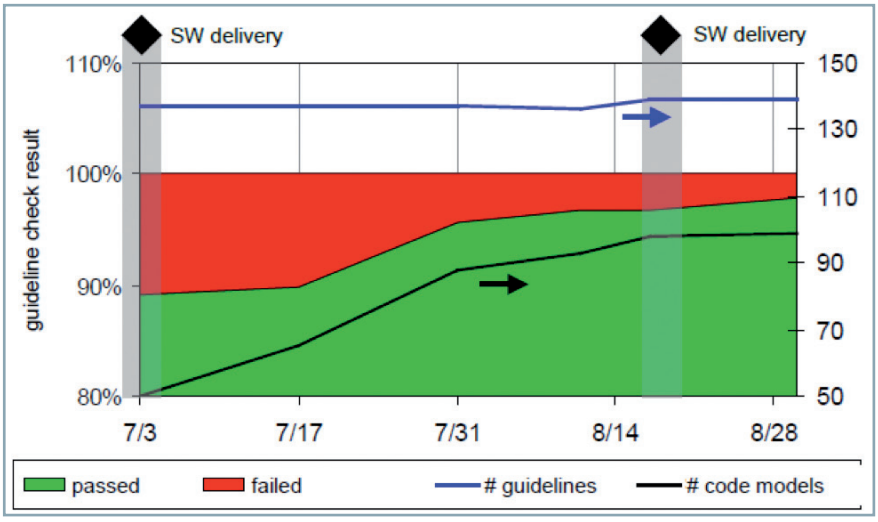

Bild 6: Tägliches Nachverfolgen der Ergebnisse der Richtlinienüberprüfung für ein Serien-software-Entwicklungsprojekt

Projekte mit vielen Modelldaten ausschlaggebend. Eine Optimierung des Ressourcenbedarfs für die Ausführung der Funktionen zur kontinuierlichen Integration ist ein weiteres Feld für zukünftige Optimierungen.

Schlussfolgerung

Die verschiedenen Antriebskonzepte machen – zusammen mit einer steigenden Zahl von Produktvarianten für verschiedene Märkte - agile Entwicklungsmethoden unentbehrlich für die Entwicklung von Automobilsoftware. Kontinuierliche Integration spielt bei der Durchführung agiler Entwicklungsmethoden eine Schlüsselrolle. Jedoch erfüllen bereits existierende Rahmenwerke nicht die MBD-Anforderungen, weil diese für die Softwareentwicklung mit textbasierten Sprachen entwickelt wurden. Daher wurde ein Rahmenwerk entwickelt, das diese agilen Methoden in der Entwicklung von Automobilsteuerungssoftware nutzen kann. Basierend auf Matlab wurde ein durchgängiger Ansatz implementiert, um alle Anforderungen bezüglich der Softwareentwicklung für Automobilsteuergeräte abzudecken. Als Beispiel für das Potenzial der entstandenen CI-Lösung wurde die automatisierte Überprüfung der Modellierungsrichtlinien vorgestellt. Der Automatisierungsgrad wurde erheblich erhöht, systematische Fehler können identifiziert werden, die Zeit, die für die Integration beansprucht wird, wird verringert und nicht standard- bzw. richtlinienkonforme und stattdessen entwicklerspezifische Individual-Lösungen werden vermieden. Um das volle Potenzial der kontinuierlichen Integration ausschöpfen zu können, muss weiter nach aussagekräftigen Softwaremetriken und frühen Prüfungen der Hardwareleistung bereits auf Modellebene geforscht werden. *(sc)* ■

Autoren

Dr. Philipp Orth, Dr. Axel Schloßer, FEV und Johannes Richenhagen, RWTH Aachen

Quellen

[1]J. Mössinger: Software in Automotive Systems, in: IEEE SOftware, Vol. 27, 2, 2010, pp. 92-94

[2]European Parlament and Council of the European Union: EU Directive 443/2009 (EG 2009c), Brussels, 2009

[3]United States Environmental Protection Agency, Department of Transportation: Light-Duty Vehicle Greenhouse Gas Emission Standards and Corporate Average Fuel Economy Standards - Final Rule, Washington D.C., 2010

[4]H. Helms et al.: Electric vehicle and plug-in hybrid energy efficiency and life cycle emissions, proceedings 18th International Symposium Transport and Air Pollution, Zurich, 2010, pp. 113-124

[5]A. Balazs et al. : Optimized Layout of Gasoline Engines for Hybrid Powertrains under Real World Driving
Conditions, proceedings 20th Aachen Colloquium Automobile and Engine Technology, Aachen, 2011

[6]A. Janssen et al.: Tailor-Made Fuels from Biomass for Homogeneous Low Temperature Diesel Combustion, Energy Fuels, Vol. 25, 10, 2011, pp. 4734–4744

[7]European Parlament and Council of the European Union: EU Directive 28/2009, Brussels, 2009

[8]United States Environmental Protection Agency, Department of Transportation: Renewable Fuel Standard
Program (RFS2) Regulatory Impact Analysis, Washington D.C., 2010

[9]United States Energy Information Administration: Annual Energy Review 2009, Washington D.C., 2009

[10] European Green Cars Initiative: European Roadmap Electrification of Road Transport Version 2.0, www.green- cars-initiative.eu, 2010

[11] N.N.: Nationaler Entwicklungsplan Elektromobilität der Bundesregierung, German Federal Government, Berlin, 2009

[12] J. Koepernik et al.: Funktionale Sicherheit in der Entwicklung von Fahrwerks- und Fahrerassistenz-Systemen: Fokus Systemarchitektur, Vector Congress, Stuttgart, 2010

[13] J. Dannenberg et al.: The Coming Age of Collaboration in the Automotive Industry, Oliver Wyman Group, http://www.oliverwyman.com/pdf_files/MMJ17-AutoIndustry-Collab.pdf, last view on 21.02.2011

[14] P. Braun et al.: Guiding requirements engineering for software-intensive embedded systems in the automotive industry, in: Computer Science – Research and Development, DOI 10.1007, 2010

[15] K. Grimm: Software technology in an automotive company - major challenges, in: Proceedings of the 25th int conference on software engineering, IEEE Computer Society, Washington, 2010, pp 498–503

[16] N.N.: Automotive SPICE® Process Reference Model. SPICE User Group / German Association of the Automotive
Industry (VDA), Berlin, 2010

[17] N.N.: CMMI Product Team: CMMI for Development, Version 1.2, Software Engineering Institute, Carnegie Mellon
University, Pittsburgh, 2006

[18] N.N.: ISO/DIS 26262 (2009) Road vehicles – Functional safety, Draft International Standard, International
Organization for Standardization (ISO), Geneva, 2009

[19] O. Kindel et al.: Softwareentwicklung mit Autosar – Grundlagen, Engineering, Management in der Praxis, Heidelberg, 2009

[20] Beine, M.: Model-Based Software Development for Safety-related Systems, dSpace, Paderborn, 2010

[21] H. Hungar et al.: SW-Entwicklung und Zertifizierung im Umfeld sicherheitskritischer und hochverfügbarer Systeme: Bedeutung modellbasierter und formaler Ansätze für effiziente Entwicklung und Zertifizierung, Software Engineering (Workshops) (2008), pp. 299-302

[22] D. Lederer: Systematische Software-Qualität mittels einer durchgängigen Analyse- und Teststrategie. conference „Software im Automobil", Stuttgart, 2010

[23] H. Giese et al.: MBEERTS: Model-Based Engineering of Embedded Real-Time Systems. International Dagstuhl
Workshop. Dagstuhl, 2010

[24] F. Deißenböck et al.: Kontinuierliche Qualitätsüberwachung mit CONQAT. Proceedings of GI Jahrestagung - 2 (2006), pp. 118-125

[25] F. Deißenböck et al.: Tool Support for Continuous Quality Control. IEEE Software, vol. 25 (2008), pp. 60-67

[26] N.N.: Annual Report 2010, German Federal Office for Motor Traffic, Flensburg, 2010

[27] N.N.: Emergency Road Service statistics, German automotive society (ADAC), Munich, 2009

[28] M. Fowler et al.: The Agile Manifesto, Software Development, vol. 8 (2001)

[29] M. Fowler: Continuous Integration, www.martinfowler.com, last view on 21.02.2011

[30] P. Duvall et al.: Continuous Integration: Improving Software Quality and Reducing Risk, Boston , 2007

[31] R. Koschke: Zehn Jahre WSR - Zwölf Jahre Bauhaus. Proceedings 10th Workshop Software Reengineering, vol. P-126 (2008), pp. 51–65

[32] N.N.: QALab, http://qalab.sourceforge.net/, last access on 08/24/2011

[33] International Standardization Organaization (ISO): ISO/IEC 25010: System and software engineering - Systems and Quality Requirements and Evaluation (SQuaRE) - System and software quality models, Geneva, 2011

[34] J. Richenhagen et al.: Continuous Integration for automotive model-based control software development, in: Proceedings AUTOREG, Baden-Baden, 2011, pp. 235-246