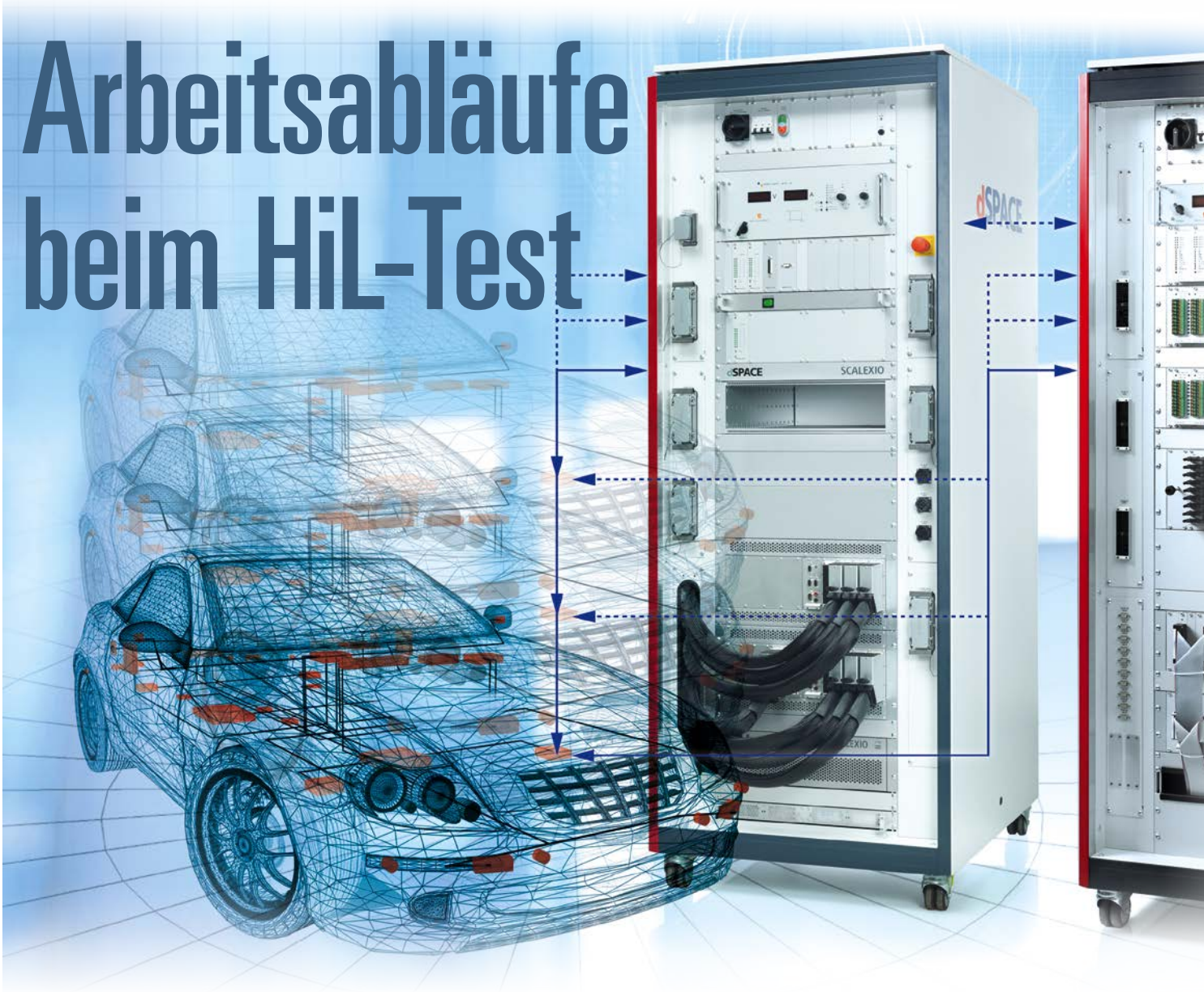
Variantenbasierte Arbeitsabläufe beim HiL-Test



Durch die Vielfältigkeit der Varianten und Versionen von Daten sowie der eingesetzten Software-Werkzeuge kann es von großem Nutzen sein, ein übergeordnetes Tooling zu verwenden, das Varianten und Versionen auf jedes Software-Werkzeug aufprägen kann. Mit dem Datenbanksystem SYNECT und dem Variant-based Workflow Management (VBWM) bietet dSpace die Möglichkeit, kundenspezifische Arbeitsabläufe mit einer oder mehreren Varianten-Konfigurationen anzusteuern.

Die Variantenvielfalt in der Automobilindustrie und damit die Anzahl der zu testenden Fahrzeug- und Steuergeräte-Kombinationen nimmt seit Jahren stetig zu. Eine Unterstützung für den Umgang mit Varianten für den in der Steuergeräteentwicklung etablierten HIL-Test ist damit unerlässlich geworden. Varianten sind im HIL-Testumfeld nahezu in jeder einzelnen Komponente vorhanden. Dabei lassen sich zunächst die Hardware- und die Software-Varianten unterscheiden (Bild 1). Die im HIL-Umfeld komplexeste Form der Testumgebung ist die HIL-Farm als Sammlung

Alle Bilder: dSpace

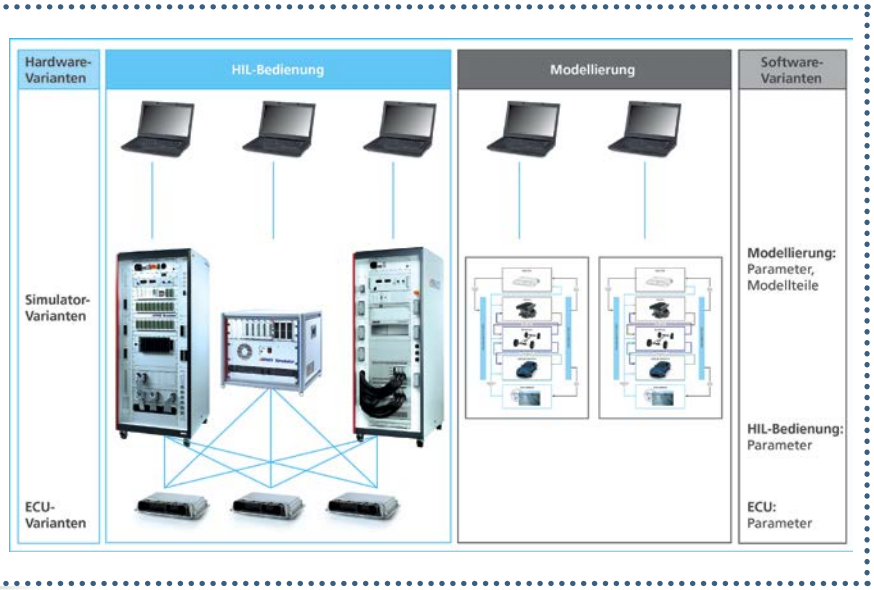


Bild 1: Variantenvielfalt der Hard- und Software.

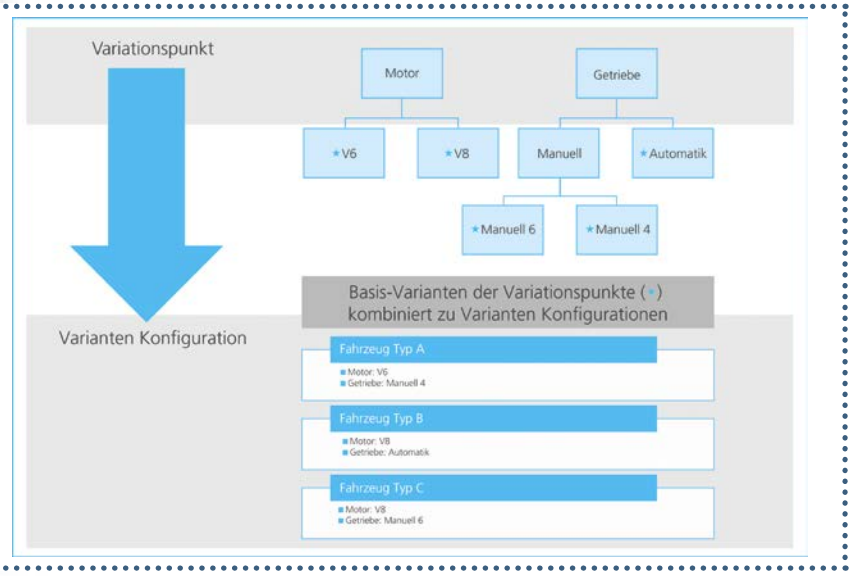


Bild 2: Bildung von Varianten-Konfigurationen durch Varianten aus Variationspunkten.

verschiedener HIL-Simulatoren. Solche HIL-Farmen wachsen oft im Laufe der Zeit. So entstehen durch Nachbauten weitere baugleiche Simulatoren, aber es entstehen durch Umbauten, Weiterentwicklungen oder Spezialisierungen auch unterschiedlichste Arten von Simulatoren.

Häufig werden Komponenten-Simulatoren auch zeitweilig zu Verbundsimulatoren zusammengeschaltet und betrieben. Weiterhin können Simulatoren in sich auch wieder Varianten beinhalten, indem sie hardwaretechnisch umschaltbar sind. Dies kann mit Hilfe weniger Relais bis hin zum gesamten Umschalten kompletter Steuergeräte-Stecker realisiert werden. Oft ermöglicht die Betriebssoftware die Umschaltung einer Hardware-Variante im laufenden Testbetrieb.

Hardware- und Software-Varianten

Eine oder auch mehrere ECUs liegen für sich ebenfalls wieder in diversen Varianten vor und müssen entsprechend an die Simulatoren angeschlossen werden. Dies geschieht entweder manuell vor einem automatischen Test oder über die oben genannten Hardware-Schalter durch die Software auch während des Testens.

Die nächste Stufe der Varianten ist die entsprechende Software auf dem Simulator und auf der ECU. Hier muss man zwischen den Varianten unterscheiden, die ein neues Laden des Programms auf den Simulator oder die ECU erfordern, und den Varianten, die durch Parameter im laufenden Programm umgeschaltet werden können. Für Varianten, die ein erneutes Laden erzwingen, gibt es meistens zwei Gründe: Zum einen, dass der Build/Compiler es nicht zulässt, alle erforderlichen Varianten in einem Code zu bearbeiten. Zum anderen, dass andere Limitierungen wie Laufzeit oder Speicher dieses verhindern.

Abhängig von diesen Varianten müssen entsprechend sowohl die Bedien-Software am Simulator-Bedien-PC als auch die Testautomatisierung variantenabhängig arbeiten. So müssen sich die Bedien-Oberflächen der Simulator-Software anpassen, und/oder das Diagnosewerkzeug und seine Bedienung müssen sich den Varianten der ECU anpassen. Die Testautomatisierung muss sowohl die Diagnose-/Kalibriereingriffe als auch die Eingriffe in das Programm des Simulators variantenabhängig durchführen. Die einzelnen Varianten lassen sich in einer Gruppierung durch Variationspunkte beschreiben (Bild 2).

Die Zusammenstellung der Varianten aus allen Variationspunkten, also die Varianten-Konfiguration, bildet dann die Information über das Testsystem und seine Zusammensetzung.

Unterschiedliche Datenquellen

Der HIL-Entwicklungsprozess beinhaltet die in jedem HIL-Entwicklungszyklus wiederkehrenden Schritte von der Entwicklung der Simulator-Software, die meist in einer grafischen Entwicklungsumgebung erstellt wird, bis zum Test am Simulator. In diesem Kreislauf, der oft synchronisiert mit den Entwicklungszyklen der ECUs abläuft, werden zum einen viele Schritte immer wieder durchlaufen und zum anderen sind heutzutage diverse Datenquellen an diesem Zyklus beteiligt. Diese Datenquellen liegen in unterschiedlichster »

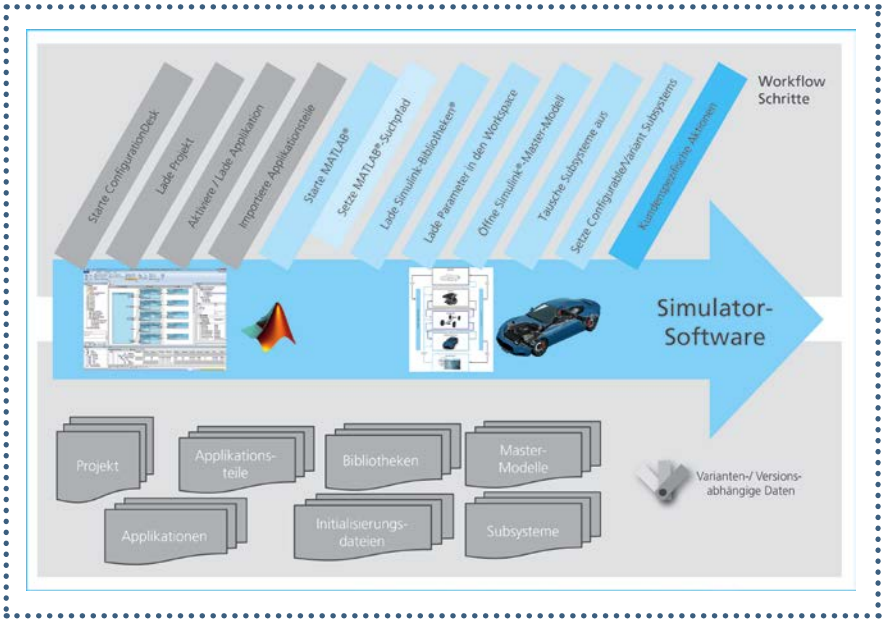


Bild 3: Workflow am Bedien-PC.

Form vor, wobei sie, motiviert durch die genutzten Software-Werkzeuge, auch gemischt genutzt werden. Zu den gängigsten Formen gehören die Folgenden:

- Daten in einer Datenbank
- Dateien auf dem Dateisystem oder einem Netzlaufwerk
- Dateien in einem Versionierungswerkzeug, synchronisiert in eine lokale Sandbox

Eine der Herausforderungen in diesem Prozess ist es also, die zur Erstellung der Simulator-Software notwendigen Daten zusammenzustellen und dann unter einem Variantenbezug konsistent zu einem Programm zu kompilieren.

Durch die immer weiter entwickelten Standards für Sicherheit wird auch die Zuverlässigkeit solcher Software-Entwicklungsprozesse immer wichtiger. Dies bedeutet vorrangig die eindeutige Versionierung und ein entsprechendes Zusammenstellen (Baselining) der einzelnen Datenversionen zu einer Versionskonfiguration.

Durch die komplexen Arbeitsabläufe und die damit möglichen Fehler im Zusammenstellungsprozess bis hin zum Kompilat der Simulator-Software wird aber auch die Automatisierung solcher Prozesse immer wichtiger, um eine Reproduzierbarkeit zu gewährleisten und um diese Komplexität mit begrenzten Ressourcen beherrschbar zu halten.

Zusätzlich zu diesen vielfältigen möglichen Datenquellen wird ein HiL-Entwicklungsprozess geprägt von den benötigten Software-Werkzeugen, die am Ende zum kompilierten Programm für den Simulator führen oder diesen später bedienen.

So wie die Einzelkomponenten, die zu der Gesamtsoftware des Simulators führen, versioniert und variantenabhängig sind, so muss auch ein Arbeitsablauf unter diesen Randbedingungen gestartet werden. Für jeden einzelnen der zu automatisierenden Arbeitsabläufe sind somit die folgenden Randbedingungen vorzugeben:

- Varianten-Konfiguration
- Versionskonfiguration
- Informationen über die aktuelle Umgebung (Nutzer, PC), um eventuelle Abhängigkeiten zu diesen Daten ebenfalls mit zu berücksichtigen. So kann der PC ausschlaggebend dafür sein, wo zum Beispiel die Daten abgelegt sind oder die benötigte Software installiert ist.

Neben den einzelnen Daten, die jedes Software-Werkzeug in unterschiedlicher Ausprägung benötigt, um seinen Anteil am Gesamtprozess zu leisten, ist somit auch eine zusammenfassende Datenhaltung notwendig. Diese stellt die Verbindung (Link) zwischen den Varianten, den Daten und ihren Versionen her, die ein Arbeitsablauf benötigt, um sämtliche notwendigen Daten zu erhalten. Nachfolgend werden zwei typische Arbeitsabläufe vorgestellt.

Beispiel 1: Entwicklung der Simulator-Software

Konkret im Beispiel der Simulationsumgebung der dSpace-Werkzeugkette mit dem Konfigurationswerkzeug *ConfigurationDesk* und der Modellierungsumgebung *MATLAB/Simulink* sind vielfältige Daten notwendig. ConfigurationDesk muss bezüglich der Simulator-Hardware bedatet werden. In MATLAB/Simulink werden Pfade auf Libraries gesetzt und es können unterschiedliche Basis-Modelle in Simulink notwendig sein. Dieses Modell, das meist auch simulatorspezifische Elemente beinhaltet, legt damit den Grundstein für die variantenabhängige Software des Simulators. In diesem Basis-Modell sind vor allem die hardwarespezifischen Anteile definiert, ergänzend zu den Konfigurationen in ConfigurationDesk. So würde erst ein hardwaretechnischer Umbau des Simulators eine Änderung dieser Schichten erfordern. Im Optimalfall werden alle anderen Anteile in diese Basis verlinkt oder entsprechend integriert, so dass sie in einer anderen Varianten-Konfiguration ebenfalls zur Anwendung kommen. So lassen sich Redundanzen vermeiden und eine Erweiterung der HIL-Farm um einen neuen Simulator wird erleichtert.

Beispiel 2: Einsatz des Simulators für den Anwender am Bedien-PC

Auch das Starten des Simulators am Bedien-PC erfordert die Zusammenstellung der entsprechenden Daten. Basis ist hier die im ersten Prozess entstandene Software für den Simulator, die nun auf den Simulator geladen werden muss. Dazu kommt die Software für die ECUs, die Konfiguration des Applikationswerkzeugs und die Konfiguration für die Simulator-Bedien-Software. Auch diese Daten werden aus einer der Datenquellen unter Varianten- und Versionsbezug entnommen.

Besonders im Fall der Bedienung des Simulators ist davon auszugehen, dass hier die Spezialisten der ECUs mög-

lichst einfach zu einer Konfiguration gelangen wollen, die es ihnen ermöglicht, den Simulator zu nutzen. Dies kann durch eine Automatisierung sichergestellt werden, die, wie oben beschrieben, wieder die Varianten-Konfiguration und die Versionskonfiguration als Eingangswerte benötigt. Die Automatisierung erfordert zudem weniger Simulator-Kenntnisse von den ECU-Spezialisten (Bild 3).

Variantenbasiertes Workflow-Management

Durch die Vielfältigkeit der Varianten und Versionen von Daten sowie der eingesetzten Software-Werkzeuge und deren unterschiedliche Möglichkeiten, sich mit Varianten und/oder Versionen direkt zu konfigurieren, kann es von großem Nutzen sein, ein übergeordnetes Tooling zu verwenden, das Varianten und Versionen auf jedes Software-Werkzeug aufprägen kann. Voraussetzung für ein solches Software-Werkzeug ist lediglich eine Automatisierungsschnittstelle, die die von Varianten und Versionen abhängigen Teilkonfigurationen austauschbar macht.

Mit dem Datenbanksystem *SYNECT* und dem darauf basierenden Variant-based Workflow Management (VBWM) bietet dSpace speziell für die dSpace-Simulator-Werkzeugkette die Möglichkeit, kundenspezifische Arbeitsabläufe mit einer oder mehreren Varianten-Konfigurationen anzusteuern und diese mit Daten aus der SYNECT-Datenbank zu unterstützen. Diese Daten aus der SYNECT-Datenbank sind dabei, wie oben beschrieben, vornehmlich Links zwischen den Varianten und den Daten aus den unterschiedlichen Datenquellen. Neben diesem generischen Ansatz bietet das VBWM vorgefertigte Anteile, um zum Beispiel die beiden oben beschriebenen gängigen Arbeitsabläufe in der dSpace-Werkzeugkette optimal zu unterstützen. ■ (oe)

» **www.dspace.com**

Dipl.-Ing (FH) Ulf Homann ist Gruppenleiter Testautomation und Tools bei der dSpace GmbH in Paderborn.

Dipl.-Ing. Markus Plöger ist Abteilungsleiter Engineering HIL bei der dSpace GmbH in Paderborn.