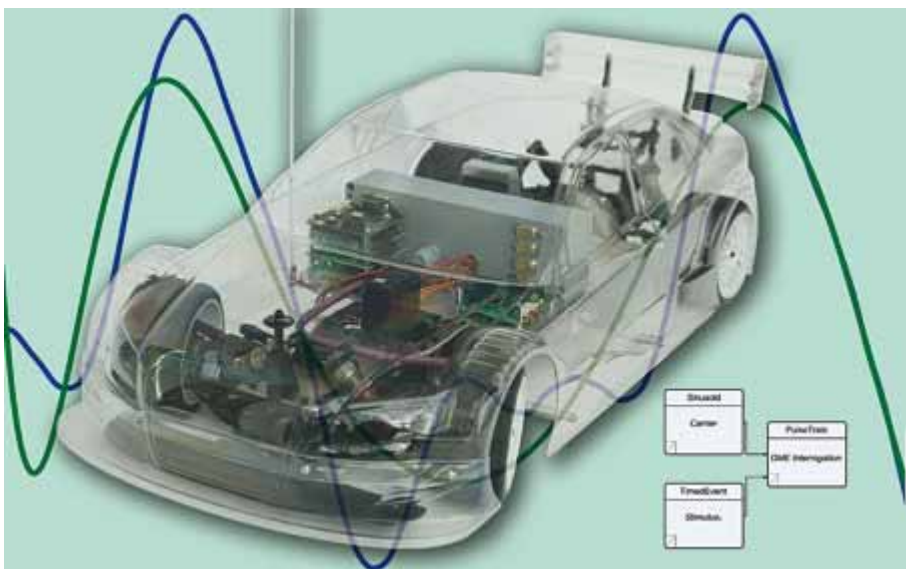


Hohe Produktivität und weniger Fehler:

Modellbasierte Entwicklung soll Aufwand und Zeit reduzieren. In einer Industrie, in der Zeitdruck täglicher Begleiter von Entwicklungsingenieuren ist und in der die Kosten nicht niedrig genug sein können, sind das allemal Argumente, sich damit zu beschäftigen.



Modellbasierte Entwicklungs- und Testverfahren werden besonders in der Fahrzeugentwicklung eingesetzt.

Das Thema ist heiß: Studenten der Mechatronik, der Fahrzeugelektronik oder der Avionik diskutieren auf Partys darüber. Keine Besprechung zwischen Entwicklungsmanagern für Embedded Systems, auf der nicht das Stichwort „Modellbasierte Entwicklung“ fällt. Bereits in Programmierhochsprachen wie C und Fortran greifen Programmierer auf immer wiederkehrende Funktionen zurück, um komplexe Aufgaben zu lösen (Funktions-Libraries). Die modellbasierte Entwicklung greift auf diesen Erfahrungsschatz zurück, verbindet ihn mit grafischen Symbolen und stellt ihn dem Programmierer als Bibliothek zur Verfügung. Elektronikentwickler beispielsweise können Schaltsymbole direkt am Bildschirm zu einem System zusammenstöpseln. Klar, dass damit die Arbeit flüssiger von der Hand geht.

Wiederverwendung sorgt für Reife

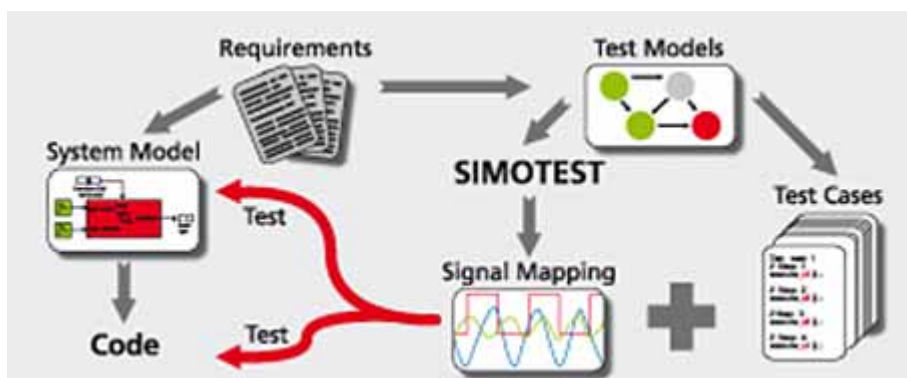
Mittels modellbasierter Techniken erzeugte Software weist eine höhere Qualität auf als konventionell programmierte, propagieren die Verfechter des Ansatzes. Unabhängige Studien stützen diese These und nennen zwei Gründe: Die einzelnen Funktionsblöcke sind durch zigfache Wiederverwendung ausgereift; die damit erzeugte Softwarelösung enthält somit tendenziell weniger Fehler. Außerdem stehen

für viele Sprachen sogenannte Validatoren zur Verfügung, welche die formale Korrektheit des modellierten Systems automatisch durchchecken. Dagegen

schreiben Entwickler bei textbasierten Sprachen die Testroutinen selbst, in den seltensten Fällen ohne Fehler.

Auffälligstes Merkmal von modellbasierten Strategien ist, dass sie sich zunehmend der Anwendungswelt des jeweiligen Entwicklers annähern – in ihren Begrifflichkeiten, Konstrukten, Denkweisen und Lösungsansätzen. Das ist Vorteil und Nachteil zugleich. Vorteile sind die intuitive Arbeitsweise und die praxisnahen Lösungselemente, die eine modellbasierte Umgebung anbietet. Der Nachteil ist die schwindende Universalität hochentwickelter Sprachen.

Ein Beispiel dafür ist die Specification and Description Language (SDL), die in den neunziger Jahren Entwicklern in der Telekommunikationsbranche das Leben erleichtern sollte. Mit ihrer Definition spezifischer Standard-Zustände und -Übergänge ist sie einerseits nahe dran an der Problemstellung und damit an der Denkweise der Experten. Damit erwies sie sich als produktivitätsfördernd für Aufgabenstellungen der Telekommunikation. Andererseits ist sie in anderen Branchen nicht zu verwenden. „Genau das ist das Problem modellbasierter Verfahren“, sagt Daniel Görlich, Business Area Manager am Fraunhofer-Institut für Experimentelles Software-Engineering (IESE) in Kaiserslautern: „Man benötigt für jede Domäne eine eigene Sprache.“ Übergreifende Ansätze wie UML (Unified Modeling Language) weisen eine überbordende Komplexität auf. „Das U wird hier häufig als ‚universal‘ fehlinterpretiert“, so Görlich. „Aber das ist falsch, auch UML löst nicht alle Probleme.“



Organisation eines modellbasierten Entwicklungs- und Testprozesses mit dem Tool Simotest.

Fachspezifisch modellieren ist angesagt

Seit einigen Jahren geht der Trend daher wieder zurück zu domänenspezifischen Sprachen, beobachtet Görlich. Entstanden sind leicht – weil intuitiv – zu bedienende Systeme, etwa Matlab/Simulink, das in der Industrie als die Standardplattform für die Lösung mathematischer Probleme gilt. Ein anderes Beispiel ist die quelloffene und erweiterbare Entwicklungsumgebung Eclipse. Auch in der kommerziellen Informationsverarbeitung und in Consumer-Märkten spielen modellbasierte Verfahren

eine Rolle: Genauso wie sich eine elektronische Regelschleife oder ein ABS-System modellieren lassen, ist es möglich, Geschäftsprozesse oder Nutzerverhalten als Modell abzubilden und dieses Modell zur automatischen Generierung von Software

zu nutzen. „Ein solches System erlaubt es, vieles automatisch zu generieren, weil die Aufgabe formal bereits durchmodelliert ist“, erläutert Michael Ochs, Geschäftsführer Informationssysteme am Fraunhofer IESE. Allerdings unterscheiden sich die Anforderungen an Embedded Systems und Enterprise-Solutions in wesentlichen Punkten: Während bei eingebetteten Systemen regelmäßig das Echtzeitverhalten eine wesentliche Komponente des Gesamtentwurfs darstellt, ist dies bei Enterprise-Solutions ein nachrangiger Gesichtspunkt. Wenn sich die Seite einer Handelsplattform im Internet mal schneller, mal langsamer aufbaut, so ist dadurch die Funktion im Kern nicht tangiert, bei einer Software zur Regelung des Fahrverhaltens eines Autos dagegen schon.