Softwareentwicklung für die Mechatronik

Schon seit längerer Zeit ist der Maschinenbau nicht mehr reinrassig, sondern vermischt mit Elektrik und Software, deren Anteil im Steigen begriffen ist. Die ständig wachsende Komplexität wird in Zukunft nur zu beherrschen sein, wenn die unterschiedlichen Komponenten einander durchdringen, die Entwicklungsdisziplinen Hand in Hand arbeiten. Dieser mechatronische Ansatz braucht Entwicklungsumgebungen, die diese Interdisziplinarität unterstützen, indem sie die Komplexität auflösen und mit offenen Schnittstellen Brücken entstehen lassen.

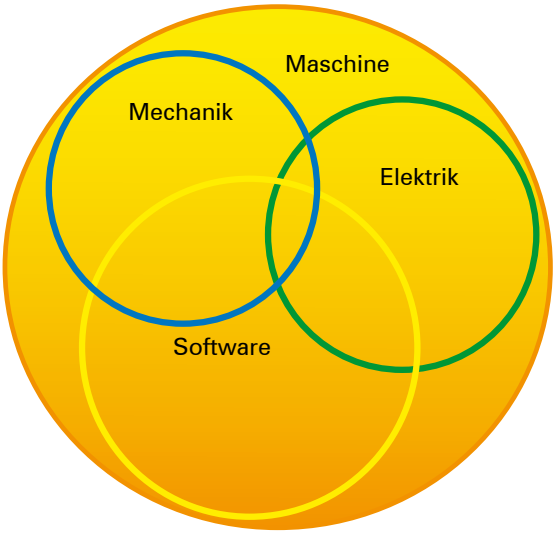
Der Maschinenbau hat den Ruf, eine konservative Sparte zu sein und größeren Neuerungen mit Zurückhaltung zu begegnen. Dennoch sind in den letzten Jahren durch anspruchsvolle Marktanforderungen und das Bestreben, gleichzeitig die Produktkomplexität zu steigern und Marktreifezeiten zu verkürzen, gewaltige Veränderungen in Gang gekommen, die zu einer Abkehr von traditionellen Entwicklungsmethoden geführt haben. Dazu gehört die Mechatronik, ein Begriff, der zu den am meisten strapazierten Buzzwords der Branche geworden ist.

Das ruft zunächst nach einer Definition des Begriffs, denn es gibt kein einheitliches Verständnis von Mechatronik. Ein Gebilde, das aus mechanischen Konstruktionen, elektrischen Komponenten und einem Stück Software besteht, erfüllt für mich noch nicht den Tatbestand der Mechatronik, denn sie folgt der sequenziellen Denkweise im Produktentstehungsprozess, nach der zunächst eine Konstruktion zu schaffen, diese anschließend mit Steuerungs- und Regelungstechnik auszustatten und zuletzt mittels Software zum Leben zu erwecken ist.

Mechatronik als Denkweise

Mechatronik geht darüber hinaus, denn sie ist in erster Linie eine Denkweise. Eine Denkweise, die wesentlich näher am Kundenbedürfnis und am Kundennutzen ist, denn in der Mechatronik sind die relevanten und meist sequenziell betrachteten Einheiten die Funktionen des Gesamtgebildes. Mechanik, Elektrik und Software werden miteinander gedacht und in wechselseitiger Abstimmung parallel konzeptioniert, geplant und umgesetzt. Im Endeffekt werden die steigende Komplexität der Anforderungen an Serienmaschinen mit Losgrößen bis herunter auf 1 und die resultierende Komplexität der Entwicklungsaufgabe auf andere Weise nicht mehr zu bewältigen sein, jedenfalls nicht in der geforderten Zeit und zu den geforderten Kosten.

Ein immer wieder angezogenes Musterbeispiel für eine mechatronische Lösung ist die Fahrwerksregulierung, die im Automobil dafür sorgt, dass durch bedarfsgesteuerte Nachführung die Grenzen der Mechanik ausgedehnt werden. Ähnliche Beispiele gibt es durchaus bereits auch im Maschinenbau, etwa dort wo in der Verpackungsindustrie Verfahrgeschwindigkeiten so hoch sind, dass das natürliche Schwingungsverhalten der mechanischen Komponenten die Maschinendynamik limitieren kann. Durch eine intelligente Antriebsregelung können die negativen Effekte jedoch nicht nur kompensiert, sondern auch bisherige Grenzen überschritten werden.



Maschine = Mechatronik

Nicht jede Zusammenstellung von Mechanik, Elektrik und Software macht eine Maschine zur Mechatronik. Erst das ineinander greifende Zusammenspiel aller Disziplinen, auch und vor allem bereits im Entwicklungsprozess, lässt Mechatronik entstehen.

Optimierung durch Disziplinübergriff

Der Großteil der Maschinen reizt jedoch die Gesetze der Physik und die Möglichkeiten der Mechanik nicht in diesem Maß aus. Dort steckt der Vorteil der Mechatronik in erster Linie im Entwicklungsprozess. Durch die mit diesem Ansatz mögliche Parallelisierung der Gesamtaufgabe kann der Termindruck erheblich reduziert werden. Der ist traditionell im Software-Engineering besonders hoch, wenn mit ihrem Start auf das Vorliegen von Mechanik und Elektrik gewartet wird. Zudem sind im sequenziellen Entwicklungsansatz auch gegenseitige Abstimmung und Einflussnahme keine Option, da die Vorleistungen aus den anderen Disziplinen als gegeben angesehen werden müssen. Im Gegensatz dazu sind an einer mechatronischen Entwicklung beteiligte Techniker in der Lage, durch disziplinübergreifende Optimierungsiterationen zu Ergebnissen zu gelangen, die durch das aufbauende Zusammenfügen der einzelnen Entwicklungsschritte nicht zu erreichen wären.

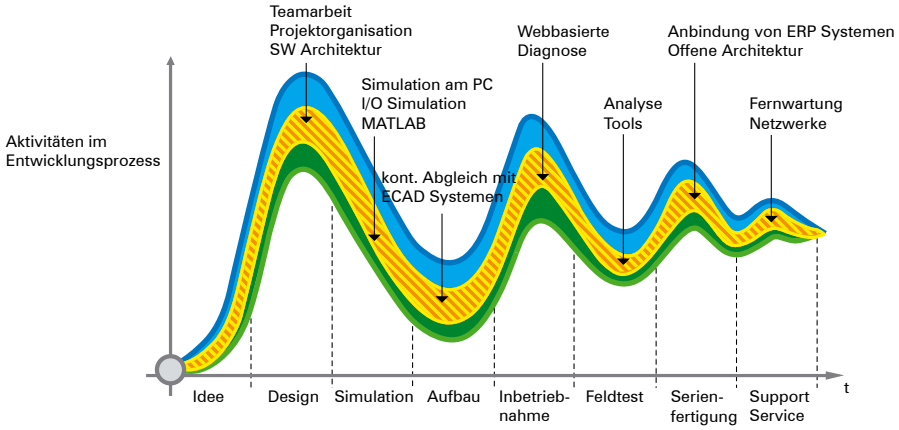
Eine der wesentlichsten Veränderungen für die Automatisierungstechnik durch die mechatronische Herangehensweise an Problemstellungen der Maschinenentwicklung ist Zeitpunkt und Dauer des Einsatzes von Softwareentwicklungssystemen. Diese beginnt bereits in der Phase des Überganges von der Produktidee zur Spezifikation, denn bereits zu diesem Zeitpunkt sind durch die Softwareentwicklung Machbarkeiten zu klären und vorteilhafte Abläufe festzulegen. Mit wechselnder Intensität kommt die Softwareentwicklung während aller Phasen des Produktlebenszyklus zum Einsatz, von Design und Simulation über Inbetriebnahme, Feldtest und Serienfertigung bis zu Instandhaltung und Wartung.

Das schafft natürlich neue Verantwortlichkeiten im Entwicklungsprozess. Da die Softwareentwicklung immer mehr zum Treiber der Gesamtentwicklung wird, muss sie mehr liefern als nur Code, nämlich einen interdisziplinären Systementwurf. Dadurch steigt die Komplexität der Softwareentwicklung und ihr Anteil an der Gesamtverantwortung.

Schlüsselfaktor Kommunikation

Es ist vorteilhaft, nicht nur Strategien zur Beherrschung dieser gestiegenen Komplexität zu finden, sondern durch organisatorische Maßnahmen auch die >>

Lebenszyklus einer Maschine



Automation Studio im Entwicklungsprozess

In mechatronischen Systemen findet die Softwareentwicklung in unterschiedlicher Intensität während des gesamten Produktlebenszyklus statt. Das führt dazu, dass die Aufgaben und Verantwortlichkeiten in der Software mehr werden.

verschiedenen an der Maschinenentwicklung beteiligten Teams innerhalb der Unternehmen enger zusammen zu bringen. Wichtig ist eine Erweiterung des Horizontes aller Beteiligten, denn es wird nicht länger genügen, das Verständnis über die Gesamtaufgabe nur in den Köpfen etwaiger Projektleiter zu haben, dieses Gesamtverständnis muss vielmehr in den Köpfen aller beteiligter Spezialisten sein, die dazu erforderlichen Informationen müssen ihnen zur Verfügung stehen.

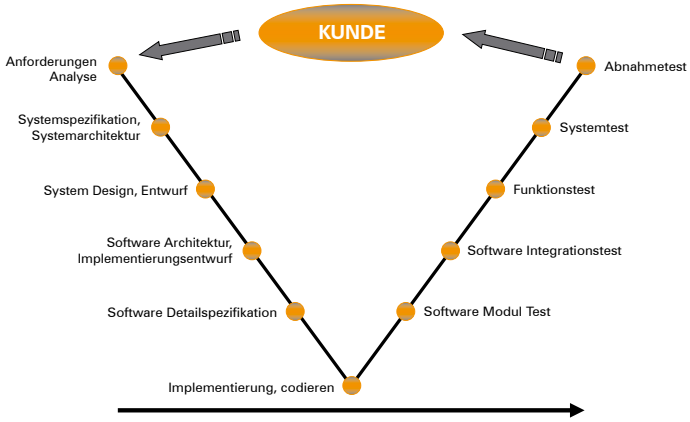
Vor allem aber ist eine Sprachbarriere zu überwinden, denn die einzelnen Disziplinen verwenden unterschiedliche Methoden der Problem- und Systembeschreibung und ihre jeweiligen Werkzeuge ebenfalls. Der mechatronische Problemlösungsansatz macht es erforderlich, den mitarbeitenden Mechanismen zur Verfügung zu stellen, die einerseits Freiraum für spezifische Beschreibungsformen lassen, andererseits die Komplexität reduzieren und den Kommunikationsbedarf auf die wesentlichen Punkte zu reduzieren. Nur so entsteht ein inhärentes Verständnis und kann die zu entwickelnde Funktion die logische gemeinsame Konvention bilden, ohne dass überzogene Qualifikationsansprüche erfüllt werden müssen.

Vor allem unter diesem Aspekt ist daher der Ansatz, hohe Systemkomplexität durch objektorientierte Entwicklungsmethoden und -werkzeuge zu reduzieren kritisch zu betrachten. Auch wenn sie sich in der IT Welt durchaus bewährt hat, wird die objektorientierte Programmierung nicht lückenlos in alle Bereiche der Industrieautomatisierung vordringen können, denn in einer Lebenszyklusbetrachtung ist auch auf den Fall Rücksicht zu nehmen, dass Anpassungen im Feld durch Servicepersonal durchzuführen sein werden. Da kann man sich etwas Handlicheres vorstellen als objektorientierten Code, womöglich mit aufwändigen Klassenstrukturen und Vererbungshierarchien.

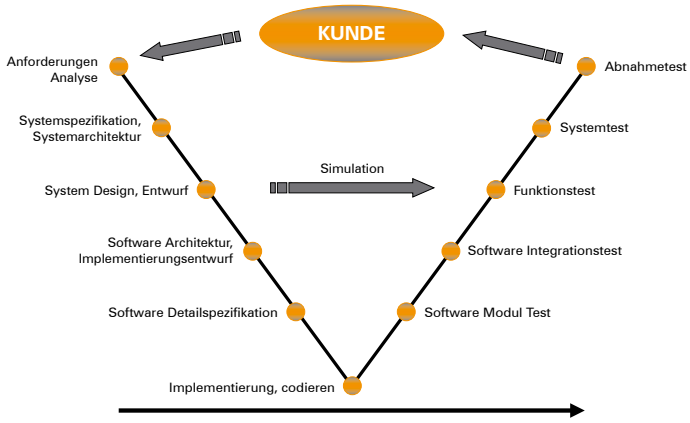
Veränderte Anforderungen an die Entwicklungsumgebung

Ganz klar ist, dass die Entwicklungssysteme der Zukunft große Offenheit aufweisen und mit rollenbasierten Möglichkeiten zur Systemmodellierung und Darstellung ausgestattet sein müssen. Zugleich macht die Verwendung über den gesamten Produktlebenszy-

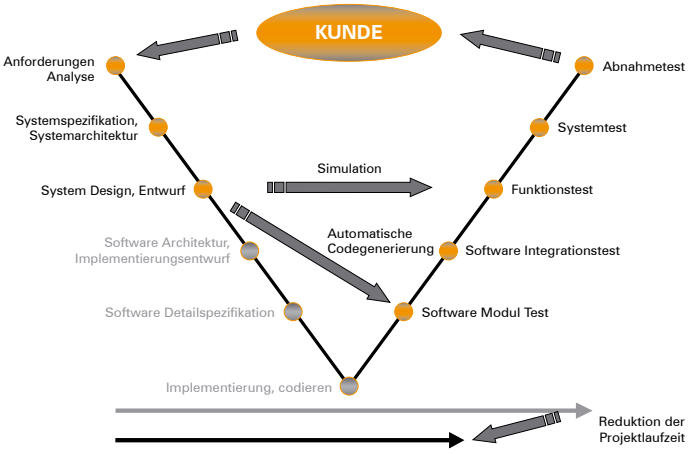
Modellgetriebene Entwicklung



Modellgetriebene Entwicklung



Modellgetriebene Entwicklung



Anhand des V-Modells der modellbasierten Entwicklung leicht erkennbar: Softwaremethoden wie die Simulation verkürzen bereits die Gestaltungszyklen, wesentlicher für die Verkürzung der Projetlaufzeit ist jedoch der Wegfall mehrerer Projektschritte durch die Automatische Codegenerierung aus der Simulation.

klus Werkzeuge und Automatismen für die Dokumentation, das Änderungs- und Variantenmanagement ebenso erforderlich wie noch umfangreichere Möglichkeiten für Simulation, Test und Diagnose.

Bereits heute ist Automation Studio, die durchgängige Entwicklungsumgebung von B&R, auf viele dieser Herausforderungen gut vorbereitet. Obwohl bisher die Unterstützung der prozedural orientierten internationalen Norm IEC 61131 im Vordergrund stand, ist die Objektorientierung durch die Integration von C++ als eine der zahlreichen möglichen Programmiersprache ebenfalls gegeben. Bereits von der Projektierung weg können unterschiedliche Systemperspektiven nach logischen, konfiguratorischen oder physischen Gesichtspunkten gewählt werden, mehrere Varianten und beliebige Dateitypen in einem gemeinsamen Projekt verwaltet oder unterschiedliche Versionen gehalten werden.

Die Richtungsvorgabe ist ein Topologie-Editor, mit dem ein grafischer Zugang zur Maschinenkonfiguration geschaffen wird. So entsteht ein Einstiegspunkt in eine objektbasierte Entwicklungswelt, der hoch genug ist, um nicht auf die Verwendung durch eine bestimmte Benutzergruppe innerhalb der Entwicklungsabteilungen beschränkt zu sein. Das wird es noch wesentlich einfacher und komfortabler als bisher machen, Top-down Design ausgehend von der Konzeptionierung bis zum ausformulierten Gesamtsystem zu betreiben und bedarfsorientiert die funktionsbasiert entwickelte Software auf unterschiedliche Hardwarekonfigurationen zu mappen.

Entwicklungsverkürzung durch Software

Wesentlich ist in allen Überlegungen die Verkürzung der Zykluszeiten einzelner Entwicklungsschritte. Dazu tragen neben den unterschiedlichen Sprachen und Programmiermethoden vor allem die Verwendbarkeit von Programmcode aus anderen Quellen bei. Prominentestes Beispiel ist die Integration der Simulation. Diese verkürzt an sich schon die Iterationsdauer, da sie Prototypenbau und Testprozeduren ersetzen kann. Darüber hinaus kann jedoch - und das seit geraumer Zeit - von Simulationswerkzeugen automatisch generierter Code in Automation Studio übernommen

und im Runtime ausgeführt werden, was eine weitere echte Entwicklungsverkürzung bedeutet und darüber hinaus hilft, Interpretationsfehler zu vermeiden. Solche Möglichkeiten zur Integration von Beschreibungsdaten aus anderen Systemen und Disziplinen, etwa durch eine SysML Schnittstelle, werden in Zukunft sehr stark zunehmen.

Trotz eines disziplinübergreifenden Ansatzes der Mechatronik Entwicklung wird es nie dazu kommen, dass ein einheitliches Entwicklungssystem für alle Teilproblematiken praktikabel wird. Umso wichtiger wird es sein, durch die Schaffung leistungsfähiger Schnittstellen die entscheidende Information zwischen den einzelnen Systemen zu transportieren. Um den Entwicklungsprozess nicht unnötig aufzublähen, wird dazu in der Regel nur die Änderung zu transportieren sein. Das macht das Änderungsmanagement umso bedeutender.

Um heutigen Marktbedürfnisse gerecht zu werden, ist Automation Studio das am weitesten gehende Entwicklungssystem für Automatisierungssoftware. Ziel und Vorhaben der Business Unit Automatin Software bei B&R ist eine Weiterentwicklung von Automation Studio mit derselben Geschwindigkeit und Richtung wie die Marktbedürfnisse. Dazu gehört die bestmögliche Abdeckung des mechatronischen Ansatzes.

Zum Autor:

Dr.-Ing. Hans Egermeier (35) ist seit Dezember 2009 Business Manager der Business Unit ASW (Automation Software) bei B&R. In dieser wichtigen Entwicklungsabteilung des Automatisierungsunternehmens arbeiten die Mitarbeiterinnen und Mitarbeiter an der Weiterentwicklung des Engineering Paketes Automation Studio, des Betriebssystems Automation Runtime und der integrierten Visualisierungssoftware Visual Components.

Bereits während seines Maschinenbaustudiums und der anschließenden leitenden Tätigkeit im Institut für Werkzeugmaschinen und Betriebswissenschaften der TU München beschäftigte sich Hans Egermeier mit der Entwicklung von Software zur Virtualisierung manueller Montageprozesse. Die Entwicklung interaktiver Software setzte sich in seiner späteren Berufslaufbahn fort, wo er Spiele zur Unterstützung von Therapien entwickelte und zuletzt für einen Kamera-basierten Abschlagsensor für das Golftraining mit dem Product Innovation Award ausgezeichnet wurde. Als sein Ziel bei B&R bezeichnet es Hans Egermeier, Automation Studio zur erfolgreichsten Entwicklungsumgebung zu machen.