Programmierung von Kfz−Steuergeräten

Zusammenfassung

Der steigende Anteil von Software im Kfz verlagert Entwicklungsaufwendungen und verändert die Entwicklungsprozesse. Den kürzer werdenden Entwicklungszyklen und den wachsenden Softwareanforderungen kann nur durch eine Optimierung des Entwicklungsprozesses begegnet werden. Es werden der heutige Stand der industriellen Praxis aufgezeigt und Möglichkeiten beschrieben, den Entwicklungsprozeß zu verbessern, und ein Ausblick auf zukünftige Trends gegeben.

Motivation

Für viele Autofahrer unbemerkt haben elektronische Steuerungen die Kontrolle im Kraftfahrzeug übernommen. Der Anteil der Elektronik an den Produktionskosten wird in naher Zukunft den Anteil von 25 Prozent erreicht haben. Noch stärker nimmt der Umfang der Software in den Steuergeräten zu und die damit einhergehende Programmentwicklung. Auch die Vernetzung der Geräte über mehrere Busse und die gegenseitigen Abhängigkeiten steigen weiter an.

In den Entwicklungsabteilungen der Automobilhersteller und der Zulieferer entsteht damit die Anforderung, immer mehr Software, die immer komplexere Anforderungen erfüllen muß, in immer kürzerer Zeit zu entwickeln. Es ist daher wichtig, sich über den Prozeß der Softwareentwicklung Gedanken zu machen und über die Möglichkeiten der Optimierung und Ressourceneinsparung.

Elektronik im Kraftfahrzeug

In hochwertigen Fahrzeugen befinden sich heute zwischen 50 und 100 Steuergeräte, die die Sensoren auswerten, über komplexe Regelalgorithmen Sollwerte berechnen und über Aktuatoren ausgeben. Beispiele hierfür sind Steuergeräte für den Motor, das Getriebe, Navigationssysteme oder Airbags. In wenigen Jahren werden die mechanischen Verbindungen am Gas−, Brems−, Kupplungspedal und dem Lenkrad entfallen und durch eine elektronische Datenübertragung ersetzt (X−by−Wire).

Derzeit wird begonnen, die Steuergeräte über Bussysteme zu vernetzen, die abgestuft über unterschiedliche Leistungsanforderungen neue Dimensionen eröffnen. Die Verkabelung, die relativ kostenintensiv ist, kann dadurch teilweise eingespart werden. Weiterhin werden neue Möglichkeiten für zusätzliche Merkmale geboten. Beispielsweise kann dann jedes Steuergerät bei Bedarf auch auf Sensorinformationen anderer Regelkreise zugreifen und z.B. über die Abfrage der Fahrzeuggeschwindigkeit eigene Algorithmen verbessern.

Software in den Steuergeräten

Möglich wird dies alles durch den Einsatz von Software im Kraftfahrzeug, der sich steigender Beliebtheit erfreut. Vorteile gibt es mehrere:

1. Kosten: Das entscheidende Kriterium in der Automobilindustrie sind die Fertigungskosten, die bekanntermaßen bei Software nicht anfallen. Komponenten aus der Mechanik oder der Elektronik, die durch Software ersetzt werden können, reduzieren damit automatisch die Fertigungskosten, die aufgrund der hohen Stückzahlen entscheidenden Einfluß auf die Produktkosten haben. Es wird auch akzeptiert, daß ein höherer Entwicklungsaufwand investiert wird, wenn dadurch die Fertigungskosten verringert werden können. Bedingung ist aber die Erfüllung der anderen Anforderungen wie z.B. Einhaltung der Entwicklungszeiten.

2. Zeit: Die Entwicklungszeiten neuer Modelle verkürzen sich auf ca. 2 Jahre. Dadurch entsteht umso häufiger die Anforderung, noch kurzfristig Änderungen an der Spezifikation in späten Enwicklungsstadien durchzuführen. Hier hat die Software den Vorteil, daß sie schnell geändert werden kann und keine weiteren Produktionsvorbereitungen notwendig sind. Im Extremfall kann die Software noch am Ende des Produktionsbandes in das Fahrzeug eingespielt werden. Auch Änderungen nach dem Start of Production` können beim Einsatz von Flash−PROMs an bereits produzierten Fahrzeugen relativ einfach vorgenommen werden. Diese Flexibilität kann zum Teil die verkürzten Entwicklungszyklen ausgleichen.

3. Gewicht: Durch die Forderungen der Gesellschaft nach Schonung der Energieressourcen und damit der Reduzierung des Energieverbrauchs, wird auch der Vorteil des nicht vorhandenen Gewichts (mit Ausnahme des notwendigen Programmspeichers) relevant. Wenn mechanische, elektrische oder elektronische Komponenten oder auch z. B. Kabelbäume durch Software ersetzt werden können, bedeutet dies Verbesserungen für die Energiebilanz.

4. Zuverlässigkeit: Im Gegensatz zu mechanischen und elektrischen Komponenten unterliegt die Software keinen

Alterungsprozessen. Daher ist auch nicht mit Ausfällen aus diesem Grunde zu rechnen. Wenn doch, dann handelt es sich um Entwicklungsfehler, wie sie auch bei allen anderen Komponenten auftreten können.

5. Service: Nach der Herstellung und dem Verkauf an den Kunden müssen regelmäßige Wartungsarbeiten durchgeführt werden. Die Software bietet hier die Möglichkeit, schon während des Betriebes Überwachungen der Systeme durchzuführen und fehlerhaftes Verhalten zu speichern. In der Werkstatt können diese Fehler ausgewertet werden und zusammen mit den Diagnosefunktionen die fehlerhaften Komponenten einfacher ermittelt werden.

6. Innovation: Ungefähr 90 Prozent der heutigen Innovationen im Kfz werden durch Elektronik inkl. Software entwickelt. Da insbesondere Innovationen ein wichtiges Alleinstellungsmerkmal bedeuten, steigt damit die Bedeutung der Elektronik und Software für alle Hersteller.

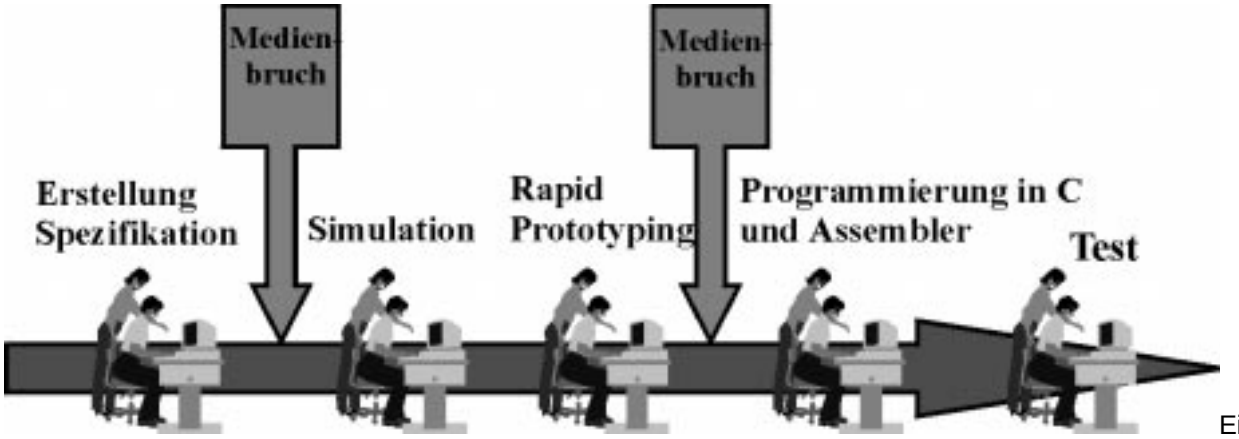
Aus diesen genannten Gründen, die alle für den Einsatz von Software sprechen, steigt der Anteil der Software am Fahrzeug stetig an.

Ebenso wie der Umfang nimmt damit auch die Bedeutung der Software zu. Die Algorithmen, die die Regelungsaufgaben im Kfz übernehmen, werden fast ausschließlich in Software realisiert. Das Know how, welches zum großen Teil bei den Autozulieferern angesiedelt ist, bekommt eine größere Beachtung auch durch die Hersteller selbst, da sie weiterhin die Systemführerschaft über das Fahrzeug besitzen möchten.

Stand der Softwareentwicklung/ Entwicklungsprozeß

Nach der Darstellung des Standes und der Bedeutung der Software, soll nun in einem zweiten Schritt der Entwicklungsprozeß der Software untersucht werden.

Während früher die Softwareentwicklung noch die Tat einzelner Ingenieure war, ist heute die Zusammenarbeit von vielen Beteiligten notwendig. Dabei gibt es ein breites Spektrum der Entwicklungsprozesse in der industriellen Praxis.



Ein typischer Entwicklungsprozeß beginnt mit der Übergabe der Spezifikation an den Automobilzulieferer. Es wird dann ein Prozeß, wie in Abbildung 1 skizziert, bis zur Freigabe des Steuergerätes inkl. Software durchlaufen.

Während vor einigen Jahren noch größere Anteile der Software in Assembler entwickelt wurden, hat sich mittlerweile zu einem großen Teil die Hochsprache C durchgesetzt. Stellenweise werden noch einige Assemblerbefehle eingefügt. Leider sind die C−Compiler nur für Microcontroller ausreichend, da diese im Gegensatz zu digitalen Signalprozessoren (DSPs) homogene Strukturen mit geringer Parallelität aufweisen. Die letzten beiden Jahrzehnte der Forschung auf dem Bereich des Compilerbaus haben hier die notwendigen Verbesserungen geliefert.

Die empfehlenswerte Sprache zur formalen Erfassung der Anwendung im Rechner hängt von der Struktur der Spezifikation ab. Folgende wesentliche Formen sind Stand der Technik:

1. Blockschaltbilder (MatLab/ SIMULINK)

   Die graphische Darstellung eignet sich insbesondere für datenflußorientierte Anwendungen. Verständlich für alle Beteiligten kann der Fluß der Daten von den Sensoren über Standardelemente wie z.B. Multiplizierer modelliert werden und bietet sich daher z.B. für DSPs an.

2. Zustandsübergangsgraphen (StateMate, MatLab/ Stateflow)

   Reaktive Systeme, die jeweils einen oder mehrere Zustände haben und deren Zustandsübergänge durch andere Eingangssignale initiiert werden, können ideal auf diese Weise beschrieben werden.

3. Sprache C (div. Compiler)

   Die Stärke einer Beschreibung in der textuellen Programmiersprache C liegt hier beim Einsatz von Algorithmen. Diese lassen sich effizient in dieser Form beschreiben.

4. Unified Modeling Language (UML)

   Eine weitere neue Form ist die Modellierung mit UML (Unified Modeling Language), die als Basis für die Sprache einen Standard gesetzt hat. Ebenso wie die Sprache Java ist der Vorteil der objektorientierten Sprachen u.a. in den Möglichkeiten der Vererbung und der Datenkapselung in Klassen zu suchen. Ein entscheidender Nachteil ist jedoch, daß nach heutigem Stand die generierten Programme länger sind als bei C−Programmen und einige Eigenschaften wie beispielsweise die Speicherverwaltung zu unvorhersehbaren dafür benötigten Zeitfenster führen, die zu Problemen bei der Einhaltung von harten Echtzeitbedingungen führen können.

Der Softwareentwicklungsprozeß beinhaltet heute noch Schwachstellen, die aufgrund mangelnder Fähigkeiten der Tools auftreten. Die Spezifikation wird teilweise mit Hilfe von Spezifikationstools wie MatLab/SIMULINK oder StateMate umgesetzt und daraus dann per Knopfdruck C−Code erzeugt. Die Qualität des automatisch generierten C−Codes reicht für die Simulation am Rechner und das Rapid Prototyping (= Test der Software auf leistungsfähiger, spezieller Hardware mit realer Mechanik) aus, jedoch noch nicht für den letztendlichen Einsatz im Steuergerät. Daher muß nochmals neu in der Sprache C implementiert werden.

Diese erneute Implementierung kostet Ressourcen und führt zu zusätzlichen Fehlerquellen.

Die ersten Toolhersteller sprechen davon, daß sie auch C−Code für das endgültige Steuergerät generieren können. Sicherlich wird das nicht von Beginn an problemlos und ohne jegliche Eingriffe und Nacharbeiten funktionieren. Jedoch ist damit die nächste Phase der Automatisierung eingeläutet worden und es ist mittelfristig davon auszugehen, daß die erneute Codierung per Hand abgelöst wird.

Doch auch dieses kann noch nicht der letzte Schritt sein. Immer stärker werden die Softwaremodule einzelner Lieferanten miteinander verzahnt. Dies liegt an der steigenden Vernetzung und den sich daraus ergebenden neuen Möglichkeiten der Innovationen und dem dauernden Bestreben, Kosten zu sparen. Ein Trend zu einer Zusammenlegung von Funktionen mehrerer Systeme in einem einzigen oder verteilt auf mehrere Steuergeräte ist daher nicht mehr weit und verlangt nach neuen Formen des Concurrent Engineering.

Zusammenarbeit zwischen Automobilhersteller, −zulieferer und Softwareanbietern

Die beschriebenen neuen Strukturen der Steuergeräte müssen durch Änderungen im Entwicklungsprozeß unterstützt werden. Während bisher die erstellte Software relativ isoliert entwickelt werden konnte und man auch die Tests der Steuergeräte unabhängig von anderen Beteiligten durchführte, tritt nun eine stärkere Verzahnung auf. Die entwickelte Software muß ggf. auch auf anderer Hardware lauffähig sein, sie muß mit anderen Modulen auf einer Hardware zusammenarbeiten können und es wird eine stärkere Kommunikation zwischen den Modulen auftreten.

Für die Software bedeutet dies höhere Anforderungen bei den Qualitätsmerkmalen Robustheit, Portabilität und Wiederverwendbarkeit.

Jedoch sind diese neuen Anforderungen nicht nur technischer Art, sondern verlangen auch nach neuen Organisationsformen zwischen Herstellern und Lieferanten.

Weiterhin werden zukünftig auch reine Softwareanbieter eine größere Bedeutung erlangen, die z.B. das Betriebssystem oder andere hardwareunabhängige Komponenten entwickeln. Durch diese parallele Softwareentwicklung in unterschiedlichen Organisationen und an unterschiedlichen Standorten, gekoppelt mit den Forderungen nach höherer Wiederverwendbarkeit, wird sich die Entwicklung stark verändern. Die Verfügbarkeit der Technik, die durch die Toolhersteller geliefert werden muß, ist notwendig, aber nicht hinreichend. Die Erfahrungen anderer Softwareentwicklungen zeigen, daß ein hoher Aufwand investiert werden muß, um die Tools auszuwählen, einzuführen, die Mitarbeiter zu schulen, und letztendlich die Prozesse umzustellen. Hier fällt der weitaus größte Aufwand an.

Trotzdem kann sich kein Unternehmen diesem Trend verschließen und erfolgreich sind die, die vielleicht nicht die ersten, aber die zweiten sind, die sich auf den Weg zu neuen Organisationsformen und −abläufen machen.

Standards

Die große Chance, die Entwicklungsressourcen einzusparen, liegt für die Automobilindustrie in der Möglichkeit, Standards setzen zu können. Wie beim CAN−Bus oder beim OSEK/VDX können Standards Mehrfachentwicklungen vermeiden und den gegenseitigen Austausch von Komponenten fördern. Dies gilt für alle Aspekte, die nicht wettbewerbsrelevant sind und wo auch nicht die andere Branchen, wie z.B. die Konsumgüterbranche bereits akzeptierte Standards gesetzt haben.

Forschung

1. Auch wenn die ersten Tools aus der Spezifikation automatisch C−Code generieren können, müssen noch viele Aspekte untersucht und Lösungen erforscht werden. Neben der Generierung des C−Code muß beispielsweise eine automatische Einbindung in die Betriebssysteme erfolgen.
2. Berücksichtigung sollten weiterhin die Anforderungen an die Realzeitbedingungen finden. Wie ist sichergestellt, daß die Reaktionszeiten eingehalten werden? Derzeit kann dies nur durch Tests verifiziert werden. Sinnvoll wäre hier die formale Spezifikation der zeitlichen Restriktionen und eine automatische Umsetzung im fertigen Produkt. Da Tests nie eine 100%−Abdeckung liefern, wird ansonsten das Risiko von Softwarefehlern bei steigendem Umfang und steigender Komplexität zu hoch werden. Denn es darf nicht vergessen werden, daß der Kunde trotz aller Neuerungen im Fahrzeug äußerst sensibel auf Fehler reagiert. Was nützt das schönste neue intelligente Radio, wenn ein Airbag unberechtigt zündet?
3. Weitere Arbeiten sind erforderlich, um die C−Code−Generierung nicht nur für Microcontroller, sondern auch für DSPs zu optimieren. Aufgrund ihrer heterogenen Struktur und der Möglichkeit der Parallelverarbeitung, müssen die Compiler dieses unterstützen. Die Forschung liefert noch erstaunliche Verbesserungen und man nähert sich hier erst der Qualität von per Hand erzeugtem Code.

Ausblick

Die Programmierung von Software für Steuergeräte wird mittelfristig nicht mehr nur eine an die Hardware angelagerte zusätzliche Dienstleistung sein, sondern die Verhältnisse werden sich umdrehen. Autmobilelektronikhersteller können dadurch, überspitzt gesagt, zu Softwarehäusern mit angelagerter oder sogar ausgelagerter Hardwareabteilung migrieren.

Die Mitarbeiterzahlen der Softwareentwicklung werden aufgrund der vielen Vorteile der Software im Kfz weiter steigen und sich damit die Entwicklungsprozesse mehr denen anderer größerer Softwareprojekte angleichen. Um die größer werdenden Softwaremodule zu entwickeln, muß ein Teil der zusätzlichen Arbeit durch den Einsatz von Tools aufgefangen werden. Parallelen kann man hier sicherlich z.B. zum Hardwareentwurf ziehen, wo sowohl für den Chipentwurf als auch z.B. für das Design von Platinen der Einsatz von Tools unabdingbar geworden ist.

Herr Stefan Steinke ist wissenschaftlicher Mitarbeiter des Lehrstuhls "computergestützter Entwurf integrierter Schaltungen" am Fachbereich Informatik der Universität Dortmund und forscht auf dem Gebiet der C−Codegenerierung von Spezifikationstools.