Automatisierung und Kreativität

Ist die Grenze beherrschbarer Komplexität erreicht?

Dr. Robert Hilbrich

1 Einführung

Der technologische Fortschritt unserer Gesellschaft spiegelt sich in der hohen Leistungsfähigkeit unserer Fahrzeuge wider. Piloten müssen ihre tonnenschweren Flugzeuge heute nur noch indirekt steuern, in dem sie über einen kleinen Joystick mit dem Steuerungssystem interagieren. Die Komplexität der richtigen Ansteuerung von Höhen-, Seiten- und Querruder bei gleichzeitiger Anpassung der Schubkraft tritt fast vollkommen in den Hintergrund. Autofahrer können sich durch eine Vielzahl von Assistenzsystemen bei der Fahrt unterstützen lassen. Unliebsame Aufgaben können sie an den *Autobahn-Piloten*, den *Stau*-Assistenten oder den *Park*-Assistenten delegieren. Der Schritt zu voll-automatischen Fahrzeugen ist nicht mehr weit.

Der *Computer* in seiner Rolle als automatisiertes Steuersystem nimmt dem Menschen immer mehr Aufgaben ab. Dies betrifft insbesondere Aufgaben, von denen unsere persönliche Sicherheit abhängt. Ein kleiner Fehler im Steuerungssystem eines Flugzeugs – zum Beispiel ein kleiner Zeitversatz zwischen der Ansteuerung des linken und rechten Höhenruders – kann bereits genügen, um das Flugzeug in einen aerodynamisch instabilen Zustand zu bringen. Im Auto können Ungenauigkeiten bei der Spur-Erkennung eines *Autobahn-Piloten* schnell katastrophale Folgen haben.

Trotz dieser Risiken übergeben wir die Verantwortung für unser Wohlergehen immer häufiger an Computersysteme und vertrauen auf deren Zuverlässigkeit. Die Statistik der Aus- und Unfälle belegt, dass diese Entscheidung rückblickend richtig war. In Anbetracht der sprunghaften technologischen Entwicklungen in den letzten Jahren stellt sich jedoch die Frage, ob dieses Vertrauen auch *zukünftig* gerechtfertigt ist. Sind unsere Entwicklungsmethoden und -werkzeuge auch zukünftig „mächtig“ genug, um die Komplexität der Entwicklung derartiger Systeme sicher zu beherrschen?

Die Steuersysteme im Auto und Flugzeug bestehen aus vielen vernetzten Computern. Sie interagieren mit zahlreichen Sensoren und Aktuatoren, um ihre Aufgaben zu erfüllen. Ein vollwertiges „Fly-by-Wire“ Steuersystem für Verkehrsflugzeuge setzt sich aus ca. 40 vernetzten Computern zusammen. Moderne Fahrzeuge beinhalten nicht selten mehr als 75 Computer, um die Vielzahl an Funktionen zu realisieren. Die Entwicklung derartiger

Steuersysteme gerät in Anbetracht dieser Komplexität an ihre Grenzen. Sie ist gefordert, alle Funktionen unter Berücksichtigung verschiedener denkbarer Umgebungsbedingungen und -zustände zu realisieren und dabei eine maximale Zuverlässigkeit bei minimalen Kosten zu erzielen.

Diese Komplexität ist ohne den Einsatz von Computer in der Entwicklung nicht mehr zu bewältigen. Obwohl sich das Einsatzspektrum der Computer in der Systementwicklung in den letzten Jahren deutlich erweitert hat, ist die grundlegende Arbeitsteilung zwischen dem menschlichen Entwickler und seinem elektronischen Assistenten noch immer unverändert. Während der Computer die wiederkehrenden *Routine-* und *Analyseaufgaben* übernimmt, bleiben die *kreativ-konstruktiven Tätigkeiten* dem Menschen überlassen. Der Mensch erschafft mit seiner Arbeit etwas Neues und Originelles – zum Beispiel die Architektur einer neuen Flugsteuerung oder einen Algorithmus für eine neue Assistenzfunktion. Erst die Transformation dieser neuen „Artefakte“ in Maschinencode und deren Analyse auf Programmierfehler wird durch den Computer durchgeführt. Aufgrund dieser Arbeitsteilung ist die Komplexität der zu entwickelnden Systeme durch die menschliche Verarbeitungskapazität begrenzt, denn der Entwickler kann nur eine begrenzte Zahl an Anforderungen an ein neu zu entwickelndes System verarbeiten und diese auch nur mit einer begrenzten Zahl zur Verfügung stehender „Lösungsbausteine“ in Übereinkunft bringen. Eine vollständige und fehlerlose Berücksichtigung *aller* Anforderungen bei der Systementwicklung übersteigt bereits heute in vielen Fällen die Verarbeitungskapazität der Entwickler.

Spätestens mit der Nutzung von *Mehrkernprozessoren*, die nicht nur eine stärkere Konzentration von Softwaremodulen auf einem Computer, sondern auch deren parallele Ausführung zur gleichen Zeit ermöglichen, ist die Grenze der sicher beherrschbaren Komplexität überschritten. Eine präzise und vollständige a priori Abschätzung *aller* Auswirkungen dieser Verdichtung und Parallelisierung überschreitet das Vermögen menschlicher Entwickler.

Derart hochkomplexe Systeme führen nicht nur die menschliche Entwicklungsarbeit an ihre Grenzen. Auch die automatisierten Verfahren zum Testen der Neuentwicklungen sind dieser hohen Komplexität nicht mehr vollumfänglich gewachsen. Wurde bisher die Korrektheit eines Systems durch die extrinsische Beobachtung seines funktionalen Verhaltens getestet, genügt dies heute nicht mehr. Die Zeit, die dann für die Prüfung aller System- und Umgebungszustände benötigt werden würde, übersteigt die vorgesehene Entwicklungszeit häufig um ein Vielfaches. Daher lässt sich der Nachweis über die *vollständige* Korrektheit eines Steuersystems unter *allen* Randbedingungen nicht mehr allein durch die Beobachtung von dessen Verhalten erbringen.

Dies bedeutet, mit Blick auf die gewachsenen Ansprüche an die Funktionalität von Steuersystemen ist die Grenze der Mächtigkeit der etablierten Entwicklungsmethoden und -werkzeuge erreicht. Ihre Weiterentwicklung ist die Voraussetzung, um auch zukünftig neue Steuersysteme mit einem erweiterten Funktionsumfang bei gleichzeitiger Aufrechterhaltung einer hohen Zuverlässigkeit zu konstruieren.

Ein vielversprechender Ansatz zur Lösung dieser Problematik liegt in der *Automatisierung* der kreativ-konstruktiven Tätigkeiten. Dies bricht mit der traditionellen Arbeitsteilung in der Entwicklung. Bei diesem Ansatz besteht die Aufgabe des menschlichen Entwicklers darin, die Anforderungen an das zu entwickelnde Steuersystem vollständig, präzise und fehlerfrei zu beschreiben. Im Anschluss konstruiert der Computer unter Zuhilfenahme der verfügbaren Lösungsbausteine automatisiert ein geeignetes „Artefakt“, das alle gestellten Anforderungen erfüllt.

Vorteile dieses Ansatzes liegen zum einen darin, dass Computer in der Lage sind, die gestellten Anforderungen *vollständiger* und *präziser* mit den technischen Möglichkeiten in Übereinkunft zu bringen. Die Grenze der beherrschbaren Komplexität lässt sich damit nach oben verschieben. Zum anderen eröffnet die Automatisierung der Konstruktion die Möglichkeit, die Korrektheit eines Systems auf der Grundlage seines Konstruktionsprozesses zu begründen anstatt dafür eine unvollständige Beobachtung des Systemverhaltens zu verwenden. Daher ist die Automatisierung der kreativ-konstruktiven Tätigkeiten eine zentrale Herausforderung für die ingenieurwissenschaftliche Forschung im Bereich der Systementwicklung, um die Weiterentwicklung von Steuersystemen zu ermöglichen und auch zukünftig deren Zuverlässigkeit zu gewährleisten.

NOCH NICHT ÜBERARBEITET

Die zugrunde liegende Dissertation befasst sich mit der automatisierten Konstruktion eines zentralen Artefakts sicherheitskritischer Systeme: der Zuweisung von Ressourcen, zum Beispiel der Rechenzeit auf einem Prozessor oder einem Teil des Arbeitsspeichers, zu den einzelnen Softwarekomponenten – oder anders formuliert: der Platzierung der Softwarekomponenten auf den Ressourcen des Systems. Diese Zuweisung ist scheinbar unspektakulär, doch dieser Eindruck trügt. Eine „richtige“ und „ausreichende“ Zuweisung von Ressourcen, ist für das Gesamtsystem *essentiell*. Sie ist die Grundvoraussetzung dafür, dass zeitkritische Steuerungsmodule, zum Beispiel das Airbagsteuergerät im Auto, zum „richtigen“ Zeitpunkt über „genügend“ Ressourcen zur Abarbeitung ihrer Steuerungslogiken verfügen.

Mit der Einführung von modernen „Mehrkern-“ Prozessoren, die eine parallele Ausführung von Softwarekomponenten erlauben, wurde die Möglichkeit zu einer starken Funktionsverdichtung in der Architektur von sicherheitskritischen Systemen geschaffen. Ein Mehrkern-Prozessor kann – im Gegensatz zu einem klassischen Ein-Kern-Prozessor – verschiedene Softwarekomponenten zur gleichen Zeit ausführen, so dass Geräte eingespart und damit die Effizienz des Systems gesteigert werden können. Die Ressourcenzuweisung ist damit nicht nur eine Grundvoraussetzung für die korrekte Funktionsweise des Systems, sondern zugleich auch der zentrale Ansatzpunkt für Optimierungen zur Steigerung der Ressourceneffizienz.

Die Synthese einer solchen Zuweisung erfordert eine vollständige Koordination aller Ressourcenzugriffe der Softwarekomponenten bereits zum Zeitpunkt der Entwicklung. Ihre Erstellung ist äußerst aufwändig und fehlerträchtig, denn sie ist sowohl mit der Komplexität der Anforderungen *aller* Softwarekomponenten als auch der Komplexität *aller*

Ressourcen der Hardwareplattform konfrontiert. Während die Ressourcenzuweisung bei realen Systemen in der Vergangenheit noch manuell konstruiert werden konnte, ist spätestens mit der Einführung von Mehrkern-Prozessoren ein Komplexitätsniveau erreicht, bei dem eine Fortsetzung tradierter Vorgehensweisen nicht länger zielführend ist.

Vor dem Hintergrund dieser Herausforderung wird in der zugrunde liegenden Dissertation ein automatisiertes Verfahren zur Konstruktion einer solchen Ressourcenzuweisung für Mehrkernprozessoren entwickelt und anhand von zwei Fallbeispielen für komplexe Systeme aus der Luft- und Raumfahrt erprobt.

Mittlerweile wird dieses Verfahren in leicht angepasster Form bei einem Hersteller ziviler Flugzeuge für die Optimierung der Systemarchitektur eines Flugzeugtyps eingesetzt. Durch die Verwendung dieses Verfahrens konnte die benötigte Zeitdauer für die Konstruktion einer vollständigen und fehlerfreien Zuweisung von ca. 12-15 Monaten auf einen Zeitraum von etwa 5 Minuten reduziert werden. Die Kürze der Konstruktionsdauer ermöglichte erstmals die Synthese einer Vielzahl unterschiedlicher Zuweisungen, so dass erstmalig überhaupt Entwurfsalternativen zur Optimierung verglichen und letztendlich Geräte eingespart werden konnten.

Insbesondere im Bereich der Luftfahrt ist die Reduzierung des Eigengewichts eines Systems von herausragender Bedeutung. Zur groben Abschätzung der Ersparnis wird zumeist die Annahme zugrunde gelegt, dass eine Gewichtsreduzierung des Flugzeugs um 1% zu einer Senkung des Kerosinbedarfs um 0,75 % führt. Konkret führt jedes eingesparte Kilogramm Gewicht bei einem zivilen Verkehrsflugzeug der aktuellen Generation zu einer Reduzierung der benötigten Treibstoffmenge in Höhe von etwa USD 3000 pro Jahr. Die entwickelten Verfahren zur Automatisierung der Entwicklung und Beherrschung der Komplexität sind daher ein wesentlicher Baustein, um die Ziele der Luftfahrtindustrie in Sachen Klimaschutz zu erreichen: ein CO_2-neutrales Wachstum ab dem Jahr 2020 sowie eine Halbierung der CO_2-Emissionen (in Relation zum Jahr 2005) bis zum Jahr 2050.

In den folgenden Abschnitten ...

2 Ergebnisse der Dissertation

Aufgabenstellung

Das Ziel der zugrunde liegenden Dissertation lag in der Entwicklung eines Verfahrens, mit dem Softwarekomponenten automatisiert und ohne Einschränkung bei der Erfüllung von Zuverlässigkeitsanforderungen auf die verschiedenen Computer eines Steuersystems platziert werden können. Dieses Verfahren sollte nicht nur die Konstruktion *einer* Platzierung erlauben, sondern auch *alternative Platzierungen* produzieren, so dass eine Optimierung des Gesamtsystems trotz großer Komplexität durchgeführt werden kann. Mit Hilfe der Optimierung sollte eine hohe „Funktionsdichte“ erzielt werden, so dass die Ressourcen der Computer, zum Beispiel die Prozessoren, möglichst *gemeinsam* durch verschiedene

Softwarekomponenten verwendet werden und damit eine hohe *Ressourceneffizienz* erzielt wird.

2.1 Ergebnisse

Die Ergebnisse der Dissertation untergliedern sich in einen *methodischen*, einen *konzeptuellen* und einen *praktischen Teil*. Im methodischen Teil wird ein Vorgehensmodell entwickelt, das einen neuartigen Ablauf der Entwicklungsschritte beschreibt und dabei insbesondere die neuen Möglichkeiten einer automatisierten Platzierung von Softwarekomponenten berücksichtigt.

Der konzeptuelle Teil bildet das Herzstück der Arbeit. Er enthält die im Rahmen dieser Arbeit entwickelten Formalismen, mit denen die Anforderungen und die Freiheitsgrade einer Platzierung eindeutig und maschinenlesbar spezifiziert werden können. Hier werden auch die neuen Transformationen vorgestellt, mit deren Hilfe ein formal spezifiziertes Platzierungsproblem auf einen generischen, mathematischen Formalismus – ein *Constraint Satisfaction Problem* – überführt werden kann. Diese Überführung ist die Voraussetzung für den Einsatz effizienter Algorithmen, mit denen Lösungen für ein derartiges *Constraint Satisfaction Problem* automatisiert erstellt werden können.

Im praktischen Teil wird die Umsetzung dieses Verfahrens in Form des Softwarewerkzeugs ASSIST dokumentiert. ASSIST ermöglicht es einem Fachexperten mit geringen IT-Kenntnissen, das zu entwickelnde Steuersystem und dessen Anforderungen textuell zu beschreiben und anschließend eine Platzierung aller Softwarekomponenten automatisiert konstruieren zu lassen. Eine Demonstration des Einsatzes von ASSIST am Beispiel einer Flugsteuerung für ein ziviles Verkehrsflugzeug sowie am Beispiel eines Steuersystems für ein Raumfahrzeug ist ein weiterer praktischer Bestandteil dieser Arbeit.

2.1.1 Ergebnisse im methodischen Teil

Die Platzierung von Softwarekomponenten ist ein wichtiger Dreh- und Angelpunkt in der Entwicklung eines Steuersystems, denn sie repräsentiert die Ergebnisse einer Abstimmung zwischen den Möglichkeiten der Hardware-Entwicklung und den Anforderungen der Software-Entwicklung. Während viele Aspekte der Entwicklung von Steuersystemen in diversen Normen und Industriestandards detailliert beschrieben sind, ist dies für die Konstruktion einer Platzierung von Softwarekomponenten nicht gegeben. Die notwendigen Schritte zu ihrer Erstellung werden – trotz ihrer zentralen Bedeutung für die korrekte Arbeitsweise des Steuersystems – nicht dokumentiert. Sie wird stattdessen als „korrekt“ und „vollständig vorliegend“ angenommen.

Im Rahmen der Dissertation wurden zunächst die Entwicklungsprozesse in der Praxis der Luft- und Raumfahrt analysiert, so dass ein Katalog mit Anforderungen an ein Vorgehen zur Konstruktion einer Platzierung erstellt werden konnte. Auf der Basis dieser

Anforderungen wurde im Anschluss ein konkretes Vorgehensmodell entwickelt. Es beschreibt die Abfolge der notwendigen Entwicklungsaktivitäten und Informationsflüsse zur Erstellung einer Platzierung.

Eine zentrale Herausforderung bei der Entwicklung des Vorgehensmodells bestand darin, die unterschiedlichen *Detaillierungsgrade* der Anforderungen zu verarbeiten, denn die Konstruktion einer Platzierung erfolgt nicht nur einmalig. Sie kann stattdessen als iterativer Prozess verstanden werden, bei dem die Ergebnisse schrittweise verfeinert und konkretisiert werden. Am Anfang der Entwicklung eines Steuersystems liegen beispielsweise meist nur sehr grobe Informationen über die Struktur und die Bedarfe der Softwarekomponenten vor. Zugleich sind zu diesem Zeitpunkt meist auch erst sehr wenige Informationen über die Fähigkeiten der Prozessoren verfügbar. Dennoch müssen bereits zu diesem Zeitpunkt grundlegende Entscheidungen zum Aufbau eines Steuersystems getroffen werden. Zum Beispiel muss bereits zu diesem Zeitpunkt abgeschätzt werden, wieviele Computer notwendig sind, damit das Steuersystem die geforderte Zuverlässigkeit bietet. Die Konstruktion einer Platzierung von Softwarekomponenten ist zu diesem Zeitpunkt ein wertvolles Hilfsmittel, um den Bedarf nach redundanten Computern und Prozessoren abzuschätzen.

Am Ende der Entwicklung sind die Anforderungen der Softwarekomponenten und Fähigkeiten der Prozessoren detailliert bekannt. Die Aufgabe der Platzierung liegt dann nicht nur in der groben Zuordnung der Softwarekomponenten zu den Prozessoren in den verschiedenen Computern eines Steuersystems. Sie muss darüber hinaus auch für jeden Prozessor, der von mehreren Softwarekomponenten benutzt wird, einen *Ablaufplan* erstellen. Dieser Ablaufplan bestimmt, welche Softwarekomponente zu welchem Zeitpunkt wieviel Rechenzeit auf dem Prozessor erhält. Dies ist insbesondere für zeitkritische Softwarekomponenten, wie zum Beispiel die Airbagsteuerung, essentiell, denn sie können ihre Funktion nur dann erfüllen, wenn sie zum *richtigen* Zeitpunkt *ausreichend* Rechenzeit für ihre Ausführung erhalten. Die Erstellung eines solchen Ablaufplans für *alle* Softwarekomponenten auf *allen* Prozessoren eines Steuersystems unter Beachtung *aller* Anforderungen zeitkritischer Komponenten, ist sehr komplex, aufwändig und fehlerträchtig. Ein geeignetes Vorgehensmodell muss daher sowohl in frühen Phasen mit groben Informationen als auch in späteren Entwicklungsphasen mit sehr detaillierten Informationen Unterstützung bieten.

Aufgrund dieser Anforderungen wurde die Konstruktion einer Platzierung im entwickelten Vorgehensmodell auf zwei Phasen aufgeteilt: eine *räumliche* und eine *zeitliche Platzierung*. Die räumliche Platzierung wird in frühen Entwicklungsphasen durchgeführt und platziert die Softwarekomponenten primär unter Berücksichtigung ihrer Zuverlässigkeitsanforderungen auf die einzelnen Computersysteme und Prozessoren. Es wird zu diesem Zeitpunkt beispielsweise sichergestellt, dass redundante Softwarekomponenten auch auf unterschiedlichen Prozessoren ausgeführt werden, um die Auswirkungen beim Ausfall eines Prozessors zu minimieren. In frühen Entwicklungsphasen sind in der Regel alle notwendigen Informationen verfügbar, um diese Entscheidungen sicher treffen zu

können und damit den Aufbau der grundlegenden Architektur eines Steuersystems zu begleiten.

2.1.2

3 Gesellschaftliche Bedeutung

- Arbeit schlägt die Brücke zwischen der Theorie und Praxis
- Werkzeug erlaubt es auch einem Fachexperten, die Forschritte aus dem Bereich der Informatik nutzen zu können
- Zunächst kann damit die Entwicklung von Systemen beschleunigt werden
- Systeme können optimiert werden, so dass weniger Ressourcen benötigt werden
- Weniger Fehler im System
- ...

Aber, diese Arbeit ist ein guter Gegenstand, um Fragen nach der Beherrschbarkeit heutiger Technologien zu stellen und zu diskutieren

- Was sind die Nachteile die sich daraus ergeben?
- Ist es nicht sinnvoll, wenn sich Menschen so lange mit den Anforderungen beschäftigen?
- Sollten wir Dinge konstruieren und nutzen, deren Aufbau das Verständnis des Menschen übersteigt?
- ...

4 Zusammenfassung