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 der neuen *Mehrkernprozessoren* ist die Grenze der sicher beherrschbaren Komplexität überschritten. Während klassische *Einkernprozessoren* die Ausführung von nur *einem* Softwaremodul zu jedem Zeitpunkt ermöglichten und daher auch nur eine kleine Anzahl an Softwaremodulen pro Prozessor integriert werden konnte, bieten *Mehrkernprozessoren* aufgrund ihrer deutlich gestiegenen Rechenleistung mehr Flexibilität. Mit Hilfe ihrer unabhängig arbeitenden Rechenkerne ermöglichen sie erstmals eine *parallele Ausführung* verschiedener Softwaremodule zur gleichen Zeit, so dass – zumindest theoretisch – wesentlich mehr Softwaremodule auf einem Prozessor integeriert werden können. Mittlerweile sind Mehrkernprozessoren mit über 1000 parallel arbeitenden Rechenkernen frei verfügbar. Im Vergleich zu den klassischen *Einkernprozessorenn* versprechen sie eine deutliche Steigerung der Rechenleistung bei einem äußerst geringen Energieverbrauch und stellen damit vielversprechende „Lösungsbausteine“ für die Steuersysteme der Zukunft dar. Allerdings überschreitet eine präzise und vollständige a priori Abschätzung *aller* Auswirkungen dieser starken Konzentration von Softwaremodulen auf einem Mehrkernprozessor und der daraus resultierenden Ressourcenkonkurrenz das Vermögen menschlicher Entwickler. Sie können die Zuverlässigkeit eines Systementwurfs auf Basis von Mehrkernprozessoren nur noch sehr eingeschränkt beurteilen.

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 und die Effizienz von Steuersystemen ist die Grenze der Mächtigkeit der etablierten Entwicklungsmethoden und -werkzeuge erreicht. Ihre Weiterentwicklung ist die Voraussetzung, um auch zukünftig effiziente 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.

Die zugrunde liegende Dissertation befasst sich mit der automatisierten Konstruktion eines zentralen „Artefakts“ eines Steuersystems: der Platzierung der verschiedenen Softwaremodule auf den Computern eines Steuersystems. Diese Zuweisung von „Ressourcen“, wie beispielsweise Rechenzeit auf einem Prozessorkern oder einem Bereich im Arbeitsspeicher, zu einem Softwaremodul ist scheinbar unspektakulär, doch dieser Eindruck trügt. Ein Steuersystem kann seine Funktion meist nur dann erbringen, wenn seine Softwaremodule die „richtigen“ Ressourcen in „ausreichender“ Menge zum „richtigen“ Zeitpunkt bekommen. Wenn das Softwaremodul zur Auslösung der Airbags im Auto beispielsweise nicht rechtzeitig Rechenzeit auf dem Prozessor bekommt, dann besteht die Gefahr einer verzögerten Auslösung. Die Schutzfunktion des Airbags ist damit gefährdet.

Neben der korrekten Berücksichtigung von solchen zeitkritischen Softwaremodulen, muss die Konstruktion einer Platzierung von Softwaremodulen auch die Auswirkungen auf die

Zuverlässigkeit in Betracht ziehen. So wird bei Steuersystemen eine hohe Zuverlässigkeit meist durch den Einsatz redundant arbeitender Softwaremodule realisiert. Sobald ein Modul bedingt durch einen technischen Fehler ausfällt, kann ein anderes Modul dessen Aufgaben übernehmen. Dies funktioniert selbstverständlich nur dann, wenn alle redundanten Softwaremodule auf *unterschiedlichen* Prozessoren platziert wurden. Eine „richtige“ und „ausreichende“ Zuweisung von Ressourcen ist daher für ein Steuersystem – und auch für viele andere Systeme – von *essentieller* Bedeutung. Die Entwicklung einer Zuweisung steht daher in dem Zielkonflikt, einerseits eine hohe Zuverlässigkeit durch die Verwendung möglichst vieler Prozessoren gewährleisten zu müssen und andererseits eine hohe Integration von Softwaremodulen auf möglichst wenigen Prozessoren zu realisieren.

Die Konstruktion einer Ressourcenzuweisung für alle Softwaremodule eines Steuersystems ist sehr aufwändig und fehlerträchtig. Dazu müssen bereits während der Entwicklung *alle* theoretisch möglichen Ausfälle technischer Komponenten adäquat berücksichtigt und *alle* Ressourcenzugriffe der zeitkritischen Softwaremodule koordiniert werden. Dies erfordert ein präzises und vollständiges Verständnis sowohl der Ressourcenbedarfe *aller* Softwaremodule als auch der Kapazitäten und Eigenschaften *aller* verfügbaren Ressourcen – zum Beispiel Prozessoren, Speicherbusse, Gerätecontroller, ... Während eine solche Ressourcenzuweisung in der Vergangenheit noch manuell konstruiert werden konnte, ist dies bei Mehrkernprozessoren nicht länger sinnvoll möglich. Das resultierende Komplexitätsniveau macht eine Weiterentwicklung der etablierten Verfahren zwingend notwendig. Vor dem Hintergrund dieser Herausforderung wird in der zugrunde liegenden Dissertation ein *automatisiertes Verfahren* zur Konstruktion einer Ressourcenzuweisung entwickelt. Dieses Verfahren wird als Software-Werkzeug umgesetzt und anhand von zwei Fallbeispielen aus der Luft- und Raumfahrt erprobt.

Die gesellschaftliche Bedeutung der Ergebnisse der zugrunde liegenden Arbeit ist zweischichtig. Zum einen belegen die Ergebnisse, dass sich auch komplexe kreativ-konstruktive Entwicklungsaufgaben automatisieren lassen. In kürzerer Zeit können effizientere und optimierte Steuersysteme entwickelt werden, ohne dabei unbeabsichtigte Beeinträchtigungen in der Zuverlässigkeit befürchten zu müssen. Das Niveau der beherrschbaren Komplexität wird durch das entwickelte Verfahren signifikant angehoben.

Mittlerweile wird dieses Verfahren in leicht angepasster Form bei einem Hersteller ziviler Flugzeuge zur Optimierung der Systemarchitektur eines Flugzeugtyps produktiv eingesetzt. Durch den Einsatz des Software-Werkzeugs konnte die benötigte Zeitdauer für die Konstruktion *einer* vollständigen und fehlerfreien Zuweisung von ehemals ca. 12-15 Monaten auf den Zeitraum von etwa 5 Minuten reduziert werden. In der Folge konnten erstmals viele unterschiedliche Zuweisungen konstruiert und miteiander verglichen werden, so dass letztendlich auch Geräte eingespart werden konnten. Dies resultiert in einem geringeren Gesamtgewicht des Flugzeugs und wird zu einer Reduzierung der CO_2 Emissionen beitragen.

Zum anderen bildet diese Arbeit einen wichtigen Ausgangspunkt für eine längst überfällige Debatte über die Beherrschbarkeit der Komplexität zukünftiger Computersysteme.

Was passiert mit der technischen Kompetenz einer Gesellschaft, wenn die Entwicklung der zentralen Steuersysteme unserer zukünftigen Fahrzeuge in immer kürzerer Zeit durch Computer durchgeführt wird? Besteht dann nicht die Gefahr, dass nicht nur die Anwender, sondern auch die Entwickler eines Steuersystems, dieses immer mehr als „Black Box“ begreifen? Ist es dann noch immer zu verantworten, dass diese Systeme immer mehr Verantwortung für unsere Sicherheit übernehmen?

Im Anschluss werden zunächst ... Anschließend ... Zum Abschluss wird eine kurze Zusammenfassung der Darstellungen gegeben.

2 Ergebnisse der Dissertation

2.1 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.

2.2 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.

2.2.1 Methodischer Teil: Ein neues Vorgehensmodell

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 mehr in einer groben Zuordnung der Softwarekomponenten zu den Prozessoren in den verschiedenen Computern eines Steuersystems. Sie muss nun 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.

Die zeitliche Platzierung koordiniert dagegen den Zugriff mehrerer Softwarekomponenten auf die Rechenzeit eines Prozessors in der zeitlichen Dimension. Nach der Zuordnung der Softwarekomponenten zu den Prozessoren wird in dieser Phase ein Ablaufplan für deren Ausführung auf den Prozessoren erstellt. Dieser Ablaufplan muss die zeitlichen Anforderungen aller Softwarekomponenten berücksichtigen und mit der Verarbeitungsgeschwindigkeit des Prozessors in Relation setzen, denn auf einem langsamen Prozessor benötigt eine Softwarekomponente wesentlich mehr Rechenzeit, um „vollständig“ ausgeführt zu werden.

Darüber hinaus finden im entwickelten Vorgehensmodell auch zusätzliche Aktivitäten zur Analyse des Zeitverhaltens einer Softwarekomponenten auf einem spezifischen Prozessor sowie Aktivitäten zur Validierung der erstellten Platzierung Berücksichtigung. Der Schwerpunkt der Dissertation liegt jedoch auf der Entwicklung eines Verfahrens zur Automatisierung der räumlichen und zeitlichen Platzierung von Softwarekomponenten.

Einfügen der Grafik S. 54 aus der Dissertation

2.2.2 Konzeptioneller Teil: Formalisierung einer Platzierung

Im konzeptionellen Teil der Arbeit werden die Grundlagen für die Automatisierung geschaffen. Dazu wird zunächst die Frage der Korrektheit einer Platzierung adressiert: wann kann eine Platzierung als „fehlerfrei“ angesehen werden? Welche Anforderungen muss sie dazu erfüllen?

Bei Steuersystemen in Fahrzeugen sind es vor allem zwei Aspekte, die neben der reinen Funktionalität, von großer Bedeutung für dessen Korrektheit sind: *Zuverlässigkeit* und *Echtzeitfähigkeit*. Zuverlässigkeit beschreibt die Robustheit des Systems in Gegenwart von Fehlern und Ausfällen, während Echtzeitfähigkeit die „rechtzeitige“ Ausführung aller zeitkritischen Funktionen beinhaltet. Eine Platzierung ist daher erst dann korrekt, wenn sie alle Zuverlässigkeits- und Echtzeitanforderungen erfüllt. In der Dissertation wurden die relevanten Korrektheitskriterien einer Platzierung für die Aspekte Zuverlässigkeit und Echtzeitfähigkeit aus den gängigen Normen und Standards der Luft- und Raumfahrt abgeleitet und formalisiert. Mit Hilfe dieser Kriterien lässt sich ein Platzierungsproblem bei einem Steuersystem präzise und eindeutig beschreiben. Dies bildet die Grundlage für den nächsten konzeptionellen Schritt: die Entwicklung eines Verfahrens zur *Lösung* eines Platzierungsproblems.

Der Ansatz zur Lösung eines Platzierungsproblems besteht in dessen Modellierung als mathematisches *Constraint Satisfaction Problem* – auch Bedingungserfüllungsproblem genannt. Dies ist eine spezielle Klasse von mathematischen Problemstellungen, bei denen für eine Menge von ganzzahligen Variablen eine gültige Belegung gesucht wird, die allen gestellten Randbedingungen (*engl.: Constraints*) genügt. Der große Vorteil dieser speziellen Modellierungsform besteht darin, dass sie rein deskriptiv erfolgt. Es muss lediglich spezifiziert werden, welche Anforderungen eine korrekte Platzierung erfüllen muss, aber nicht, wie eine korrekte Platzierung gefunden werden kann. Die Synthese von Lösungen wird stattdessen leistungsfähigen und sehr effizienten Algorithmen überlassen. Der zentrale Beitrag dieser Arbeit besteht daher in der Modellierung eines räumlichen und zeitlichen Platzierungsproblems als *Constraint Satisfaction Problem*. Dies ist die Voraussetzung, um die konstruktiv-kreative Tätigkeit der Platzierung von Softwarekomponenten mit Hilfe eines Computers zu automatisieren.

Die Konstruktion einer korrekten – und damit realisierbaren – Platzierung ist in der Praxis noch nicht ausreichend. Eine Platzierung muss darüber hinaus auch verschiedenen projektspezifischen Optimierungskriterien genügen. Sie muss beispielsweise die zukünftige Erweiterbarkeit des Steuersystems erleichtern, in dem sie die verfügbaren Ressourcen nicht zu 100% auslastet, um eine Reserve für zukünftige funktionale Erweiterungen bei den Softwarekomponenten zu schaffen. Die Herausforderung besteht daher in der Konstruktion einer korrekten *und* optimierten Platzierung.

Obwohl bereits äußerst effiziente Algorithmen zur Lösung eines Platzierungsproblems eingesetzt werden, ist der vorliegende Lösungsraum bei realistischen Steuersystemen noch immer zu groß, um in sinnvollen Zeiträumen *vollständig* durchsucht zu werden

und damit eine korrekte und optimale Platzierung zu finden. Selbst bei einem kleinen System mit nur 10 Prozessoren und lediglich 25 Softwarekomponenten gibt es theoretisch bereits 10^{25} Platzierungsmöglichkeiten für die einzelnen Softwarekomponenten. Selbstverständlich sind nicht alle Platzierungen realisierbar und in Hinblick auf die gestellten Optimierungskriterien geeignet, aber welche dieser Lösungen wären es? Selbst wenn man jede Möglichkeit innerhalb einer Hundertstelsekunde diesbezüglich analysieren könnte, würde es noch immer mehr als 3 170 979 198 376 458 Jahre dauern, bis man den vorliegenden Lösungsraum *vollständig* durchsucht hat. Aus diesem Grund wurden im Rahmen der Dissertation hochoptimierte Strategien und spezielle Heuristiken entwickelt, um auch bei einer *unvollständigen* Durchsuchung des Lösungsraums, möglichst viele korrekte und zugleich auch „gute“ Platzierungen zu finden.

2.2.3 Praktischer Teil: Softwarewerkzeug und Fallbeispiele

3 Gesellschaftliche Bedeutung

- ASSIST ist frei verfügbar; Quellcode ist offen und für jeden nutzbar
- Nächster Schritt
- 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?
- Meta-Programmer - eine kleine Elite erstellt die Algorithmen, deren Komplexität sich dem Verständnis einem Großteil der Bevölkerung entzieht.

4 Zusammenfassung