Potenziale und Risiken der Selbstoptimierung für die Verlässlichkeit mechatronischer Systeme[1]

Ursula Frank*, Holger Giese, Thomas Müller*, Simon Oberthür*, Christoph Romaus, Matthias Tichy, Henner Vöcking
Universität Paderborn
**Heinz Nixdorf Institut*
Fürstenallee 11, D-33102 Paderborn
Tel.: +49 (0) 5251 / 60 - 6186, Fax.: +49 (0) 5251 / 60 - 6278
E-Mail: thomas.mueller@hni.upb.de

Zusammenfassung

Fortschrittliche mechatronische Systeme der nächsten Generation werden sich mit Hilfe komplexer Software intelligent verhalten und selbstständig ihr Verhalten durch Bildung von Gemeinschaften autonomer Agenten verbessern können. Der Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) untersucht solche mechatronischen Systeme. Mit Hilfe ihrer inhärenten Intelligenz bestimmen die Teilsysteme dabei autonom neue oder auch veränderte Ziele und führen nachfolgend entsprechende Verhaltensanpassungen durch. In diesem Beitrag werden die aus dem Paradigma der Selbstoptimierung resultierenden Potenziale zur Verbesserung der Verlässlichkeit des Systems sowie mögliche Gefahren diskutiert. Neben den identifizierten Potenzialen stellen wir in diesem Beitrag Konzepte und Methoden vor, die im SFB 614 entwickelt wurden, um Chancen zu nutzen und die Risiken zu reduzieren. Insbesondere erläutern wir neben der generellen Berücksichtigung der Verlässlichkeit im Zielsystem eines selbstoptimierenden Systems Korrektheitsbeweise für die selbstoptimierende Software, ein Monitoring-Konzept zur Einordnung des Systemzustands in vier Sicherheitsbereiche, ein flexibles Ressourcenmanagement für die Notfallbehandlung und verschiedene Ansätze für Potenziale im Bereich der selbstoptimierenden Wartung, der Verfügbarkeitssteigerung durch Verhaltensanpassungen und der Selbstheilung.

Schlüsselwörter

Mechatronische Systeme, Selbstoptimierung, Verlässlichkeit, Methoden/Konzepte

---

[1] Diese Arbeit ist im Sonderforschungsbereich (SFB) 614 „Selbstoptimierende Systeme des Maschinenbaus“ der Universität Paderborn entstanden. Die Autoren danken der Deutschen Forschungsgemeinschaft für die Förderung des Vorhabens.

1 Einleitung

Fortschrittliche mechatronische Systeme [BSDB00] entwickeln heutzutage durch Kombination der klassischen Bereiche Mechanik und Elektronik mit der Softwaretechnik anpassungsfähige und somit leistungsfähigere Lösungen für komplexe technische Systeme. Es wird erwartet, dass die entsprechenden mechatronischen Systeme der nächsten Generation sich mit Hilfe komplexer Software intelligent verhalten und selbstständig ihr Verhalten durch Anpassung der Ziele und Bildung von Gemeinschaften autonomer Agenten verbessern. Diese vernetzten mechatronischen Systeme können sich durch Anpassung ihrer Netzwerktopologie und komplexe Echtzeitkoordination eine koordinierte Selbstoptimierung erreichen [GBSO04], die weit über bisherige Formen selbstadaptiver Systeme [SKB98, MGPK99, OGT+99] hinausgeht, die nur lokal und mit festen Zielen operieren.

Der Sonderforschungsbereich 614 „Selbstoptimierende Systeme des Maschinenbaus“ (SFB 614) untersucht solche selbstoptimierenden mechatronischen Systeme, die eine inhärente Intelligenz besitzen. Diese erlaubt es ihnen, autonom neue Ziele zu bestimmen oder auch Zielveränderungen vorzunehmen und darauf mit Verhaltensanpassung zu reagieren (vgl. Abschnitt 1.1).

Aus der Perspektive der Sicherheitstechnik führen solche selbstoptimierenden und vernetzten mechatronischen Systeme zu zusätzlichen Potenzialen zur Verbesserung der Verlässlichkeit (vgl. Abschnitt 1.2), aber auch zu Gefahren für die Verlässlichkeit aufgrund der erhöhten Komplexität. Das Paradigma der Selbstoptimierung kann dabei als Chance für die Erhöhung der Verlässlichkeit genutzt werden, wenn zusätzliche Funktionalität und insbesondere Variabilität von selbstoptimierenden Systemen dazu genutzt wird, Gefahren zur Laufzeit zu begegnen (vgl. Abschnitte 2 und 4). Auf der anderen Seite steigt durch Selbstoptimierung aber auch die Komplexität des mechatronischen Systems. Eine höhere Komplexität bedeutet in der Regel auch eine größere Gefahr von systematischen Fehlern bzw. höhere Fehleranfälligkeit (vgl. Abschnitt 3). Neu bei selbstoptimierenden Systemen ist zudem, dass infolge der selbständigen Zielanpassung das resultierende Verhalten der Systeme unter Umständen nicht oder nur mit sehr großem Aufwand im Vorfeld zu bestimmen ist.

Aus diesem Paradigma der Selbstoptimierung resultieren Potenziale zur Verbesserung der Verlässlichkeit des Systems, die die Teilbereiche Sicherheit, Zuverlässigkeit, Verfügbarkeit und Vertraulichkeit umfassen. Gleichzeitig beinhaltet es allerdings auch gewisse Risiken. Neben den identifizierten Potenzialen stellen wir in diesem Beitrag Konzepte und Methoden vor, die im SFB 614 entwickelt wurden, um Chancen zu nutzen und die Risiken zu reduzieren.

1.1 Grundlagen der Selbstoptimierung

Im SFB 614 wird unter Selbstoptimierung eines technischen Systems die endogene Anpassung der Ziele des Systems auf veränderte Einflüsse und die daraus resultierende zielkonforme autonome Anpassung der Parameter und ggf. der Struktur und somit des Verhaltens dieses Systems verstanden. Damit geht Selbstoptimierung über die bekannten Regel- und Adaptionsstrategien wesentlich hinaus; Selbstoptimierung ermöglicht handlungsfähige Systeme mit inhärenter Intelligenz, die in der Lage sind, selbständig und flexibel auf veränderte Betriebsbedingungen zu reagieren [FGK+04].

Das Zusammenwirken der wesentlichen Aspekte eines selbstoptimierenden Systems ist in Bild 1 dargestellt. Auf Basis der Einflüsse bestimmt das selbstoptimierende System die aktiv zu verfolgenden internen Ziele, d. h. die Optimierungsziele. Diese internen Ziele beruhen auf externen Zielen, die von außen dem System gestellt werden (z. B. durch den Benutzer oder andere Systeme), und inhärenten Zielen, die den Entwurfszweck des Systems widerspiegeln. Die Anpassung der Ziele bedeutet, dass die Gewichtung der Ziele verändert wird, neue Ziele hinzukommen oder vorhandene Ziele entfallen und nicht mehr verfolgt werden. Diese Anpassung der Ziele führt zu einer Anpassung des Systemverhaltens. Die notwendige Verhaltensanpassung wird durch Parameter- und ggf. durch Strukturanpassungen erreicht. Unter einer Parameteranpassung wird die Anpassung eines Systemparameters verstanden, z. B. das Ändern eines Regelparameters. Strukturanpassungen betreffen die Anordnung und Beziehungen der Elemente eines Systems. Folgende drei Aktionen drücken den Ablauf der Selbstoptimierung aus: 1. Analyse der Ist-Situation, 2. Bestimmung der Systemziele, 3. Anpassung des Systemverhaltens.

Die die Selbstoptimierung realisierende Informationsverarbeitung ist sehr komplex. Zur Beherrschung dieser Komplexität wurde im SFB 614 das Strukturierungs- und Architekturkonzept des Operator-Controller-Moduls (OCM) entwickelt. Es entspricht aus informationstechnischer Sicht einem Agenten. Das OCM gliedert sich in die drei Ebenen Controller, Reflektorischer Operator und Kognitiver Operator[2] [OHG04].

Controller: Dieser Regelkreis wird als „Motorischer Kreis“ bezeichnet. Er greift direkt auf das technische System (Strecke) ein. Dazu analysiert er kontinuierlich Messsignale, berechnet Stellsignale und übergibt diese unter harten Echtzeitbedingungen an die Aktoren des technischen Systems. Der Controller kann aus mehreren Reglervarianten bestehen. Zwischen diesen Varianten kann in einem Schritt

[2] Die Begriffe der Ebenen wurden gezielt für die selbstoptimierende Informationsverarbeitung entwickelt. In industriellen Anwendungen werden zum Teil vergleichbare, aber häufig enger zu verstehende Begriffe verwendet; z. B. Regler, Konfigurator und Optimierer.

umgeschaltet werden; erforderliche Überblendungsmechanismen u. ä. werden durch separate Reglerelemente realisiert. Grundlegende Funktionen des Controllers sind demnach *Beobachten und Regeln*.

Reflektorischer Operator: Er greift nicht direkt in das Verhalten der Strecke ein sondern überwacht und rekonfiguriert den Controller. Dabei verändert er das Verhalten des Controllers durch Anstoßen von Parameter- oder Strukturänderungen. Bei Strukturänderungen wird zwischen Reglern und entsprechenden Kontroll- und Signalflüssen umgeschaltet. Kombinationen aus Reglern, Schaltelementen und zugehörigen Kontroll- bzw. Signalflüssen werden als Controllerkonfigurationen bezeichnet. Die Umschaltung zwischen den Konfigurationen erfolgt durch eine Konfigurationssteuerung. Diese legt fest, wann welche Konfiguration eingesetzt werden soll und bestimmt die Art und Weise der Umschaltung. Der Reflektorische Operator arbeitet überwiegend ereignisorientiert und genügt harten Echtzeitbedingungen. Er bietet eine Schnittstelle zwischen dem Kognitiven Operator, dessen Elemente weichen Echtzeitbedingungen genügen und dem unter harten Echtzeitbedingungen arbeitendem Controller. Schließlich ermöglicht der Reflektorische Operator die Echtzeitkommunikation mit anderen OCM. Hauptfunktionen des Reflektorischen Operators sind folglich *Überwachen und Umschalten*.

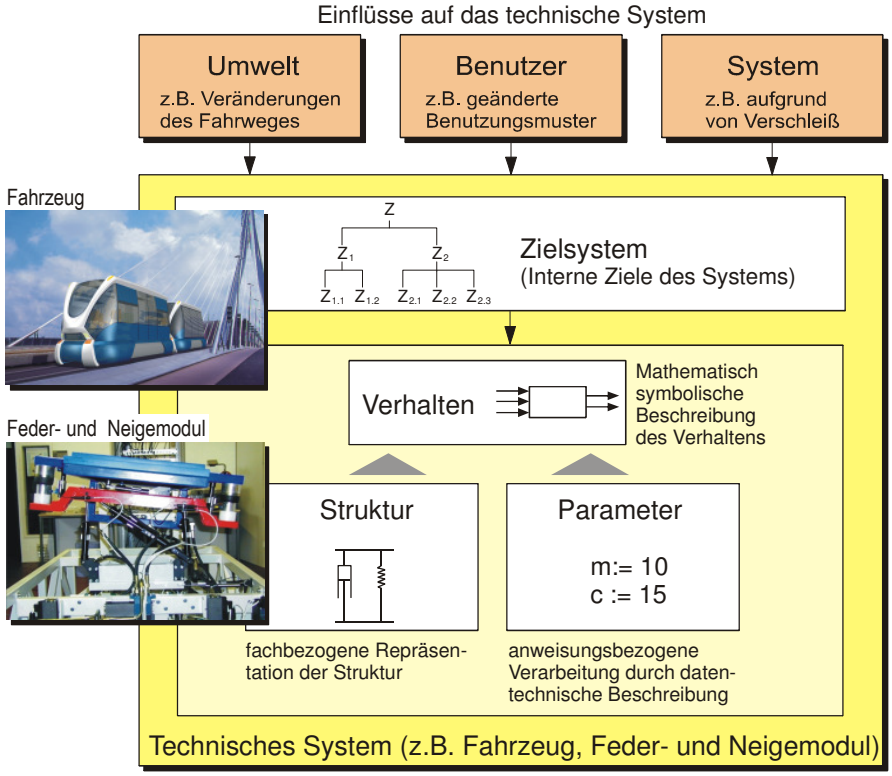

Abbildung 1: Die Hauptaspekte eines selbstoptimierenden Systems

Kognitiver Operator: Mit Hilfe von Verfahren wie Lernverfahren, modellbasierten Optimierungsverfahren oder dem Einsatz wissensbasierter Systeme ermittelt das System Wissen über sich und sein Umfeld und nutzt dieses Wissen zur Verbesserung des eigenen Verhaltens. Ziel ist die Nutzung von kognitiven Fähigkeiten zur Realisierung der Selbstoptimierung. Der Kognitive Operator übernimmt somit Funktionen wie *Simulieren und Optimieren*.

1.2 Grundlagen der Verlässlichkeit

Begriffe im Kontext der Verlässlichkeit erfahren häufig keine einheitliche Verwendung. Daher wird im Folgenden die Sichtweise dargestellt, die der Verwendung in diesem Beitrag und unserem Verständnis im SFB 614 entspricht. Dieses ist darüber hinaus notwendig, um sowohl Risiken als auch Potenziale detaillierter analysieren und zuordnen zu können, die durch den Einsatz von Selbstoptimierung entstehen. Die Begriffserläuterungen folgen dem in der Informatik verbreiteten Verständnis nach [Lap92] das ebenfalls auf dem Gebiet der Mechatronik angewendet wird [KP02]. Die Verlässlichkeit wird als übergeordneter Begriff verstanden, der die beschreibenden Attribute Verfügbarkeit, Funktionsfähigkeit/Zuverlässigkeit, Sicherheit und Vertraulichkeit integriert. Die englische Übersetzung verwendet dafür das Wort „dependability“ mit der Bedeutung, dass Vertrauen in die erbrachte Leistung des Systems gesetzt werden kann. Die Bedeutungen der Attribute werden im Folgenden dargestellt:

- *Verfügbarkeit*: Verlässlichkeit in Bezug auf das „Bereit sein“ zur Benutzung.
- *Funktionsfähigkeit/Zuverlässigkeit:* Verlässlichkeit mit Bezug auf die Kontinuität der Leistung – Überlebenswahrscheinlichkeit.
- *Sicherheit:* Verlässlichkeit mit Bezug auf das Nichtauftreten von kritischen Ausfällen.
- *Vertraulichkeit:* Verlässlichkeit mit Bezug auf die Verhinderung von nicht autorisiertem Zugriff und/oder Behandlung von Informationen [Lap92].

2 Verlässlichkeit als Ziel eines selbstoptimierenden Systems

In Abschnitt 1.1 wurde bereits die Fähigkeit eines selbstoptimierenden Systems erläutert, Ziele selbständig auszuwählen, anzupassen oder sogar neu zu entwickeln. Innerhalb dieses Prozessschrittes der Selbstoptimierung kann auch die Verlässlichkeit berücksichtigt werden. Generell existieren zur Gewährleistung bzw. Erhöhung der Verlässlichkeit zwei verschiedene Optionen, die sich nicht gegen-

seitig ausschließen, sondern sich ergänzen: Die erste Option berücksichtigt die Verlässlichkeit als inhärentes Ziel, das dem System bereits während des Entwurfs vorgegeben wird. Die zweite Option beschreibt die Verlässlichkeit als internes Ziel, das aus der Analyse des Umfelds und des Systems unter Berücksichtigung der inhärenten Ziele resultiert. In der Regel ist das inhärente Ziel allgemeiner als das interne Ziel und spiegelt die klassische Vorgehensweise zur Berücksichtigung der Attribute der Verlässlichkeit wider.

Verlässlichkeit als inhärentes Ziel: Im Zielsystem eines selbstoptimierenden Systems ist die Verlässlichkeit ein inhärentes Ziel, wenn sie dem System bereits während des Entwurfs vorgegeben bzw. impliziert wurde. Dies kann einerseits abstrakt und indirekt realisiert werden, andererseits auch durch quantifizierte Vorgaben. Die abstrakten, indirekten Ziele Sicherheit, Zuverlässigkeit und Verfügbarkeit könnten z. B. durch Überdimensionierung von Bauteilen erreicht werden. Ein solches Ziel braucht und kann selbstverständlich nicht mehr innerhalb eines Optimierungsprozesses berücksichtigt werden. Andererseits, und im Kontext der Selbstoptimierung wesentlich bedeutender, können quantifizierte Ziele Anwendung finden, z. B. durch vorgegebene Grenzwerte, die bei Überschreitung zu Aktionen wie vordefinierten Notfallmechanismen führen. Diese Art von inhärenten quantifizierten Zielen der Verlässlichkeit besitzt Potenzial zur Selbstoptimierung. Ein quantifiziertes inhärentes Ziel kann beim Ablauf des Selbstoptimierungsprozesses berücksichtigt werden und in Abhängigkeit mit externen Zielen, also vom Benutzer vorgegebenen Zielen sowie den Ergebnissen der Umfeld- und Systemanalyse, zu einem internen Ziel werden. Für das interne Ziel können dann z. B. Grenzwerte oder Notfallmechanismen in optimierter Form bzgl. der aktuellen oder vorhersehbaren Situation gelten.

Verlässlichkeit als internes Ziel: Ein internes Ziel im Bereich der Verlässlichkeit liegt vor, wenn es sich um ein Resultat handelt, das aus der Phase „Bestimmung der Systemziele“ im Selbstoptimierungsprozess hervorgegangen ist. Es handelt sich daher um Ziele, die durch Generierung, Auswahl, Anpassung oder Gewichtung inhärenter oder externer Ziele entstanden sind und die das System zu einem konkreten Zeitpunkt verfolgt. Interne Ziele der Verlässlichkeit sind folglich Ziele, die sich flexibel auf sich ändernde Umweltbedingungen, Eingriffe des Benutzers oder Einwirkungen des Systems anpassen lassen, und die das System im Sinne der Selbstoptimierung verwenden kann.

Beispielhaft kann die Zuverlässigkeit als Ziel eines selbstoptimierenden Systems erläutert werden: Im Sinne der Zuverlässigkeit könnte das quantifizierte inhärente Ziel beim Entwurf eine definierte Lebensdauererwartung sein. Diese ist abhängig von Betriebs- und Umfeldbedingungen, die zum Entwurf nur begrenzt bekannt

sind. Während des Betriebs kann das System nun die tatsächlichen Bedingungen analysieren. Darauf basierend, bestehen verschiedene Optionen für das System, sich selbst zu optimieren:

- Das Zielsystem muss nicht angepasst werden, da die Ziele nicht gefährdet sind.
- Die neu berechnete Lebensdauererwartung passt das bisherige Ziel an, so dass z. B. nur noch eine reduzierte Lebensdauer im internen Zielsystem verfolgt wird; evtl. müssen daraufhin andere Ziele angepasst werden.
- Das bestehende Ziel der Lebensdauererwartung bleibt erhalten und wird durch Systemanpassung, z. B. durch Reduzierung der Betriebsbelastungen erreicht. Auch in diesem Fall könnten andere Ziele wahrscheinlich nur schlechter erreicht werden.

Zusammenfassend lässt sich die Verlässlichkeit in einem selbstoptimierenden System sowohl durch inhärente, beim Entwurf vorgegebene Ziele berücksichtigen als auch durch interne Ziele, die im Prozess der Selbstoptimierung Eingang finden. Während Erstgenannte die Grundlage für das folgende Kapitel im Sinne von Konzepten für verlässliche selbstoptimierende Systeme bilden, beinhalten Letztgenannte das Potenzial, das die Selbstoptimierung im Bereich der Verlässlichkeit eröffnet.

3 Konzepte für verlässliche selbstoptimierende Systeme

Im Folgenden skizzieren wir verschiedene Ansätze, die es ermöglichen Selbstoptimierung einzusetzen und trotzdem die Verlässlichkeit des Gesamtsystems zu gewährleisten.

3.1 Korrekte Software

Das in Abschnitt 1.1 vorgestellte OCM ist das grundlegende Element für die Architektur eines selbstoptimierenden Systems.

Die Entkopplung von selbstoptimierendem Verhalten im Kognitiven Operator unter weichen Echtzeitrestriktionen sowie sicherem zeit-diskreten Verhalten im Reflektorischen Operator und kontinuierlichem Verhalten im Controller unter harten Echtzeitbedingungen verhindert unerwünschte Effekte auf die Sicherheit des mechatronischen Systems [OHG04]. Die Sicherheit muss daher nur noch auf Ebene des Reflektorischen Operators und des Controllers verifiziert werden.

Das zeit-diskrete Verhalten des Reflektorischen Operators wird typischerweise mittels Zustandsdiagrammen wie Statecharts oder Matlab/Stateflow modelliert. Hiermit kann die Änderung eines diskreten Zustands in Reaktion auf eingehende Nachrichten, Zeitrestriktionen, etc. spezifiziert werden. Für diese Modelle wurde in [GTB+03] eine Verifikationstechnik vorgestellt, die es ermöglicht, die Sicherheit des Verhaltens mathematisch zu beweisen. Im Besonderen können mit dieser Technik auch große Systeme durch geeignete Zerlegungs- und Kompositionsmechanismen formal verifiziert werden.

Verhaltensanpassung ist der dritte Schritt der Selbstoptimierung (vgl. Abschnitt 1.1). Verhaltensanpassung kann u. a. durch eine Anpassung der OCM-Struktur durchgeführt werden. Hier ist insbesondere die Umschaltung von Reglerstrukturen des Controllers bei Zustandsänderungen des Reflektorischen Operators zu nennen. Für diesen Fall wurde in [GBSO04, GHH+06] gezeigt, wie die Kombination aus Reflektorischem Operator und Controller modelliert werden muss, um nachfolgend einen mathematischen Beweis der Sicherheit durchzuführen.

3.2 Monitoring-Konzept

Um die Sicherheit eines selbstoptimierenden Systems in jeder Betriebsphase zu gewährleisten, wurde zur Überwachung ein Monitoring-Konzept entwickelt, das im Reflektorischen Operator des OCMs integriert ist. Der Systemzustand wird dabei einem von vier Sicherheitsbereichen zugeordnet, um vor Fehlern und Gefahren zu schützen. Diese können sowohl durch das klassische System und dessen Umfeld entstehen, als auch durch die Anwendung der Selbstoptimierung selbst. Das Monitoring-Konzept entscheidet, wann und in welchem Umfang der Einsatz von Selbstoptimierung vernünftig ist und in wie weit Ziele der Sicherheit in der Phase „Bestimmung der Systemziele“ priorisiert werden müssen, so dass das System die notwendigen Maßnahmen treffen kann. Im Detail definieren sich die Bereiche gemäß Abbildung 2.

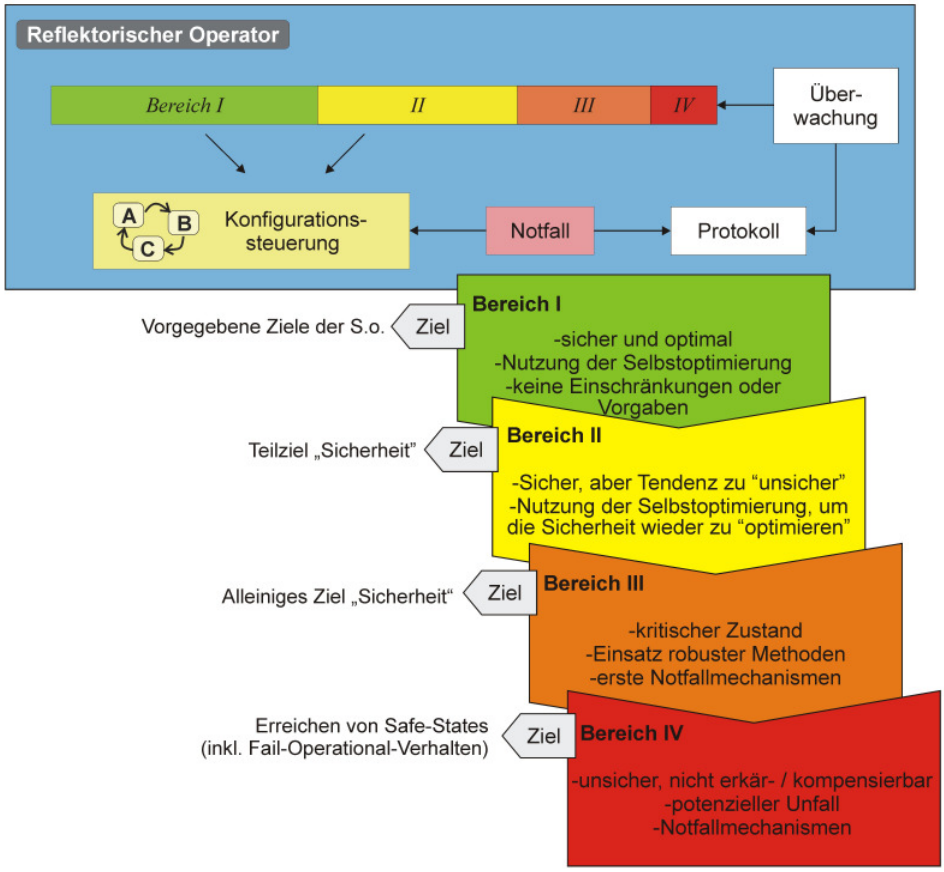

Abbildung 2: Monitoring-Konzept für selbstoptimierende Systeme

3.3 Flexibles Ressourcenmanagement

Um die Systemressourcen, wie z. B. Speicher, CPU-Zeit und FPGA-Fläche, selbstoptimierenden Anwendungen besser zur Verfügung zu stellen, wurde ein Flexibler Ressourcenmanager (FRM) im Rahmen des SFBs entwickelt [OB04]. Dieses war nötig, um dem dynamischen Ressourcenbedarf von selbstoptimierenden Anwendungen gerecht zu werden, ohne Worst-Case-Ressourcen für alle Anwendungen zurückzuhalten. Die Ressourcen zurückzuhalten, würde zu einem großen internen Verschnitt der Ressourcen führen. Anwendungen können verschiedene Implementierungsalternativen – so genannte Profile – mit unterschiedlichem Ressourcenbedarf spezifizieren. Der FRM versucht dann zur Laufzeit unter Einhaltung von Echtzeitbedingungen aktuell freie Ressourcen anderen Anwendungen zur Verfügung zu stellen indem er ein Profil mit höherem Ressourcenbedarf aktiviert.

Um für selbstoptimierende Verfahren zusätzliche Ressourcen bereitstellen zu können, wurde in [GMM+06] ein Verfahren vorgestellt, welches den Prozessen im menschlichen Körper unter Stress nachempfunden ist. In bedrohlichen Situationen stellt unser Körper dezentral und ohne bewusstes Handeln das Verhalten unserer Organe so um, dass uns mehr Kraft zur Verfügung steht, sich der Gefahr zu stellen oder schnell von ihr wegzulaufen (Fight-or-Flight-Syndrom). Erzeugt wird dieses Verhalten durch die Ausschüttung von Adrenalin. Dieses Verhalten versuchen wir in unserem Ressourcenmanagement nachzubilden. Erkennt eine Komponente eine Gefahr, schüttet sie virtuelles Adrenalin aus, welches den anderen Komponenten signalisiert wird. Das Adrenalin trägt Informationen darüber, wie viele zusätzliche Ressourcen benötigt werden, um der Gefahr mit zusätzlicher Selbstoptimierung entgegen zu wirken und um was für eine Gefahr es sich handelt. Andere Komponenten entscheiden dann anhand dieser Information, ob sie Ressourcen abgeben oder ob sie von dieser Gefahr ebenfalls betroffen sind und Gegenmaßnahmen einleiten. Dieses Verfahren erlaubt die Ressourcen des Systems im Notfall effizienter zu nutzen und beispielsweise Komponenten, die nur für den Komfort zuständig sind, zu deaktivieren.

4 Potenziale für die Verlässlichkeit

Die kontinuierliche Analyse des Systemzustands und der Umfeldbedingungen sowie die Möglichkeit, entsprechend auf Änderungen reagieren zu können, ermöglichen einem selbstoptimierenden System, auch solche Ziele zu berücksichtigen, die die Verlässlichkeit erhöhen. Dies kommt besonders bei den in Abschnitt 1.2 dargestellten Attributen der Verfügbarkeit und Zuverlässigkeit zum Tragen. Mit Hilfe selbstoptimierender Methoden lässt sich einerseits das Auftreten von Fehlerfällen verhindern, indem potenzielle Fehler vorab erkannt werden und das Systemverhalten entsprechend angepasst wird. Andererseits kann ein solches System auch im Fall eines auftretenden Fehlers reagieren, um je nach Art des Fehlers die Verfügbarkeit weiterhin zu gewährleisten. Im Folgenden sollen einige Ansätze verdeutlichen, in welcher Hinsicht die Selbstoptimierung diese Potenziale für die Verlässlichkeit bietet.

4.1 Selbstoptimierende Wartungsplanung

Während früher feste Zeitintervalle zwischen den Wartungszeitpunkten lagen, können heute bereits durch „Just in Time Wartung“ erhebliche Ressourcen eingespart werden. Durch Nutzung der Selbstoptimierung kann dieses noch übertroffen werden: Das selbstoptimierende System kann Wartungsintervalle flexibel planen und selbständig konkrete Maßnahmen zur Wartung einleiten, die auf im laufenden Betrieb gewonnenen Messwerten basieren. Zusätzlich können sowohl zur War-

tung notwendige Ressourcen besser geplant werden, evtl. bestehende Zuverlässigkeits- und Sicherheitsgefahren identifiziert werden, als auch das Wissen über die Verfügbarkeit bzw. der Nicht-Verfügbarkeit anderweitig, z. B. im Rahmen von Flottenmanagementsystemen, genutzt werden. Eine selbstoptimierende Wartungsplanung aufgrund von im laufenden Betrieb gemessenen oder sogar für die Zukunft vorhergesagten Betriebsbelastungen beinhaltet somit großes Potenzial für die Aspekte Sicherheit, Zuverlässigkeit und Verfügbarkeit.

4.2 Verfügbarkeitssteigerung durch Verhaltensanpassungen

Durch den Einsatz von selbstoptimierenden Verfahren lassen sich jedoch nicht nur die Wartungsintervalle individuell optimieren, sondern oft auch im Fall eines bereits eingetretenen Wartungsbedarfs durch Verhaltensanpassung eine weiterhin sichere und möglichst wenig eingeschränkte Funktionalität realisieren und somit die Verfügbarkeit steigern.

Als Beispiel sei hier ein Funktionsmodul genannt, welches durch mechanischen Verschleiß ein übermäßiges mechanisches Spiel aufweist. Dies wird durch das Sicherheitsmonitoring erfasst und dem betroffenen sowie den hiermit agierenden Funktionsmodulen propagiert. Das betroffene Modul kann dann als Reaktion sein Zielsystem dahingehend anpassen, dass es seine Aktivität auf ein Mindestmaß einschränkt und somit weiteren Verschleiß vermindert. Die weiteren Module, die mit dem verschlissenen Modul interagieren, können ebenfalls ihre Aktionen derart ändern, dass sie zum einen – wenn möglich – die eingeschränkte Funktionalität des geschädigten Moduls kompensieren oder übernehmen, und zum anderen ihre Aktivitäten so koordinieren, dass das Modul weniger stark angeregt wird. Auf diese Weise kann das geschädigte Modul geschont werden und ein schnelles Fortschreiten des Verschleißes und damit ein Totalausfall vermieden werden, so dass der Betrieb bis zur nächsten Wartung aufrecht erhalten werden kann.

Auch im Bereich des Energiemanagements eines Fahrzeugs finden sich Beispiele für die Verfügbarkeitssteigerung durch Verhaltensanpassungen. Im Kraftfahrzeug übersteigt heutzutage die summierte Leistungsaufnahme aller Verbraucher die Leistung des Generators um das drei- bis fünffache. Um die Versorgung sicherheitskritischer Verbraucher wie z. B. elektrischer Servolenkung oder Beleuchtung sicher zu stellen, werden bei Überlast im Bordnetz vom Bordnetzsteuergerät klassischerweise Komfortfunktionen nach einer bereits zur Implementierung festgelegten Reihenfolge abgeschaltet und stehen damit nicht mehr zur Verfügung. Problematisch ist zusätzlich, dass das zentrale Bordnetzsteuergerät einen „Single Point of Failure“ bildet, so dass es für einen sicheren Betrieb redundant ausgelegt werden muss [HWH04].

Durch den Einsatz selbstoptimierender Methoden und eines agentenbasierten Ansatzes kann eine fehlertolerante Lösung implementiert werden, bei der in den einzelnen Funktionsmodulen die Funktion „Energieverbrauch senken“ im Zielsystem implementiert ist. Je nach Energiestand im Bordnetz sowie der Sicherheits- und Funktionsrelevanz des jeweiligen Moduls kann zwischen den Modulen der Energiebedarf ausgehandelt und der verfügbaren Energie angepasst werden. Im Gegensatz zum zentralen Bordnetzsteuergerät gibt es durch den verteilt bestimmten und ausgehandelten Energiebedarf keinen „Single Point of Failure“ mehr. Außerdem werden die einzelnen Funktionsmodule nicht mehr nach einem festen Schema abgeschaltet, sondern bleiben mit einer der Situation angepassten Grundfunktionalität erhalten.

4.3 Selbstheilung

Die Informationsverarbeitung mechatronischer Systeme besteht aus ECUs und darauf ausgeführter Software. Ausfälle der ECUs beeinträchtigen daher die Verfügbarkeit und Zuverlässigkeit eines mechatronischen Systems. In [TGSP05] wird ein Ansatz zur Selbstheilung des Systems durch Umverteilung der durch den Ausfall einer ECU betroffenen Software vorgestellt.

Nach Ausfall einer ECU wird die Umverteilung der betroffenen Software als neues Ziel in das Zielsystem hinzugefügt. Es wird zwischen zwei unterschiedlichen Zielen unterschieden. Das erste Ziel ist eine möglichst schnelle Selbstheilung, die keine optimale Verteilung der Software auf die verbleibenden ECUs hinsichtlich Fehlertoleranz und Redundanz betrachtet. Wenn dieses erste Ziel erreicht wurde, wird das Ziel durch das langfristige Ziel ersetzt, eine optimale Verteilung hinsichtlich Fehlertoleranz und Redundanz zu berechnen und umzusetzen.

5 Fazit

Mit den skizzierten Ansätzen lässt sich die Verlässlichkeit auch in selbstoptimierenden mechatronischen Systemen – trotz der erhöhten Komplexität der Informationsverarbeitung, die für die gestiegenen Anforderungen benötigt wird – gewährleisten. Darüber hinaus bietet das Paradigma der Selbstoptimierung Chancen zur Steigerung der Verlässlichkeit, die mit konventionellen Vorgehensweisen so nicht genutzt werden können.

Literatur

[BSDB00] Bradley, D.; Seward, D.; Dawson, D.; Burge, S.: Mechatronics. Stanley Thornes, 2000. – ISBN 0–7487–5443–1

[FGK+04] Frank, U.; Giese, H.; Klein, F.; Oberschelp, O.; Schmidt, A.; Schulz, B.; Vöcking, H.; Witting, K.; Gausemeier, J. (Hrsg.): Selbstoptimierende Systeme des Maschinenbaus Definitionen und Konzepte. HNI-Verlagsschriftenreihe, Band 155, Paderborn, Deutschland, 2004

[GBSO04] Giese, H.; Burmester, S.; Schäfer, W.; Oberschelp, O.: Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration. In: Proc. of 12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004), Newport Beach, USA, ACM Press, November 2004, S. 179–188

[GHH+06] Giese, H.; Henkler, S.; Hirsch, M.; Tichy, M.; Vöcking, H.: Modellbasierte Entwicklung vernetzter, mechatronischer Systeme am Beispiel der Konvoifahrt autonom agierender Schienenfahrzeuge. In: Proc. Of the Fourth Paderborner Workshop Entwurf mechatronischer Systeme, Bd. 189, 2006, S. 457–473

[GMM+06] Giese, H.; Montealegre, N.; Müller, T.; Oberthür, S.; Schulz, B.: Acute stress response for self-optimizing mechatronic systems. In: IFIP Conference on Biologically Inspired Cooperative Computing, 2006

[GTB+03] Giese, Holger; Tichy, M.; Burmester, S.; Schäfer, W.; Flake, S.: Towards the Compositional Verification of Real-Time UML Designs. In: Proc. Of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-11), ACM Press, September 2003, S. 38–47

[HWH04] Heintel, B.; Wagner, T.; Hohn, G.: Agentenorientiertes dezentrales Energie- und Funktionsmanagement für Kfz, Entwicklerforum Kfz-Elektronik, Ludwigsburg/Stuttgart, Mai 2004

[KP02] Kochs, H.-D.; Peterson, J.: Verlässlichkeit mechatronischer Systeme / Mitteilungen der Fachgruppe Fehlertolerierende Rechensysteme der Gesellschaft für Informatik. 2002 (30). – Forschungsbericht

[Lap92] Laprie, J. C. (Hrsg.): Dependable computing and fault tolerant systems. Bd. 5: Dependability: basic concepts and terminology in English, French, German, Italian and Japanese [IFIP WG 10.4, Dependable Computing and Fault Tolerance]. Wien, Springer Verlag, 1992. – ISBN 3–211–82296–8 or 0–387–82296–8

[MGPK99] Musliner, D. J.; Goldman, R. P.; Pelican, M. J.; Krebsbach, K. D.: Self-Adaptive Software for Hard Real-Time Environments. In: IEEE Inteligent Systems 14 (1999), Juli/August, Nr. 4

[OB04] Oberthür, S.; Böke, C.: Flexible Resource Management – A framework for self-optimizing real-time systems. In: KLEINJOHANN, Bernd (Hrsg.); GAO, Guang R. (Hrsg.); KOPETZ, Hermann (Hrsg.); KLEINJOHANN, Lisa (Hrsg.); RETTBERG, Achim (Hrsg.): Proceedings of IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES'04), Kluwer Academic Publishers, 23 - 26 August 2004

[OGT+99] Oreizy, P.; Gorlick, M. M.; Taylor, R. N.; Heimbigner, D.; Johnson, G.; Medvidovic, N.; Quilici, A.; Rosenblum, D. S.; Wolf, A. L.: An Architecture-Based Approach to Self-Adaptive Software. In: IEEE Intelligent Systems 14 (1999), May/June, Nr. 3, S. 54–62

[OHG04] Oberschelp, O.; Hestermeyer, T.; Giese, H.: Strukturierte Informationsverarbeitung für selbstoptimierende mechatronische Systeme. In: Proc. of the Second Paderborner Workshop Intelligente Mechatronische Systeme, Bd. 145, Paderborn, Germany, 2004, S. 43–56

[SKB98] Sztipanovits, J.; Karsai, G.; Bapty, T.: Selfadaptive software for signal processing. In: Commun. ACM 41 (1998), Nr. 5, S. 66–73

[TGSP05] Tichy, M.; Giese, H.; Schilling, D.; Pauls, W.: Computing Optimal Self-Repair Actions: Damage Minimization versus Repair Time. In: DE LEMOS, Rog´erio (Hrsg.); ROMANOVSKY, Alexander (Hrsg.): Proc. of the ICSE 2005 Workshop on Architecting Dependable Systems, St.Louis, Missouri, USA, ACM Press, May 2005, S. 1–6