Fehlertoleranz-Techniken automatisiert einbetten

Die ISO 26262 unterscheidet zwischen systematischen und zufälligen Fehlern. Mit Hilfe der Laufzeitumgebung KESO lässt sich hard- und softwarebasierter Speicherschutz anwenden.

ISABELLA STILKERICH UND BERNHARD SECHSER*

Das Thema „Funktionale Sicherheit" von Embedded-Systemen ist aktueller denn je. Es geht darum, das Risiko eines Fehlers zu minimieren, der von einem elektrischen, elektronischen oder programmierbar elektronischen System ausgeht und sich gefährlich auswirken könnte. Die Funktionale Sicherheit eines eingebetteten Systems, bestehend aus Software und Hardware, beruht einerseits auf der korrekten Implementierung der Software-Module und andererseits auf dem fehlerfreien Verhalten der eingesetzten Hardware-Bausteine. Um Funktionale Sicherheit zu standardisieren, wurden diverse Normen entwickelt. Neben der „Ur-Norm" IEC 61508 gibt es in fast allen Bereichen spezifische Derivate. Für die Automobilbranche wurde im November vergangenen Jahres die Norm ISO 26262 verabschiedet. Sie gibt unter anderem Richtlinien zur Entwicklung von Software, Hardware sowie ganzer Systeme vor.

Geleitschutz: *Mit Hilfe von software-basierten Techniken lassen sich die Auswirkungen von transienten Fehlern abmildern*

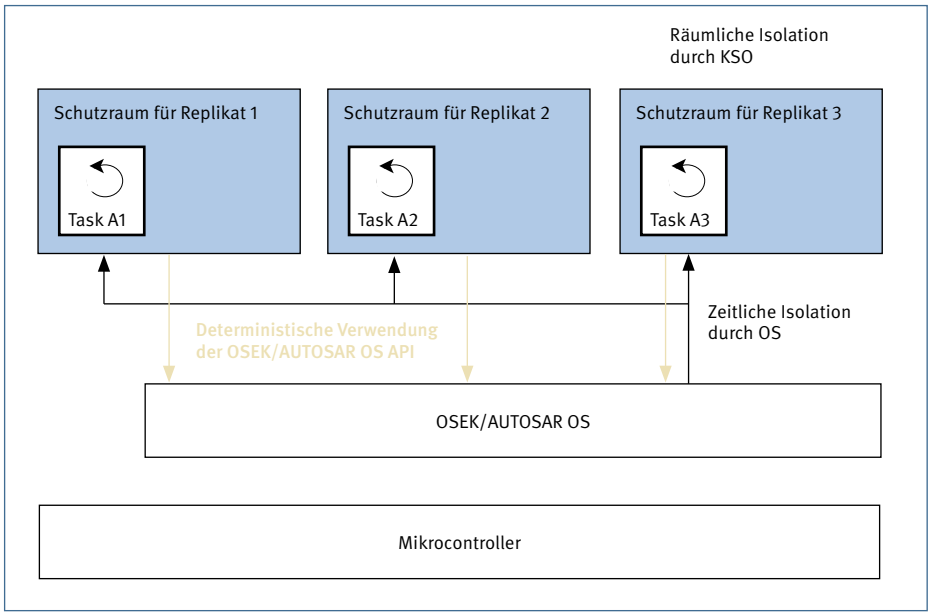
Systematische und zufällige Fehler nach ISO 26262

Der Standard unterscheidet sogenannte systematische Fehler und zufällige Fehler. Systematische Fehler befinden sich von Anfang an im System, während sich zufällige Fehler meist erst nach Auslieferung des Systems manifestieren. Sowohl Hardware als auch Software beinhalten unter Umständen systematische Fehler, wohingegen sich zufällige Fehler - permanenter oder transienter Natur - per Definition nur in der Hardware befinden. Transiente Hardware-Fehler entstehen in der Regel durch Umwelteinflüsse – wie etwa Temperatur oder Strahlung -, eine zu niedrige Stromversorgung oder durch den Trend zur Strukturverkleinerung in modernen Mikrocontrollern. Transiente Fehler machen sich als Bitflips im Random Access Memory (RAM), in Bussen oder in Registern der Central Processing Unit (CPU) bemerkbar. Dadurch kann die Ausführung eines Software-Moduls gestört werden und möglicherweise sogar zum Absturz des kompletten Systems führen. Hardware-basierte Ansätze wie physische Replizierung von Komponenten oder der Einsatz von Error-Correcting-Code-geschütztem Speicher (ECC) stellen eine mögliche Lösung dar. Diese Maßnahmen sind jedoch in vielen Anwendungsfeldern, wie beispielsweise der Automobilindustrie, aufgrund des immensen Kostendrucks oftmals schwer umzusetzen. Darüber hinaus ist Hardware-Redundanz in Systemen, bei denen die physische Größe, Gewicht und Energiebeschränkungen eine Rolle spielen, technisch nicht machbar. Im Vergleich dazu erscheinen software-basierte Fehlertoleranz-Techniken hier besonders attraktiv.

Software-basierte Techniken mildern die Auswirkungen transienter Fehler nämlich ebenfalls ab, was verschiedenste Projekte bereits gezeigt haben. Diese Techniken werden häufig jedoch manuell umgesetzt, beispielsweise mit dem Umzug einer Applikation auf eine neue Hardware-Plattform oder der Anpassung der Safety-Beschreibung muss die Fehlertoleranzmaßnahme in der Anwendung auf die neuen Gegebenheiten angepasst werden. Am Lehrstuhl 4 für Informatik an der Universität Erlangen-Nürnberg wird zur automatischen Anwendung solcher Techniken ein Forschungswerkzeug entwickelt: KESO. KESO [1] steht für „Konstruktiver

*** Isabella Stilkerich**
... arbeitet an der Uni Erlangen und promoviert zum Thema Fehlertoleranz-Techniken in sicherheitskritischen Embedded-Systemen.

Bernhard Sechser
... ist Principal Consultant bei Method Park Software und berät zu den Themen SPICE & Safety.



Replizierung auf der KESO-Plattform: *Auf der Plattform werden Java-Anwendungen flexibel konfigurierbaren Schutzräumen zugeordnet#####*

Eingebetteter Speicherschutz für OSEK“ und ist eine Multi-Java Virtual Machine (JVM) für Deeply-Embedded-Systeme. KESO ermöglicht eine flexible Zuordnung von Java-Anwendungen zu konfigurierbaren Schutzräumen.

Java-Anwendungen Schutzräumen zuordnen

Durch die logische Trennung der Speicherbereiche der Applikationen und der Typsicherheit der Programmiersprache werden die Applikationen konstruktiv voneinander isoliert; eine Memory Protection Unit (MPU) wird dazu nicht benötigt, der MPU-Einsatz kann jedoch mit KESO kombiniert werden. Bei KESO handelt es sich um eine Laufzeitumgebung, die sich auf AUTOSAR OS aufsetzen lässt. Das Transformationswerkzeug übersetzt ein in Java geschriebenes Anwendungsprogramm in speichersicheren C-Code, der problemlos mit existierenden, in C, C++ oder Assembler geschriebenen Software-Modulen integriert werden kann. Damit ist es möglich, auf ein System - bestehend aus KESO-Komponenten und Altanwendungen - eine Kombination von hardware-basiertem und software-basiertem Speicherschutz anzuwenden. Der KESO-Compiler analysiert eine Applikation und erzeugt eine maßgeschneiderte Laufzeitumgebung. Der generierte Code kann dabei hinsichtlich Speicherverbrauch und Laufzeit mit traditionell in C/C++ entwickelten Komponenten mithalten [2].

Mit KESO können einige Forderungen an die Implementierung von Software-Modulen abgehandelt werden. Aufgrund der Typsicherheit der Anwendungen und der Analyse durch das KESO-Werkzeug lässt sich eine Reihe systematischer Programmierfehler ausschließen. In Kombination mit einer statischen Analyse ermöglicht die Typsicherheitseigenschaft einer statischen Anwendung darüber hinaus die automatische Anwendung von Fehlertoleranz-Techniken zur Abmilderung der Auswirkung transienter Fehler. Beispiele für solche Techniken sind Replikationen, Checksummen-Überprüfungen oder Kontrollfluss-Überwachungen, die per Konfiguration eingestellt werden. Eine Änderung der Implementierung der Applikation bei Umstellung der Fehlertoleranz-Technik oder Anpassung der Anzahl der zu tolerierenden Fehler ist nicht notwendig.

Fehlertolerante Replikation kritischer Berechnungen

Eine Replikation von sicherheitskritischen Berechnungen durch ein Generierungswerkzeug automatisch anzuwenden und anschließend in einer typsicheren, komponenten-basierten Laufzeitumgebung zur Ausführung zu bringen, bietet gegenüber der manuellen Implementierung einige Vorteile. Zum einen befinden sich die Replikate in den Schutzräumen der KESO-Laufzeitumgebung, so dass sich transiente Fehler, die ein Replikat treffen, räumlich nicht auf andere Teile des Systems auswirken können. Zum anderen ist die Anzahl der Replikate per Konfiguration einstellbar, so dass flexibel auf Änderungen der funktionalen Sicherheitsanforderungen reagiert werden kann.

Keso

KESO is in Central Java. The nearest places to Keso are Desa Tlogotunggal, Tlogotunggal, Randu, Kanung (500 meters west), and Ngebrak (700 meters 2outh).

Keine Weltreise: *Mit dem Werkzeug KESO lassen sich systematische Programmierfehler ausschließen*

Das KESO-Übersetzungswerkzeug berechnet die applikationsspezifischen Fehlererkennungsroutinen und den spezifizierten sicheren Zustand einer statischen Anwendung. Wird ein Fehler erkannt, können diese Zustände automatisch eingenommen werden. Ein solcher Fehler kann nicht nur durch die KESO-Laufzeitumgebung signalisiert werden, sondern auch durch AUTOSAR OS. Das Betriebssystem kann beispielsweise feststellen, dass sich eine Deadline für einen replizierten Task nicht einhalten lässt, die zeitliche Isolation der Replikate also verletzt ist. Das Betriebssystem kann dann eine durch KESO generierte Zustandswiederherstellung, die in der Worst Case Execution Time Analyse (WCET) berücksichtigt werden muss, anstoßen. Darüber hinaus zeigt eine statische Analyse der Konfiguration und des Anwendungscodes an, ob bestimmte Richtlinien beachtet wurden. So kann im Falle von Replikation beispielsweise geprüft werden, ob der Replikatdeterminismus eingehalten wird. Werden in den Replikaten der Anwendung Systemdienste von AUTOSAR, Funktionen anderer C-Anwendungen oder Dienste von Java-Anwendungen in anderen Schutzräumen verwendet, dann ist sichergestellt, dass sich die Replikate identisch verhalten, falls diese Eigenschaft benötigt wird.

Der software-basierte Speicherschutz, der durch die KESO-Laufzeitumgebung angeboten wird, ist mit dem Einsatz einer MPU kombinierbar. Auf diesem Weg lassen sich die Vorteile beider Techniken ausnutzen. Beispielsweise kann eine MPU transiente Fehler in Referenzen der Applikation teilweise erkennen, falls die betroffene Referenz auf einen ungültigen Speicherbereich zeigt. Auf vielen kostengünstigen Mikrocontrollern ist jedoch keine MPU vorhanden. Unter Umständen möchte man auch die Korrektheit der Referenzen auf der Granularität von Objekten und nicht nur auf Speicherbereichen der MPU gewährleisten. In beiden Fällen bietet die generierte KESO-Laufzeitumgebung hier ebenfalls Unterstützung an. Das Sicherstellen der Referenzintegrität auf Objektebene erlaubt es, Fehlertoleranz-Techniken auch auf feingranularer Ebene einzusetzen. So ist es etwa möglich, nur ausgewählte Daten zu schützen, falls eine komplette Absicherung der Anwendung zu kostenaufwändig sein sollte.

Ist keine MPU vorhanden, stellt die JVM sicher, dass Referenzen sowohl korrekt in Bezug auf die jeweiligen Speicheradressen als auch der Typinformation sind. Die Typinformation bestimmt zum Beispiel, welche Operationen an einem Objekt ausgeführt werden können. Das Transformationswerkzeug berechnet, welche Referenzprüfungen überhaupt durch die Laufzeitumgebung durchgeführt werden müssen und welche statisch evaluiert werden können, ohne die Isolationseigenschaft zu gefährden. Damit will man den Mehraufwand, der durch die Anwendung der Fehlertoleranz-Technik entsteht, möglichst gering halten. // **HEH**

Method Park +49 (0)9131 97206

PRAXIS WERT

Was die ISO 26262 fordert und KESO bietet

Automatisierte Replikation und kombinierter Software-Hardware-Speicherschutz sind nur zwei Beispiele von Fehlertoleranz-Techniken, die sich durch KESO realisieren lassen. Normen wie die ISO 26262 fordern noch eine hohe Anzahl weiterer Techniken, um sowohl Fehler bei der Umsetzung von Software-Komponenten als auch zufällig auftretende Fehler in den verwendeten Hardware-Bausteinen zu diagnostizieren. KESO bietet in vielen Fällen auch hier Unterstützung, um den manuellen, weiterhin fehleranfälligen Aufwand zu minimieren und die Funktionale Sicherheit zu erhöhen. KESO ist frei verfügbar und steht unter der GNU Lesser General Public License [3].

Literatur und Links

[1] *KESO: A Multi-JVM for Deeply Embedded Systems*
[2] *Tailor-made JVMs for statically configured embedded systems*
[3] *Wikipedia: Das Betriebssystem GNU*

InfoClick

- KESO: A Multi-JVM for Deeply Embedded Systems
- Tailor-made JVMs for statically configured embedded systems
- Wikipedia: Das Betriebssystem GNU

www.elektronikpraxis.de **InfoClick** 3641893