Zuverlässigkeitsmechanismen für Eingebettete Systeme

Falk Salewski, Stefan Kowalewski
{salewski, kowalewski}@informatik.rwth-aachen.de

***Abstract*—Traditionelle Methoden zur Erreichung von Zuverlässigkeit haben ihren Ursprung häufig in hardware-dominierten Systemen. Software-intensive eingebettete Systeme erfordern jedoch neue Ansätze wie sie zum Beispiel von der Internationalen Norm IEC61508 vorgeschlagen werden. In diesem Zusammenhang fehlt es jedoch häufig, gerade im Zusammenhang mit Software, an den nötigen fundierten Ergebnissen. Unser Ziel ist es daher, Ansätze zu entwickeln, um mit empirischen Methoden systematisch die nötigen Daten zu sammeln und zu bewerten. In weiteren Schritten gilt es, diese Ergebnisse in entsprechende Regeln und Vorgaben umzusetzen und den Entwicklern zuverlässigkeitskritischer Systeme zur Verfügung zu stellen. Dieser Artikel gibt eine kurze Einführung in die Problematik und stellt Lösungsansätze vor. Details sind [1] zu entnehmen.**

I. EINFÜHRUNG

Eingebettete Software übernimmt immer mehr Funktionen in sicherheitskritischen Systemen, so dass jetzt auch Software-Fehler zu ernsten Unfällen führen können. Die Zuverlässigkeit der Systeme spielt daher in diesem Zusammenhang eine erhebliche Rolle. Aber auch in nicht sicherheitskritischen Anwendungen wird häufig eine hohe Zuverlässigkeit gefordert. Die Zuverlässigkeit eines Systems setzt sich aus der Zuverlässigkeit seiner Komponenten zusammen, die sowohl aus Hardware wie aus Software bestehen können. Unter Hardware verstehen wir in diesem Zusammenhang alle Formen von programmierbaren und nicht programmierbaren elektronischen Bauteilen. Unter Software werden in diesem Zusammenhang alle Sprachen verstanden, die die Funktionalität von Hardware definieren. Diese umfassen neben klassischen Programmiersprachen (Assembler, C, ADA, etc.) auch Hardwarebeschreibungssprachen wie VHDL und Verilog.

Im Bereich der Zuverlässigkeitsanalyse von Hardware kann man auf jahrelange Erfahrungen zurückgreifen. Es gibt erprobte Mechanismen um die Zuverlässigkeitseigenschaften von Hardware-Systemen aus den Eigenschaften ihrer Komponenten oder anhand statistischer Daten vergleichbarer Systeme zu bestimmen. In besonders kritischen Systemen bietet sich zudem die Möglichkeit, die Zuverlässigkeit durch eine Überdimensionierung der Komponenten oder redundante Baugruppen zu erhöhen.

Die Zuverlässigkeitsanalyse von Software gestaltet sich jedoch schwieriger. Die Überdimensionierung von Software-Komponenten ist nicht unmittelbar möglich und auch die Möglichkeiten der Erstellung von unabhängigen redundanten Software-Modulen sind beschränkt.

Da Hardware und Software in vielen Fällen erst gemeinsam eine Funktionalität realisieren, wird eine besondere Notwendigkeit in der Analyse der Wechselwirkungen zwischen Hardware und Software gesehen.

II. SOFTWARE-ZUVERLÄSSIGKEIT

Viele Ingenieure sehen in der Entwicklung verlässlicher Software den wichtigsten Teil beim Entwurf sicherheitskritischer Systeme [5]. Dies hat verschiedene Ursachen. Zum einen ist Software offenkundig schwer zu testen [5], da in der Regel nicht alle möglichen Situationen getestet werden können. Im Bereich eingebetteter Systeme müssen zudem häufig Echtzeitbedingungen eingehalten werden deren Überprüfung das Testen weiter erschwert. Zum anderen ist es, im Gegensatz zu Hardware, nicht möglich redundante, identische Software-Module anzulegen, um die Zuverlässigkeit zu erhöhen, da ein Fehler in einem der Module auch in allen redundanten Einheiten vorhanden wäre. Aufgrund dieser Bedeutung liegt es nahe, Maßnahmen zu entwickeln um bereits auf konstruktivem Weg die Anzahl der Fehler in Software gering zu halten. Für nicht vermeidbare Fehler gilt es, deren Konsequenzen auf ein tolerierbares Maß zu beschränken.

Im Weiteren schlagen wir eine Klassifizierung in sechs Gruppen von Software-Fehlern vor: Fehler auf Grund

1. einer Fehlinterpretation der Spezifikation,
2. eines falschen Verständnisses der zugrunde liegenden Hardware,
3. unentdeckter Seiteneffekte,
4. einer Fehlinterpretation der Semantik der Programmiersprache,
5. von Tippfehlern und
6. unentdeckter Effekte durch verwendete Tools und Compiler.

Nach [5] gibt es folgende vier Gruppen von Techniken um den Auswirkungen dieser Fehler zu begegnen: Fehlervermeidung, Fehlerbeseitigung, Fehlererkennung und Fehlertoleranz. In [1] wird näher auf die einzelnen Punkte eingegangen und gezeigt, dass bei vielen dieser Techniken Wechselwirkungen zwischen Hardware und Software zu beachten sind.

III. EMPIRISCHE STUDIEN

Nach [4] gibt es, gemessen an ihrer Wichtigkeit, relativ wenige veröffentlichte empirische Studien, die sich mit der Verlässlichkeit von Software beschäftigen. Auf Grund der Schwierigkeiten der Durchführung von empirischen Studien in der Industrie (Datenschutz, Scheu vor dem Mehraufwand) schlagen wir vor, Studien im Rahmen von Lehrveranstaltungen (Praktika) an Universitäten durchzuführen. Der Mehraufwand einer strukturierten Datenerhebung im Rahmen eines Praktikums steht unserer Meinung nach in einem guten Verhältnis zu dem Nutzen der Daten für spätere Analysen. Im Folgenden stellen wir exemplarisch ein an unserem Institut durchgeführtes Experiment vor.

Empirische Studien im Bereich der Software-Zuverlässigkeit sind nicht nur relativ selten, sondern sie beschäftigen sich in der Regel auch ausschließlich mit der Software (z.B.: [2], [3], [4]). Den Wechselwirkungen zwischen Hardware und Software wird in diesem Zusammenhang wenig Beachtung geschenkt. Wie aus unseren Ausführungen zum Umgang mit Software-Fehlern hervor geht, kann diese Wechselwirkung durchaus Auswirkungen auf die Zuverlässigkeit des Gesamtsystems haben. Unsere Hypothese lautet daher: Die Implementation einer Funktion mit unabhängigen Designteams auf verschiedenen Hardware-Plattformen führt zu Fehlern mit größerer Unabhängigkeit als die Implementation einer Funktion mit unabhängigen Designteams auf der gleichen Hardware. Um diese Hypothese zu überprüfen sind entsprechende Studien nötig. Eine erste haben wir bereits im Rahmen eines Praktikums durchgeführt.

Im Rahmen dieses Praktikums sollte die Aufgabe (Messen und Versenden von Geschwindigkeitssignalen über einen CAN-Bus) zum einen mit einem Mikrocontroller (MCU) und zum anderen mit einem Complex Programmable Logic Device (CPLD) gelöst werden. Vor dem eigentlichen Praktikum wurde eine zweitägige Einführungsveranstaltung durchgeführt in der die nötigen Grundlagen für beide Hardware-Plattformen vermittelt wurden. Um die Fehler zu dokumentieren, die während der Entwicklung gemacht werden, wurden bei jedem Compileraufruf die aktuelle Version des Quellcodes und die Compilermeldungen gespeichert. Abschließend musste jede Version einen Abnahmetest bestehen.

Für die Evaluation sind mehrere Techniken geplant. In einem ersten entscheidenden Schritt sollen Fehler durch Testen gefunden werden. Um eine hohe Anzahl von Testläufen durchführen zu können, wurde eine automatische Testumgebung entwickelt. Es wurden bereits einige Testläufe durchgeführt und analysiert. Der Vergleich der Ergebnisse von MCU und CPLD Implementierungen machte einige Unterschiede deutlich. Die Anzahl der fehlerhaft gesendeten Nachrichten war bei den CPLD Version im Schnitt höher als bei den MCU Versionen. Im Gegenzug war der Anteil der verspäteten und verlorenen Nachrichten bei den MCU Versionen höher. Neben dieser absoluten Auswertung der Fehler ist im Weiteren zu untersuchen wie unabhängig die Fehler zwischen den einzelnen Gruppen und Versionen sind. Diesbezüglich erfolgen augenblicklich weitere umfangreiche Analysen auf Basis der gesammelten Daten.

Der Bedarf an empirischen Studien im Bereich eingebetteter Systeme erstreckt sich unserer Meinung nach auf zahlreiche weitere Gebiete. Neben der hier vorgestellten Studie schlagen wir fundierte Untersuchungen der Auswirkungen von Programmiersprachen, Betriebssystemen und allgemeinen Architektur-Vorgaben auf die Zuverlässigkeit von eingebetteten Systemen vor.

IV. ZUSAMMENFASSUNG UND AUSBLICK

In diesem Artikel werden die Probleme im Bereich der Software-Zuverlässigkeit betont. Als Konsequenz wird auf die Notwendigkeit empirischer Untersuchungen hingewiesen um fundiertes Wissen über die Zuverlässigkeit eingebetteter Systeme, im Besonderen ihrer Software, zu erlangen. Beispielhaft wird eine von den Autoren durchgeführte empirische Studie und erste daraus resultierende Ergebnisse präsentiert.

Für fundierte Aussagen im Bereich der Zuverlässigkeit eingebetteter Systeme sind zahlreiche empirische Studien nötig. Wir planen für die folgenden Semester weitere Praktika, die ebenfalls für empirische Studien genutzt werden sollen. Wir hoffen auch bei anderen Instituten das Interesse an empirischen Studien im Rahmen von Lehrveranstaltungen oder, falls möglich, auch im industriellen Rahmen zu wecken um eine solide Daten-Basis für umfangreiche Untersuchungen im Bereich der Zuverlässigkeits-Mechanismen für eingebettete Systeme zu entwickeln.

V. LITERATUR

[1] F.Salewski and S.Kowalewski, Zuverlässigkeitsmechanismen für Eingebettete Systeme, *In Proceedings of the Workshop on Zuverlässigkeit in eingebetteten Systemen - Ada Deutschland Tagung 2005*

[2] V. R. Basili and B. T. Perricone. Software errors and complexity: An empirical investigation. *Communications of the ACM*, 27:42–52, 1984.

[3] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Trans. Softw. Eng.*, 12:96–109, 1986.

[4] T. J. Ostrand and E. J. Weyuker. The distribution of faults in a large industrial software system. In *ISSTA '02: Proceedingsof the 2002 ACM SIGSOFT*

[5] N. Storey. *Safety-Critical Computer Systems*. Prentice Hall, 1996.