

Softwarequalität in Steuergeräten

Statisch analysiert

Bei der Entwicklung komplexer Steuergeräte ist die Softwarequalität und -stabilität ein zunehmendes Problem. Abhilfe versprechen Techniken der statischen Analyse. Jedoch setzt deren Anwendung oft profundes Fachwissen voraus.

(etwa für Laufzeitfehler), existieren für andere gerade einmal erste Werkzeuge oder Prototypen (beispielsweise für Genauigkeitsabschätzungen bei numerischen Gleitkommaberechnungen) und bei einigen handelt es sich sogar um wirklich forschungsrelevante Bereiche (so bei der Synchronisation von Tasks und Multi-Threaded-Anwendungen).

Ohne Fachkenntnisse geht wenig

Doch auch bei den etablierten Werkzeugen lauern Tücken, und oft ist profundes Domänen- und Werkzeugwissen erforderlich, etwa bei falsch-positiven und falsch-negativen Meldungen. Sie sind eine Folge der Abstraktion. Bei falsch-positiven Meldungen wird ein Fehler vermutet, der im wirklichen Programmablauf nicht auftreten kann, und bei falsch-negativen Meldungen markiert das Tool eine Codezeile als fehlerfrei, obwohl die entsprechende Anweisung in der Ausführung nicht sicher ist. Falsch-positive Meldungen sind ärgerlich, weil sie bei der Analyse unnötigen Zusatzaufwand für die Überprüfung verursachen. Falsch-negative Programmzeilen, also nicht erkannte Programmierfehler, sind besonders für sicherheitskritische eingebettete Systeme unakzeptabel und sind unter allen Umständen zu vermeiden. In diesen Fällen stoßen automatisierte Techniken an ihre Grenzen, und manuelle Methoden sind nötig. Der Funktionsumfang der zur Verfügung stehenden Werkzeuge ist sehr unterschiedlich und deckt verschiedene Anforderungen ab. Für die Aufdeckung von Laufzeitfehlern etwa ist in einigen Programmiersprachen wie Java oder C# eine rudimentäre statische Analyse schon fest vor-

Rolf Hänisch
Prof. Dr.
Holger Schlingloff

Durch die zunehmende Komplexität von Steuergeräten, beispielsweise in der Medizin- und Bahntechnik, stoßen herkömmliche Methoden der Qualitätssicherung von Software oft an ihre Grenzen. Statische Analysewerkzeuge entwickeln sich dagegen immer mehr zur Technik der Wahl. Sie können bei sachgerechtem Einsatz auch in komplexen Systemen eine beträchtliche Menge von Fehlern im Code aufdecken. Zudem haben sie im Vergleich zu dynamischen Verfahren wie Tests den Vorteil, dass die Entwickler mit ihnen ein Programm bereits anhand des Quelltextes untersuchen können, ohne es tatsächlich auf der Zielplattform ausführen zu müssen. Neben manuellen Methoden wie

Code-Review, Fagan-Inspektion und Software-Walkthrough zählen zur statischen Analyse vor allem automatisierte Techniken wie Model-Checking, Shape-Analysis und die abstrakte Interpretation. Insbesondere die abstrakte Interpretation wird im Bereich sicherheitskritischer Systeme immer mehr zum Standardverfahren: Sie kann wichtige Softwareeigenschaften wie Korrektheit und Typsicherheit berechnen und wird mittlerweile in vielen Sicherheitsstandards, wie beispielsweise IEC 615018, verbindlich vorgeschrieben. Allerdings existieren nicht für alle relevanten Softwareeigenschaften etablierte statische Analysewerkzeuge, die für den industriellen Einsatz taugen.

Mit Blick auf die Softwareentwicklung und -qualitätssicherung komplexer Steuergeräte haben sich zum Beispiel für eingebettete Systeme in C oder C++

folgende Eigenschaften und Aspekte als besonders relevant herausgestellt:

- Laufzeitfehler wie Nulldivisionen, Feldgrenzüberschreitungen und allgemeine Speicherschutzalarme bei der De-Referenzierung von Zeigern,
 - fehlende beziehungsweise fehlerhafte Initialisierung von Variablen und Zeigern,
 - Erreichbarkeit oder Nichterreichbarkeit von Zuständen und Codefragmenten,
 - sichere Genauigkeitsabschätzungen für numerische Gleitkommaberechnungen,
 - Schranken für Ausführungszeiten und Stackgrößen sowie
 - korrekte Synchronisation von Tasks und Multi-Threaded-Anwendungen.
- Während es für einige dieser Eigenschaften und Aspekte mittlerweile vollkommen etablierte und gängige statische Analysewerkzeuge gibt

geschrieben, in anderen, wie C und C++, lässt sie sich durch Compiler-Flags einschalten. Allerdings prüfen die Analysewerkzeuge meist nur »einfache« Eigenschaften, beispielsweise die korrekte Initialisierung von Variablen. Weitergehende Werkzeuge, wie das klassische Lint, können auch Typisierungsfehler (Unsafe Type Casts) oder mögliche Pufferüberläufe (Buffer Overflow) erkennen. Spezialisierte kommerzielle Werkzeuge unterstützen sogar allgemeine Vor- und Nachbedingungen, das Einhalten von Kodierrichtlinien (z.B. MISRA), das Entfernen von unerreichbaren Programmteilen (Dead Code) und mehr. Welches der Werkzeuge für den jeweiligen Einsatz letztlich das richtige ist, entscheidet sich nach verfügbarem Know-how, Kosten und der möglichen Einbindung des Werkzeugs in den Programmierungsprozess. Auch hier ist eine fachkundige Einschätzung gefordert.

Sicherheitskritische Systeme

Da viele eingebettete Systeme in sicherheitskritischen Bereichen mit hohem Gefährdungspotenzial für Mensch und Umwelt arbeiten, stellen sie an die Entwicklung und Qualitätssicherung besondere Anforderungen. Auf der einen Seite wird etwa aus Sicherheitsgründen bei vielen eingebetteten Systemen auf spezifische Funktionen wie die Verwendung von dynamischem Speicher verzichtet. Trotzdem kann es bei der Ausführung auf der Zielhardware zu Speicherfehlern durch Stacküberläufe (Stack Overflow) kommen.

Seit kurzem gibt es Werkzeuge, die dieses Problem lösen: Mit einer statischen Analyse des Wertes des Stackzeigers (Stack Pointer)

kann der Entwickler feststellen, wie der Stack entlang der verschiedenen Kontrollflusspfade wächst. Auf diese Weise kann er die erforderliche Stacktiefe des analysierten Programmstücks für eine gegebene Hardware optimieren.

Andererseits bestehen für eingebettete Systeme oft harte Echtzeitanforderungen, die auf keinen Fall verletzt werden dürfen. Um den sicherheitstechnischen Nachweis dafür zu erbringen, gibt es mittlerweile Werkzeuge, die für eine vorgegebene Hardware den Zielcode auf Assemblerebene analysieren und die maximale Ausführungszeit der Prozesse ermitteln. Damit lässt sich gewährleisten, dass das System die Echtzeitbedingungen erfüllt. Da Hardwarekomponenten wie Caches und Pipelines die Ausführungszeiten stark beeinflussen, ist deren Verhalten für das gegebene Programm ebenfalls zu analysieren. Die Konfiguration für eine bestimmte Zielplattform lässt sich dabei anhand einer formalen Spezifikation des Prozessors in VHDL durchführen.

In mehreren Projekten im sicherheitskritischen Medizin- und Bahnbereich wurden die genannten statischen Analysemethoden bereits von Fraunhofer FIRST (Institut für Rechnerarchitektur und Softwaretechnik) eingesetzt. So unterzogen die Forscher für die Qualitätssicherung einer medizinischen Pumpe den gegebenen C-Code einer statischen Analyse, um uninitialisierte Variable, unvollständige Fallunterscheidungen, vertauschte Parameter und ähnliche Fehler zu entdecken. Um weitergehende

Eigenschaften nachzuweisen, zerlegten sie das System dann in Komponenten und hielten die Struktur der Komponenten in UML2-Diagrammen fest (Betriebssystem, Bibliotheken, Kommunikationsschicht, Applikation). Anhand des Codes erstellten sie schließlich weitere Modelle, welche die Interaktion zwischen den einzelnen Komponenten beschreiben. Mit den so entstandenen Modellen konnten sie durch statische Analyse die geforderten Fehlertoleranzeigenschaften und weitere Qualitäten überprüfen. Der Einsatz von statischer Analyse – insbesondere in Verbindung mit anderen Methoden der Qualitätssicherung wie Tests – spart Zeit und steigert die Softwaresicherheit und eignet sich daher besonders für komplexe Systeme wie eine medizinische Pumpe.

In einem laufenden Projekt setzt das Institut die statische Analyse zur Verifikation des Zeitverhaltens einer bahntechnischen Anlage ein. Die Spezifikation der Funktionen ist dabei durch eine grafische Programmiersprache für speicherprogrammierbare Steuerungen (SPS) gegeben. Diese wird zunächst nach C und dann in Maschinensprache übersetzt. Auf C-Ebene lassen sich dabei die Vollständigkeit der Fallunterscheidungen und das Systemstartverhalten analysieren. Wichtigste Echtzeitanforderung ist, dass alle Zweige eines Verarbeitungsweges innerhalb eines Zyklus beendet werden. Die Modellierung der Ausführungszeit einzelner Anweisungen auf der Zielhardware dient der Verifikation dieser Eigenschaft. Auf diese Weise sollen letztlich not-

wendige Nachweise für eine Zulassung im sicherheitskritischen Bereich erbracht werden.

Die Erfahrungen zeigen, dass die statische Analyse heutzutage Softwareprobleme erfassen kann, die sich mit anderen Methoden und vor allem in komplexen Systemen nicht oder nur sehr schlecht finden lassen. Allerdings ist für den Einsatz der Werkzeuge oft eine erhebliche Expertise erforderlich. Die Integration und ingenieurtaugliche Aufbereitung der Methoden für den alltäglichen Einsatz ist daher ein wichtiger nächster Schritt und wird zurzeit von Fraunhofer FIRST auf europäischer Ebene im ITEA-Verbundprojekt ES_PASS (Embedded Software Product-based Assurance) untersucht. (mc)

Rolf Hänisch

ist wissenschaftlicher Mitarbeiter,
Prof. Dr.

Holger Schlingloff

ist Professor für Informatik an der
Humboldt-Universität zu Berlin
sowie stellvertretender Leiter der
Abteilung Eingebettete Systeme
am

**Fraunhofer-Institut
für Rechnerarchitektur
und Softwaretechnik FIRST**
Telefon 030/63 92 18 14
www.first.fraunhofer.de



Halle 12
Stand 138

Embedded Gateway



Das Embedded Modul DNP/6370
macht VoIP-Anwendungen und
Datenübertragung einfach.

embedded world 2008
Exhibition & Conference
Halle 12, Stand 534

SSV ssv-embedded.de

Weitere Informationen

[1] http://www.itea2.org/public/project_leaflets/ES_PASS_profile_oct-07.pdf