Ein Konzept für ein Testframework zum Erstellen, Verwalten und Durchführen automatisierter Regressionstests für variantenreiche Deeply Embedded Systems

Dr. Jörg Nilson
3SOFT GmbH
91058 Erlangen

5. Oktober 2001

Am Beispiel von OSEK/VDX (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug/Vehicle Distributed Exclusive) wird im Vortrag das Problem des Testens von variantenreichen eingebetteten Systemen erläutert. OSEK/VDX ist ein europäischer Standard eines eingebetteten RTOS (Real-Time Operating System). Dieser Standard wird hauptsächlich in der Automobiltechnik, aber auch in der Medizintechnik eingesetzt. Der Standard wurde durch europäische Automobilhersteller, Zulieferer, die 3SOFT GmbH und andere IT Unternehmen entwickelt. OSEK ist ein statisches Betriebssystem, d.h. alle Betriebssystemobjekte und die Mehrzahl ihrere Eigenschaften müssen zur Übersetzungszeit bereits konfiguriert sein und können während der Laufzeit nicht verändert werden. Dies führt zu einer enormen Vielfalt an Konfigurationen, so gibt es beispielsweise 768 verschiedene Möglichkeiten um allein das Betriebssystemobjekt selbst zu definieren. Zusätzlich gibt es jedoch noch Events, Tasks, Counters, Interrupt Service Routinen, Messages, Alarms und möglicherweise auch ein Kommunikationsobjekt. All diese Objekte bieten eine Vielfalt von Konfigurationen, von denen viele zwar ähnlich aber nicht identisch sind.

Solche variantenreichen Systeme erfordern eine besondere Behandlung in Bezug auf Testspezifikation, -implementierung und -durchführung. Darüberhinaus möchte man auch die Abdeckung durch die Tests ermitteln. Letzteres erfordert jedoch im Bereich der Deeply Embedded Systems, also Systemen mit extrem knappen Resourcen, oft die Verwendung von kleineren selbstgeschriebenen Werkzeugen.

Die durch den Variantenreichtum hohe Anzahl an Testfällen führte zu der Idee, Testfälle mit ähnlichem oder sogar identischem Ablauf zusammenzufassen und stattdessen die statischen Eigenschaften des Testfalles automatisch innerhalb vorgegebener Parameter zu variieren. Auf diese Weise lässt sich die Anzahl der Testfälle erheblich reduzieren, ohne dass sich die Qualität des Gesamttests verschlechtert. Eine weitere Reduzierung der Testfälle durch den Einsatz von Modularisierung ist im Umfeld variantenreicher Software für Deeply Embedded Systeme nur sehr begrenzt möglich, da die Software oft so kompakt ist, dass es (fast) keine Module gibt, auf die in einzelnen Testfällen verzichtet werden könnte. Es ist vielmehr der Regelfall, dass man alle Module übersetzen und zusammenbinden muss, um überhaupt ein ausführbares System zu erhalten (vgl. Abb. 2). Zu Beginn eines Tests wird also aus einer automatisch variierten statischen Beschreibung (C-)Code erzeugt, dieser zusammen mit dem Testrahmen (=dynamischer Testanteil) übersetzt, zusammengebunden, auf die Zielplattform heruntergeladen und ausgeführt. Nach Testende wird das Testergebnis und die bei diesem Test ermittelte Abdeckung über einen Debugger aus der Zielplattform ausgelesen und steht zur weiteren Auswertung zur Verfügung. Zur Ermittlung der Abdeckung wurde der Testling zuvor manuell instrumentiert. Die Instrumentierung ist dabei fester Bestandteil des C-Codes, wird jedoch nur durch bestimmte Compilerschalter aktiviert. Bei aktivierter Instrumentierung wird in einem Bitarray abgelegt, welche Programmteile durchlaufen wurden.

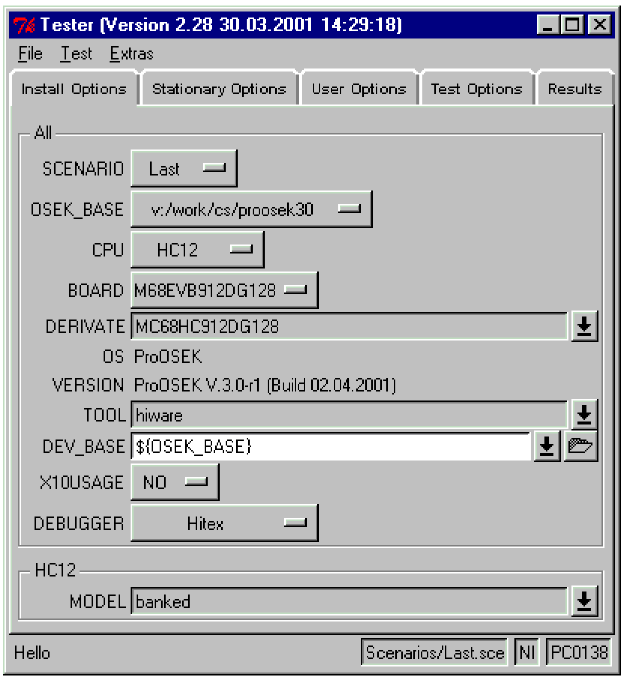

Abb. 1: Die GUI der TestSuite mit einigen Parametern

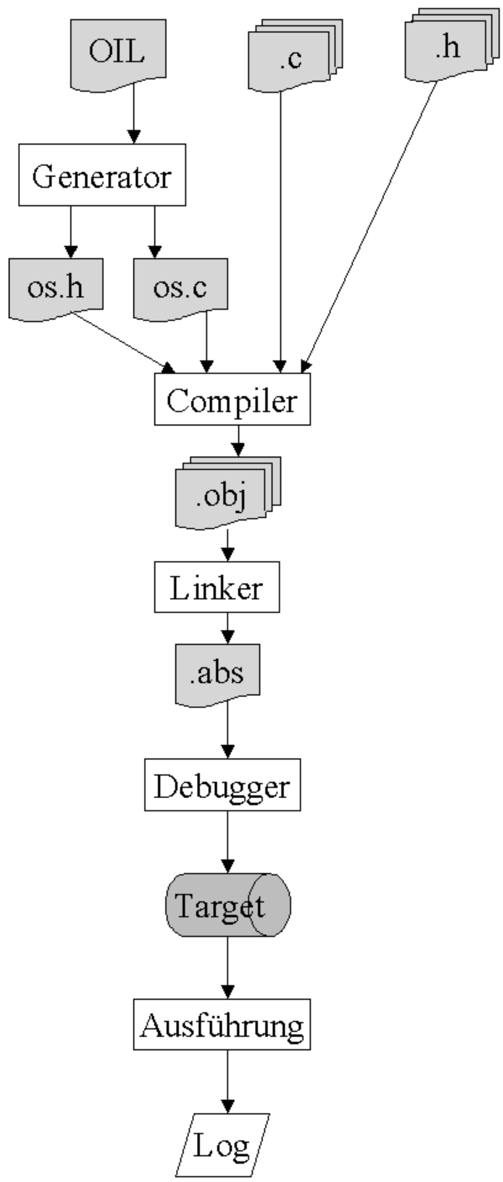

Abb. 2: Software-Entwicklung mit OSEK

Die Automatisierung der Tests erlaubt eine höhere Testhäufigkeit und reduziert den Zeitaufwand für die Mitarbeiter. Dazu erstellten wir eine grafische Benutzeroberfläche (siehe Abb. 1). Dies wurde notwendig, um die Bediensicherheit der Testsuite zu erhöhen, da es sehr leicht möglich war, die Vielzahl an Parametern in unsinniger oder sogar falscher Weise zu kombinieren bzw. einzustellen. Die Erhöhung der Akzeptanz dieses Werkzeugs ist ein nützlicher Seiteneffekt. Das Problem der Bediensicherheit wurde hierbei durch die Verwendung von Constraints gelöst. Die Änderung eines Parameters führt dadurch sofort dazu, dass ggf. unpassende Parameter automatisch einen neuen, passenden Wert erhalten. Darüberhinaus wurden die Parameter in vier Gruppen unterteilt: benutzerspezifische Parameter (z.B. der vom Benutzer bevorzugte Browser), sitzungsspezifische Parameter (z.B. die zur Zeit angeschlossene Hardware), workstationspezifische Parameter (z.B. die Installationspfade von Compilern und Debuggern) und schließlich die szenariospezifischen Parameter (z.B. die Liste der anzuwendenden Testfälle und ihrer Variationen). Dies erlaubt den einfachen Austausch und die einfache Wartung von Testszenarien zwischen verschiedenen Testern und Workstations.

Vollständige Regressionstests, wie man sie sich idealerweise wünscht, sind jedoch nicht möglich. Einerseits besitzen manche Microcontroller so geringen Resourcen, dass deren Flash-Speicher für die Testdurchführung genutzt werden muss. Dies verkürzt jedoch die Lebensdauer dieser Speicher und damit der Microcontroller signifikant. Andererseits besteht ein Teil des Variantenreichtums in einer Vielzahl unterschiedlicher Microcontroller-Derivate, so dass es mit vertretbarem Aufwand zur Zeit kaum möglich scheint alle Derivate vollständig zu testen. Schließlich ist auch noch die Testdauer zu erwähnen, die es unmöglich macht, vollständige Regressionstests jede Nacht durchzuführen.

Die Testsuite bietet also eine umfangreiche Unterstützung für verschiedene Aspekte des Testprozesses. Darüber hinaus sind Verbesserungen geplant. So soll die Granularität und der Umfang der Testabdeckungsmessung erweitert werden. Ein zweiter Punkt ist die Fertigstellung und Integration einer Testfallspezifikationssprache, um die Wartung und Erweiterung der Testfälle zu vereinfachen. Die Einschränkungen beim Regressionstest scheinen jedoch auf Grund von Hardware- und einem ungünstigen Kosten-Nutzen-Verhältnis zur Zeit nicht zu beheben sein.

Literatur

J.J.P. Tsai, S.J.H. Yang: Monitoring and Debugging of Distributed Real-Time Systems, IEEE, 1996.

S. Beatty: Sensible Software Testing, Embedded Systems Programming August 2000, CMP Media Inc. San Francisco.

3SOFT GmbH: ProOSEK User's Guide, 2001.