



06/2007

## Neue Wege in der Testautomatisierung

Einsatz grafischer Programmiersprachen in der Testautomatisierung

In vielen Fällen werden Qualitätsprobleme auf Defizite beim Test zurückgeführt. Die Tester führen zu ihrer Verteidigung an, dass sie nur unzureichende Informationen erhalten und nicht ausreichend Zeit zur Verfügung steht. Gute Gründe um über neue Wege der Kommunikation zwischen Entwicklern, Testern und der Testautomatisierung nachzudenken.

Seit mehreren Jahren führt MicroConsult Analyseworkshops mit seinen Kunden durch. Ziel ist es, mit dem Projektteam herauszufinden, wo Ansatzpunkte liegen, um sich das Leben zu erleichtern. Letzteres heißt im Projekt: Kosten- und Zeitdruck vermindern. Kosten- und Zeitdruck sind die entscheidenden Ursachen für Qualitätsmängel. Die Lösung: Bessere Anpassung der Arbeitsteilung zwischen Menschen und Maschinen bzw. Computern an die neuen Bedingungen unter Nutzung der verfügbaren Möglichkeiten. Die wesentlichste Maßnahme ist es, dabei die spezifischen Fähigkeiten von Menschen und Maschinen besser zu nutzen und aufeinander abzustimmen. Genau das ist das Ziel des hier vorgestellten Frameworks zur Testautomatisierung.

### Stärken und Schwächen

Die herausragende Stärke des Menschen liegt in seiner sehr hohen Wahrnehmungs-, Assoziations- und Kommunikationsfähigkeit. Mit ca. 10 Millionen Input- und ca. 3 Millionen Outputkanälen und einer CPU mit ca. 100 Mia Neuronen, die Signale hauptsächlich parallel verarbeiten, ist der Mensch jedem Computer auf diesem Gebiet weit überlegen. Allerdings gewichtet das Gehirn des Menschen alle Signale bei der Verarbeitung emotional. Das heißt, es gibt weder eine objektive Wahrnehmung noch eine objektive Kommunikation, auch wenn die Techniker dies gerne bestreiten. Je nach Stimmung, Situation und Hormonspiegel kann das zu außergewöhnlich produktiver oder destruktiver Kreativität führen.

Die Maschinen bzw. Computer dagegen verfügen über eine für menschliche Gehirne nicht erreichbare Wiederholgenauigkeit und Abarbeitungsgeschwindigkeit für weitgehend sequenzielle Abläufe. Sie kennen weder Stimmungen noch Hormone, die sie in positiver oder negativer Richtung vom „Pfad der Tugend“ (d.h. Programm) abbringen.

Die Anzahl störender Einflüsse ist aufgrund der wenigen IO-Kanäle vergleichsweise gering. Wer die Arbeitsteilung Mensch-Maschine optimieren will, sollte diese Eigenschaften im Hinterkopf behalten und Technologien, Tools, Methoden und Prozesse unter diesen Gesichtspunkten kritisch unter die Lupe nehmen.

Der hier beschriebene Ansatz zur Testautomatisierung ist unter anderem durch die Einbeziehung dieser Aspekte entstanden.

### **Das Prinzip**

Beim Systementwurf mit der UML entstehen unter anderem sogenannte Sequenzdiagramme. Diese beschreiben die zeitliche Abfolge von Interaktionen, die zwischen System und Umgebung ablaufen. Sie stellen damit auch das Verhalten dar, das durch einen Systemtest bestätigt werden muss. Im Gegensatz zu Beschreibungen in Prosa kommt diese Darstellung dem menschlichen Wahrnehmungsvermögen sehr entgegen.

Sie beschleunigt das Verstehen der Zusammenhänge und erleichtert die Kommunikation zwischen Testern und Entwicklern. Es liegt also nahe, die UML-Diagramme auch für die Definition von Testszenarien und für die Testspezifikation einzusetzen. Tester und Entwickler benutzen damit die gleiche Notation. Das schützt vor Missverständnissen und spart Zeit.

Auch bei der Umsetzung der Spezifikation in Testfälle und Testtreiber setzen wir auf eine effektive Mensch-Maschine-Arbeitsteilung. Test Management Center und Test Configurator stellen sicher, dass keine Parameter vergessen werden und sorgen für Ordnung in der Daten- und Dateiflut. Der Tester besetzt die Parameter nach Erfahrung und Intuition, die sich im Laufe seines Lebens entwickelt hat. Nicht selten spielt hier das berühmte Bauchgefühl eine wichtige Rolle.

In vielen Fällen wird die Umgebung des Prüflings durch einen HIL-Teststand (Hardware in the Loop) nachgebildet, weil die reale Umgebung noch nicht verfügbar, zu teuer oder zu gefährlich wäre. Um die oben beschriebenen Vorteile grafischer Programmierung auch hier zu nutzen, programmiert der Tester die Testtreiber und das Umgebungsmodell mit LabVIEW.

Der Test Executor verbindet die UML-Sequenzdiagramme mit LabVIEW-Programmen zu einem ausführbaren Testprogramm. Dieses kann dann auf einem PC oder, wenn Echtzeit gefordert ist, auf einem PXI-System (modulares Steckkartensystem mit Echtzeitrechner und IO-Baugruppen) ausgeführt werden. Das Test Management Center sorgt für eine weitgehend automatisierte Dokumentation der Testabläufe und Ergebnisse.

Wie wir alle wissen, ist Dokumentation eines der ungeliebtesten Kinder, d.h. das Gehirn entwickelt emotionale Widerstände, die das Ergebnis negativ beeinflussen. Ein guter Grund, diese Aufgabe Computern zu überlassen. Im Folgenden wird die Lösung ausführlicher vorgestellt.

### UML verbindet Requirements und Testszenarien

Die Sammlung und Strukturierung der Systemanforderungen (Requirements) sind die Grundlage für die Spezifikation eines Systems. Meist handelt es sich dabei um umfangreiche textuelle Beschreibungen. Doch solche Dokumente haben ihre Tücken:

- Inkonsistenz der Informationen über das gesamte Dokument
- Ungenau spezifizierte Abläufe
- Nicht testbare Anforderungen

Schon hier hilft die UML dabei, Defizite zu vermeiden. UML-Diagramme, beispielsweise Sequenzdiagramme (Abbildung 1), zeigen anschaulich das Zusammenspiel von System und Systemumgebung und bieten Unterstützung beim Spezifizieren.

Andere Diagrammformen aus dem UML-Werkzeugkasten, wie z.B. Use Case Diagramme, sind ebenfalls wertvolle Hilfen beim Systementwurf. Das Case Tool dient nicht nur der Darstellung und Konstruktion von UML-Diagrammen, sondern auch dem Content Management für die Spezifikation. Leistungsfähige Reporttools generieren aus dem UML-Modell und den verlinkten Informationen eine vollständige Dokumentation.

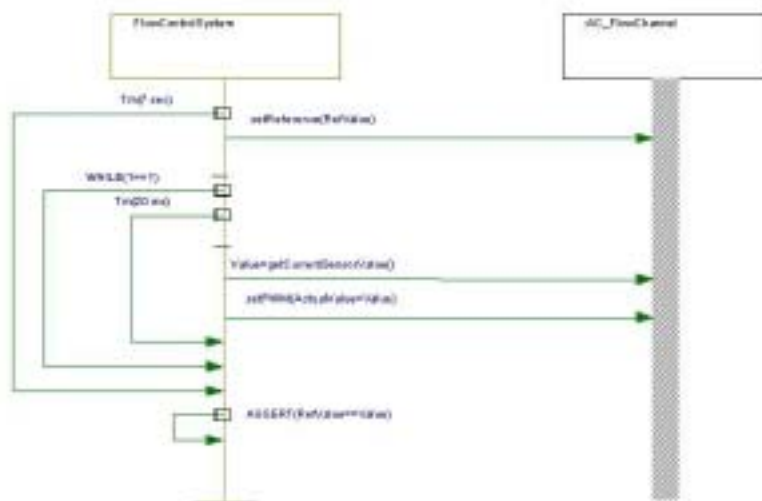


Abbildung 1: Einfaches Beispiel für ein Sequenzdiagramm

## Testsequenzen in UML

Das UML-Sequenzdiagramm in Abbildung 1 zeigt das Zusammenspiel zwischen dem Prüfling (FlowControlSystem) und einem Akteur (ACTors) seiner Umgebung (AC\_FlowChannel). Die Pfeile zwischen den Akteuren und dem Prüfling stehen für Interaktionen, die in ihrer zeitlichen Abfolge von oben nach unten dargestellt sind. Außerdem sind Zeitbedingungen, Wiederholungen von Teilsequenzen und ein Soll-Istwertvergleich mit ASSERT zu sehen. In den Operationen lassen sich Parameter (z.B. RefValue) angeben. Natürlich können in einem solchen Diagramm auch sehr viel komplexere Abläufe beschrieben werden.

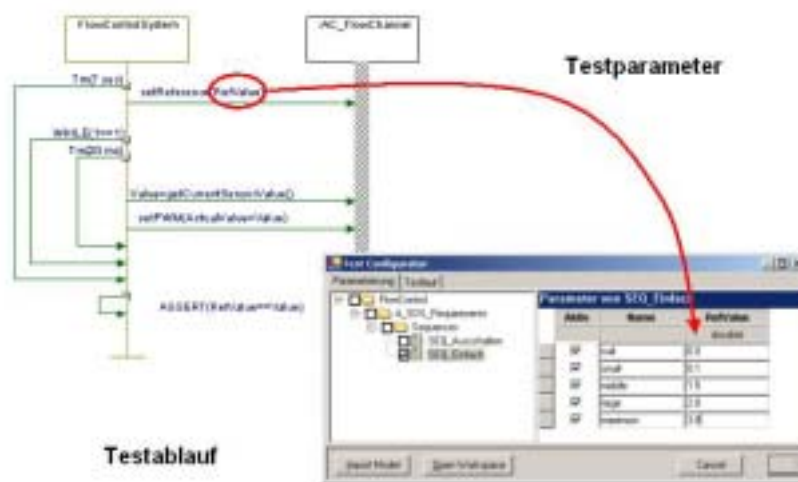


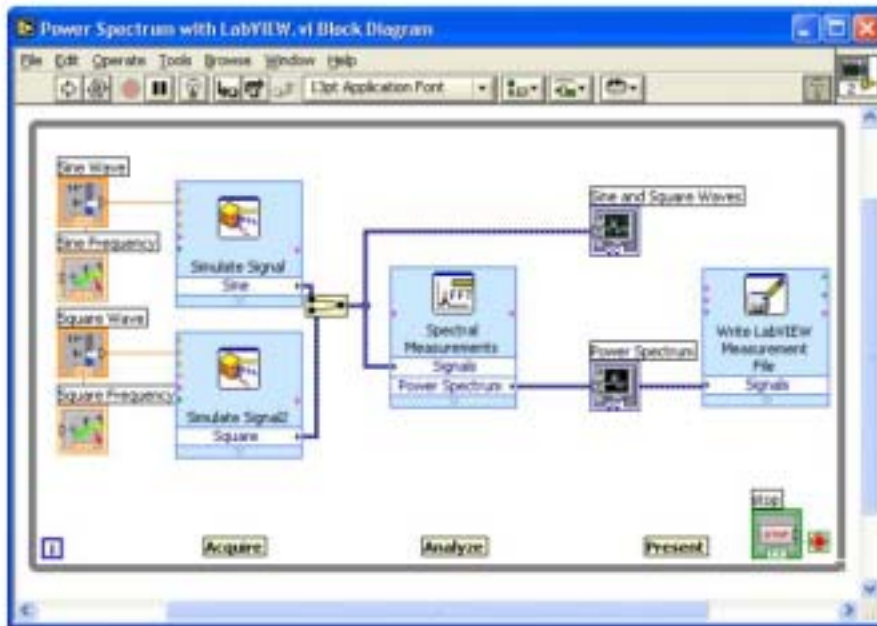
Abbildung 2: Parametrierung der Variablen und Ablage der Testfälle

Wird einem Parameter innerhalb der Sequenz kein Wert zugewiesen, so wird er in dieser Testautomatisierungslösung als Variable angesehen. Diese Variablen werden durch den Test Configurator erkannt, sobald eine Testsequenz ausgewählt wird. Dann öffnet sich eine Tabelle, in der der Tester die Variablen für die gewünschten Testfälle mit Werten versehen kann. In Abbildung 2 ist auch links im Kasten die Ablagestruktur der Testsequenzen innerhalb des Paketbaumes des UML-Modells zu sehen.

## Testtreiber und Streckenmodell mit LabVIEW

Aus dem UML-Modell und insbesondere den Sequenzendiagrammen für die Tests ergeben sich die notwendigen Testtreiber und die Eigenschaften des Streckenmodells. Der Tester ist damit z.B. in der Lage, Testtreiber und Umgebungsmodell zu programmieren und mit dem Test Executor zu einer Testumgebung für den Prüfling zu verknüpfen (Abbildung 3).

Auch hier bietet die Anwendung einer grafischen Programmiersprache Vorteile. In unserem Beispiel wurde das weit verbreitete LabVIEW eingesetzt. Damit lassen sich alle Messkarten, die konform zum PXI-Standard sind, einbinden sowie beliebige Windows-Komponenten aufrufen (DLLs, NET-Komponenten, COM/OLE-Module, C-Programme). Die Möglichkeiten sind aufgrund der Vielfalt verfügbarer Messkarten und Windows-Komponenten praktisch unbegrenzt.

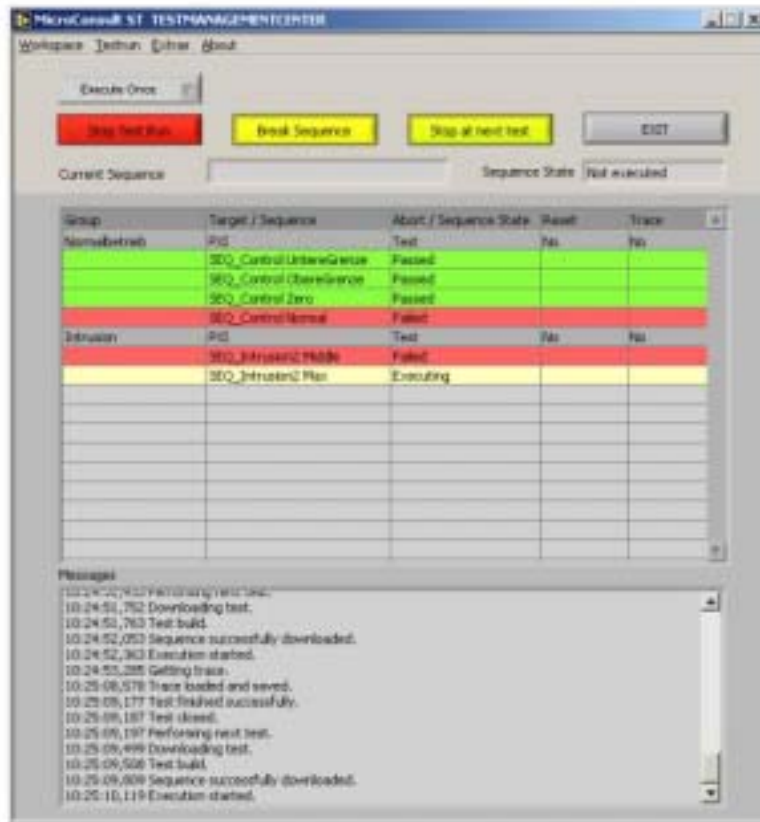


*Abbildung 3: Beispiel für LabVIEW-Programm*

### **Testausführung, -management und -dokumentation**

Nachdem die Testsequenzen konstruiert und parametrisiert sowie die Testtreiber und Streckenmodelle implementiert sind, kann der Test starten. Der Test Executor verbindet die parametrisierten UML-Sequenzen mit den Testtreibern und dem Umgebungsmodell zu einer auf dem PC oder auf einem PXI-System ausführbaren Testumgebung. Je nach Wahl des Testtargets wird eine Ausführung in Echtzeit mit einer Schrittweite ab 25 ns erreicht.

Im Test Management Center werden die einzelnen Testsequenzen in Gruppen organisiert. Jede Gruppe beinhaltet die Information über den Ausführungsort des Tests (PC, PXI oder FPGA-Messkarte). Alle Testsequenzen zusammen ergeben einen Testlauf. Bei der Ausführung des Testlaufs werden die durchgeführten Testschritte abhängig von Status und Ergebnis farblich markiert.



*Abbildung 4: Test Manager mit Bedienelementen, Statusanzeige für Testsequenzen und Visualisierung der aktuell ausgeführten Testschritte*

Die Ergebnisse des Testlaufs werden automatisch in Verzeichnissen, die nach Uhrzeit und Datum des Testlaufs benannt sind, abgelegt. Nach dem Testlauf stehen dem Tester Testprotokolle (z.B. nach IEEE 829), Problemreports für jeden gefundenen Fehler und das vollständige Trace-Protokoll für die ausgeführten Sequenzen als Debughilfe zur Verfügung. Da das Test Management Center im LabVIEW Source-Code vorliegt, können auch sehr leicht eigene Protokolle erzeugt werden.

### Zusammenfassung

Die UML bietet die Möglichkeit, eine für den Menschen besser durchschaubare Beschreibung von komplexen Systemen zu erstellen. Es liegt nahe, diese Vorteile sowohl in der Entwicklung als auch im Test einzusetzen. Jede grafische Darstellung hat ihre spezifischen Vorteile. So bietet UML gute Ansätze für die räumliche und zeitliche Strukturierung von komplexen digitalen Systemen; LabVIEW wiederum orientiert sich stärker an den Darstellungsgepflogenheiten der Messtechnik. Die hier beschriebene Lösung zeigt, dass sich durch geeignete Frameworks die Vorteile verschiedener grafischer Programmierungen in der Testautomatisierung sehr gut nutzen und kombinieren lassen.

Unter [www.microconsult.de/service](http://www.microconsult.de/service) finden Sie wertvolle Literaturtipps und interessante Downloads zum Thema Entwickeln und Testen.

**Autor:**

Dipl.-Ing. Peter Siwon, MicroConsult GmbH

Geschäftsführer

Charles-de-Gaulle-Str. 6, 81737 München

Tel.: 089 450617-44, Fax: 089 450617-17, p.siwon@microconsult.com

**MicroConsult GmbH:**

Training, Coaching und Engineering für Software- und Hardwareentwicklung in der Industrie.

[www.microconsult.de](http://www.microconsult.de)