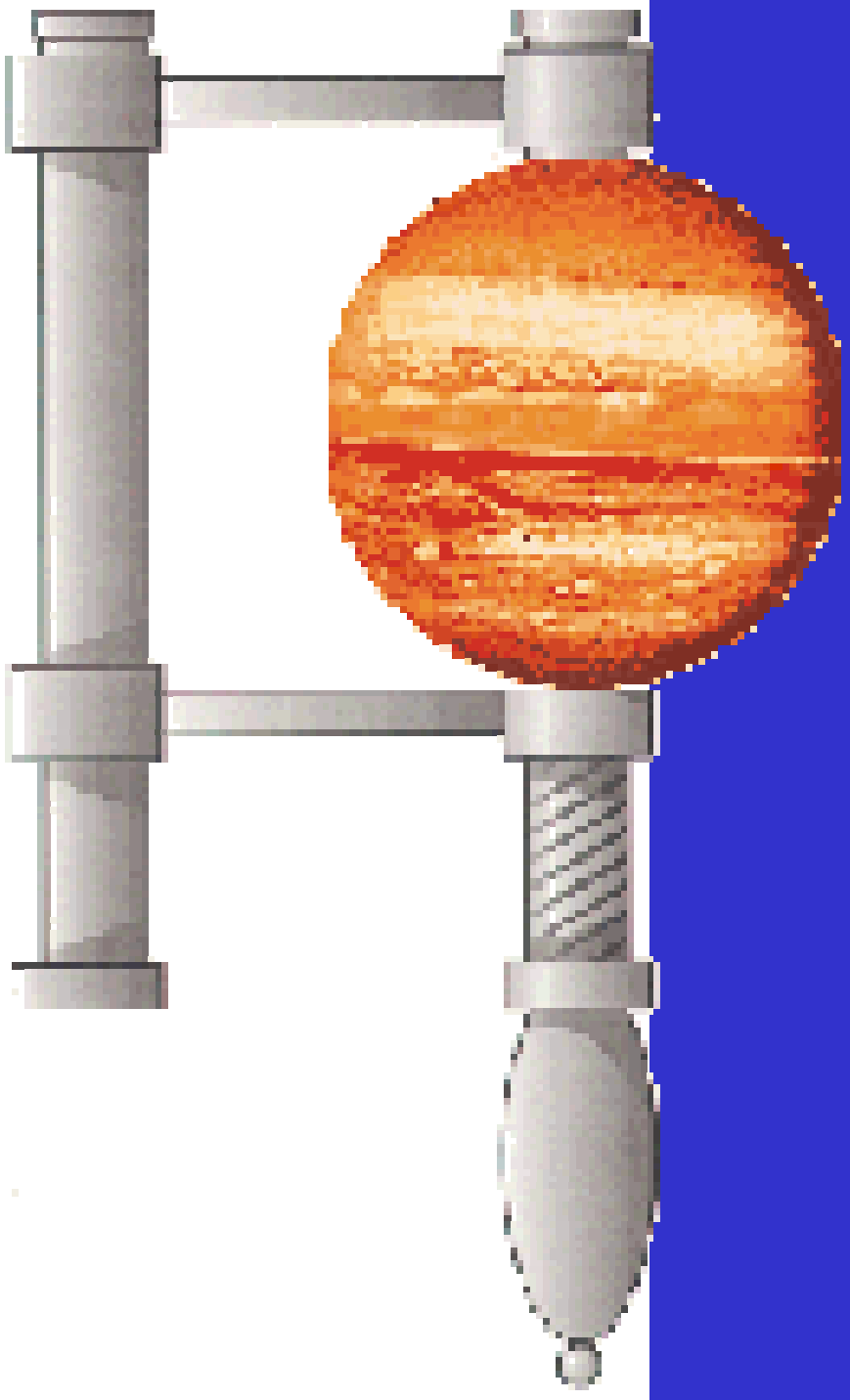


Test eingebetteter Systeme

- Systematischer Test - CTE
- Testsystem Tessy
- Target Test

Test eingebetteter Systeme

Elektronische Steuerungssysteme, sogenannte eingebettete Systeme, sind in den verschiedensten Anwendungsbereichen im Einsatz. Anwendungen finden sich in der Telematik, Luft- und Raumfahrt, Automobilbau, Konsumelektronik, Medizintechnik, u.a.

Der wesentliche Unterschied eingebetteter Systeme zu klassischen Rechnersystemen besteht darin, daß sie ein Teil eines größeren Gesamtsystems sind, bzw. primär keine Rechner sind. So ist ABS (Anti-Blockiersystem) Bestandteil eines Fahrzeugs (Bild 1).

Bild 1: Gesamtsystem Fahrzeug.

Ein weiteres Merkmal ist die große Vielfalt an Mikroprozessoren, die bei den einzelnen Anwendungsgebieten anzutreffen sind. In der Mehrzahl werden derzeit 4-, 8- und 16-Bit Prozessoren eingesetzt. Die kurzen Innovationszyklen im Hardwarebereich führen dazu, daß sich nur die Compiler-Entwicklung für C lohnt. Aus diesem Grund und auch weil mit C hardwarenah programmiert werden kann, trifft man im embedded Bereich überwiegend C als höhere Programmiersprache an. Nach wie vor wird auch Assembler für rechenintensive Algorithmen verwendet. Bei Prozessoren, die auch in Rechnersystemen zum Einsatz kommen, stehen auch andere Programmiersprachen zur Verfügung. So werden bei Navigationssystemen C++ oder Java verwendet.

Immer mehr mechanische und elektrische Komponenten werden durch Elektronik ersetzt. Stichwörter sind X-by-wire Entwicklungen, wie z.B. beim Airbus. Damit wird eine Verringerung des Gewichts und Volumens erzielt, aber auch größere Flexibilität durch die Software erreicht. Auf der anderen Seite steigt die Komplexität dieser Systeme. So hat zwar ein Rasierapparat weniger als 100 KB Steuersoftware, aber ein Navigationssystem umfaßt schon mehr als 30 MB Software. Neben der Softwaregröße erhöhen auch Echtzeitanforderungen, die Vernetzung von Steuergeräten und die zu beachtenden sicherheitskritischen Aspekte die Komplexität. Während bei einem Rasensprenger keine harten Echtzeitanforderungen vorliegen, da unkritisch ist, ob das Wasser ein paar Sekunden früher oder später abgeschaltet wird, kann ein Versagen beim Abschalten des Wassers einer Waschmaschine schon zu erheblichen Sachschäden führen. Noch kritischer ist das Versagen der Schubumkehr bei einem Flugzeug, was im schlimmsten Fall zu Personenschäden führen kann. Da Fehler in eingebetteten Systemen generell sehr kostspielig sind, wachsen auch die Anforderungen an die Software und ihre Qualität. Das wichtigste und am Qualitätssicherung von eingebetteten Systemen ist der Test. Bei einem Verhältnis von bis zu 80% Testaufwand zu 20% Codierung wird deutlich, daß nur durch systematisches Testen mit weitesten verbreitete analytische Verfahren zur hohem Automatisierungsgrad die Software-Entwicklungskosten bei gleichbleibender oder höherer Softwarequalität zu reduzieren sind.

Im folgenden werden Testverfahren, Methoden und Werkzeuge kurz vorgestellt, die bereits Eingang in die Praxis gefunden haben, wie der systematische Test (Black-Box/White-Box-Verfahren) und die Testfallermittlung auf der Basis der Klassifikationsbaum-Methode und des Werkzeugs CTE. Zur weitestgehenden Automatisierung des Tests von C-Funktionen wird vor allem im Fahrzeugbau das Testsystem Tessy eingesetzt. Für eingebettete Systeme spielt die Zielhardware die entscheidende Rolle und damit der Target-Test, auf den hier ausführlich eingegangen wird.

Systematischer Test – CTE

Ein systematischer Test gliedert sich in die Aktivitäten Testfallermittlung, Testdatengenerierung und Sollergebnisbestimmung, Testdurchführung, Monitoring und Testauswertung sowie die den Test vorbereitenden und begleitenden Aktivitäten der Testplanung, Testorganisation und Testdokumentation (Bild 2).

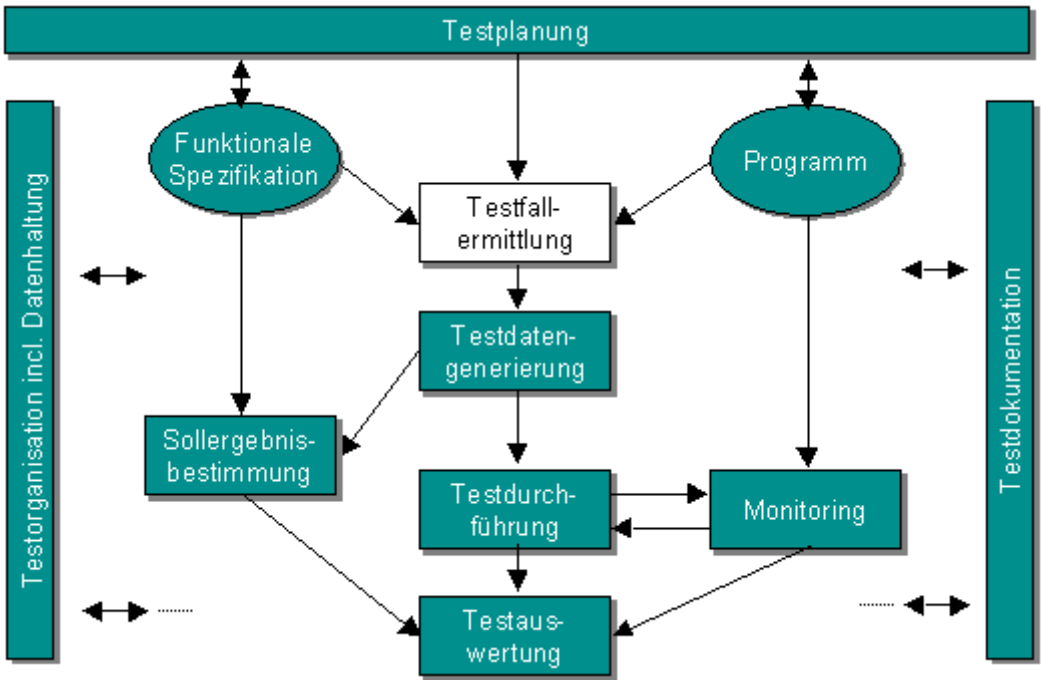


Bild 2: Testaktivitäten.

Die wichtigste Aktivität ist dabei die Ermittlung von Testfällen, mit denen der Test durchgeführt werden soll, da hier Art und Umfang der Prüfung festgelegt werden und damit die Güte des Tests bestimmt wird. Zwei in der Praxis weit verbreitete Verfahren zur Testfallermittlung sind der Funktionstest („Black-Box-Test“) und der Strukturtest („White-Box-Test“). Beim Funktionstest werden die Testfälle aus der Spezifikation, das heißt, der Aufgabenbeschreibung des Testobjekts, abgeleitet. Beim Strukturtest werden die Testfälle aus der Struktur des zu testenden Programms abgeleitet. Eines der gebräuchlichsten Verfahren ist der Zweigtest, bei dem im Laufe der Prüfung alle Verzweigungen des Programms mindestens einmal durchlaufen werden sollen. Als besonders effektiv hat sich die Kombination beider Ansätze erwiesen, wo Stärken und Schwächen beider Verfahren sich ausgleichen. Da die Ermittlung von Strukturtestfällen in der Regel aufwendig ist, ist es effizienter zunächst den Funktionstest durchzuführen. Anschließend erfolgt für die nicht überdeckten Programmbereiche der Strukturtest.

Der Funktionstest nimmt also eine zentrale Stellung beim Test ein. Daher ist ein methodisches Vorgehen unerläßlich. Mit der Klassifikationsbaum-Methode steht eine systematische und leicht erlernbare grafische Testmethode zur Verfügung, die zu redundanzarmen und fehlersensitiven Testfällen führt. Die grundsätzliche Idee der Klassifikationsbaum-Methode (Bild 3) ist es, zuerst die Menge der möglichen Eingaben für das Testobjekt getrennt auf verschiedene Weisen, unter jeweils einem geeigneten Gesichtspunkt zu zerlegen, um dann durch Kombination dieser Zerlegungen zu Testfällen zu gelangen.

Zunächst ist der Tester aufgefordert, für den Test relevante Gesichtspunkte aufzustellen. Jeder Gesichtspunkt soll eine eng begrenzte und damit übersichtliche Unterscheidung der möglichen Eingaben für das Testobjekt erlauben. Im folgenden Schritt wird unter jedem Gesichtspunkt eine Zerlegung der Menge der möglichen Eingaben vorgenommen. Diese Zerlegung ist eine Klassifikation im mathematischen Sinne, das heißt die Menge der möglichen Eingaben wird disjunkt und vollständig in Teilmengen, sogenannte Klassen, zerlegt.

Da die Zerlegung jeweils getrennt unter nur einem Gesichtspunkt erfolgt, ist sie relativ leicht durchzuführen. Zu jedem Gesichtspunkt entsteht eine Klassifikation.

werden. Die möglichen Eingabedaten sind einfache geometrische Körper, wie Kreis, Dreieck und Quadrat. Aspekte für den Test sind beispielsweise die Größe, die Farbe und die

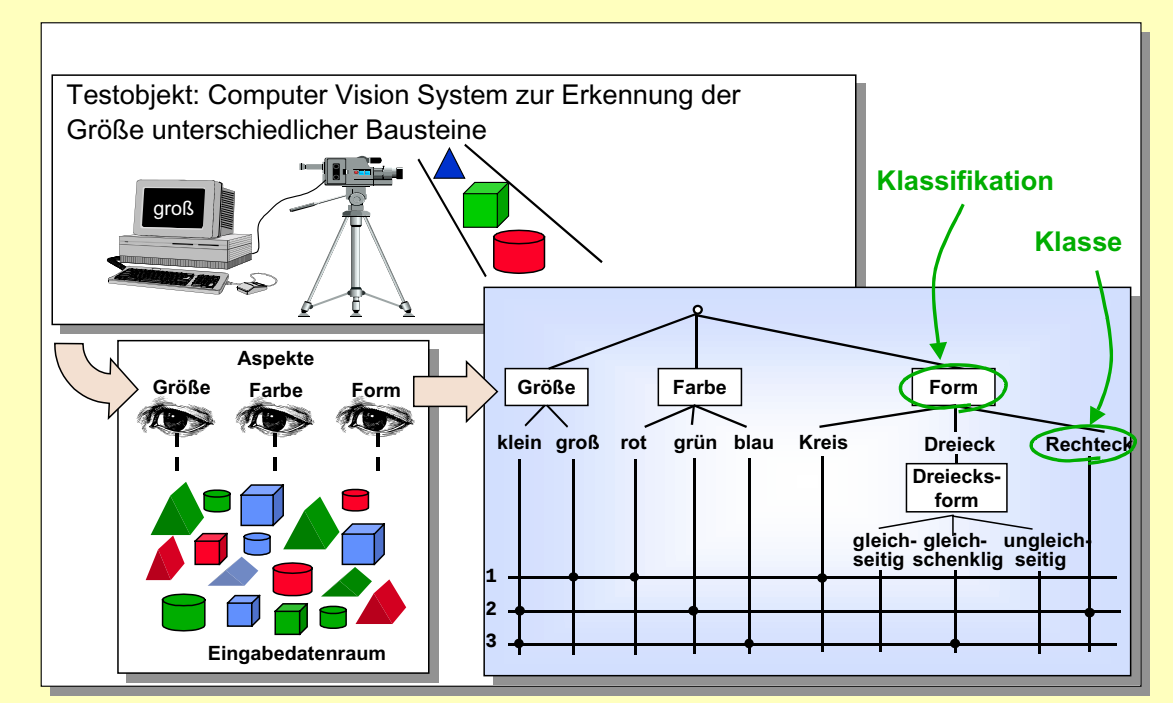


Bild 3: Klassifikationsbaum-Methode.

Vielfach ist es sinnvoll, Klassifikationen einzuführen, die nicht den gesamten Eingabedatenraum unterteilen, sondern nur eine Klasse einer anderen Klassifikation. Durch die rekursive Anwendung von Klassifikationen auf Klassen entsteht ein Baum von Klassifikationen und Klassen.

Ein Testfall entsteht durch die Kombination von Klassen unterschiedlicher Klassifikationen, wobei aus jeder eingeführten Klassifikation genau eine Klasse berücksichtigt wird. Ein Testfall ist also die durch Bildung des Durchschnitts der jeweils gewählten Klassen entstehende Schnittmenge. Bei der Kombination der Klassen ist auf logische Vereinbarkeit zu achten, das heißt die Schnittmenge darf nicht leer sein. Der Tester wählt insgesamt so viele Kombinationen als Testfälle, daß alle Gesichtspunkte ausreichend, auch in ihrer Kombination, berücksichtigt werden.

In Bild 3 ist ein Computer Vision System gezeigt. Die Aufgabe des Systems besteht darin, die Größe der Objekte auf einem Laufband zu erkennen, die von einer Kamera aufgenommen

Form der Objekte (Bild 3 unten links).

Die Klassifikation unter dem Aspekt Farbe führt zu einer Zerlegung nach roten, grünen und blauen geometrischen Elementen. Bei der Klassifikation Form erhält man eine Zerlegung in Kreis, Dreieck und Quadrat. Die Klassifikation Größe wird schließlich in die Klassen groß und klein zerlegt. Die Klassifikationen und die zugehörigen Klassen bilden den Klassifikationsbaum (Bild 3 unten rechts). Dabei werden die Klassifikationen als benannte Rechtecke dargestellt. Darunter werden die jeweiligen Klassen angeordnet. Um die Klassenkombination als Testfälle festzulegen, wird der Klassifikationsbaum als Kopf einer Tabelle verwendet, in der die zu kombinierenden Klassen markiert werden. Jede Tabellenzeile repräsentiert einen Testfall.

Mit dem CTE (Classification Tree Editor) steht ein komfortables Werkzeug zur Unterstützung der Klassifikationsbaum-Methode zur Verfügung. Das Werkzeug unterstützt die Erstellung des Klassifikationsbaums sowie der Kombinationstabelle und prüft die syntaktischen Regeln

der Methode. Er ermöglicht auch die Eingabe von Abhängigkeiten zwischen Klassen unterschiedlicher Klassifikationen und die regelbasierte Generierung von Testfällen. Umfangreiche Import und Exportmöglichkeiten erlauben eine nahtlose Integration des CTE in den Testprozeß. So ist der CTE auch integraler Bestandteil des Testsystems Tessy.

Testsystem Tessy

Das Testsystem Tessy unterstützt den Unit- und Integrationstest von C-Funktionen. Alle in Bild 2 gezeigten Testaktivitäten werden von Tessy durch passende Werkzeuge unterstützt, die eng miteinander integriert sind. Stärken des Testsystems Tessy sind die durchgängige Unterstützung aller Testaktivitäten, die Kombination von Funktionstests und Strukturtests für die Testfallermittlung, der hohe Automatisierungsrad, die vollständige Automatisierung von Regressionstests und die Testdurchführung sowohl auf dem Entwicklungsrechner als auch auf verschiedensten Target-Systemen (Debugger- und PCMCIA-Karten, Emulatoren und Simulatoren mit unterschiedlichsten Microprozessoren und C-Compilern).

Das Testsystem Tessy bietet dem Benutzer die unter Windows gewohnte einheitliche Benutzungsoberfläche an. Eine kontext-sensitive Benutzerführung unterstützt den richtigen Einsatz der Testwerkzeuge. Zur Navigation werden Browser benutzt, beispielsweise für die Verwaltung von Projekten, Modulen und Testobjekten oder zur Anzeige der Testobjekt-Schnittstelle. Orientierung über die zu prüfenden Testobjekte mit ihren Bearbeitungszuständen bieten unterschiedliche Darstellungen (Icons), wobei der Arbeitszustand farblich markiert ist (Bild 4). Die Testdokumentation liegt in XML Format vor und wird als HTML angezeigt. Sie ist wahlweise auch als ASCII-Format verfügbar.

Neben den bereits oben erwähnten Testaktivitäten unterstützt Tessy eine Reihe von weiteren nützlichen Funktionen. Hierzu zählen insbesondere:

- **Ad-hoc Test:** Ein unsystematischer Test kann nach Bekanntgabe der Quelle in nur drei Schritten komplett durchlaufen werden, z.B. um die Testbarkeit zu überprüfen.
- **Schnittstellenübersicht:** Mit der automatischen Schnittstellen- und Passierrichtungserkennung werden alle Ein- und Ausgangsvariablen des Testobjekts mit

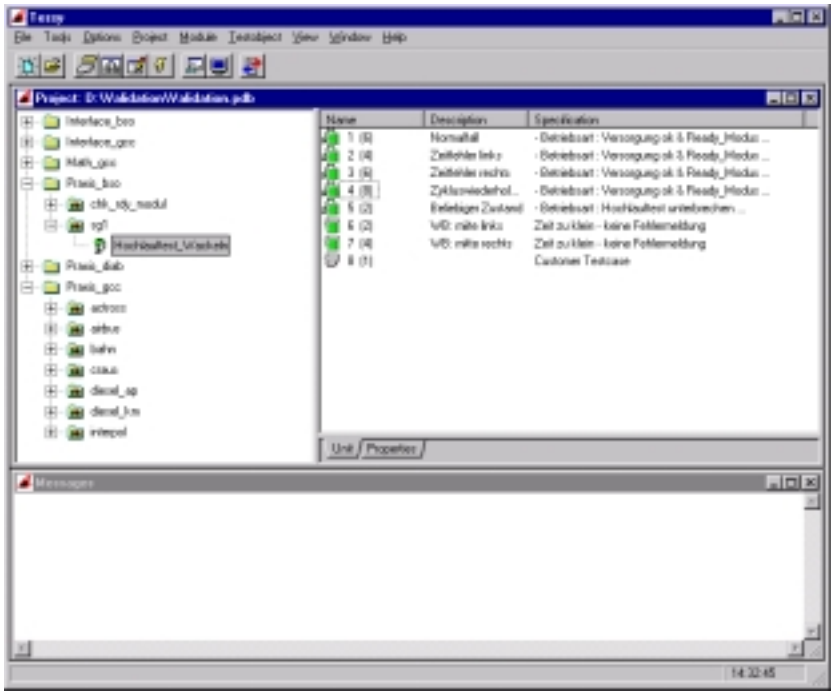


Bild 4: Startfenster Tessy.

Typ und ihrer Verwendung in einem Browser als Ein- oder/und Ausgangsparameter voreingestellt angezeigt.

- **Testumgebung:** Mit der kompletten Definition der Modulumgebung, inklusive Stubgenerierung für vom Testobjekt aufgerufene Funktionen, entfällt die manuelle Bereitstellung von Include- oder Objektdateien.
- **Projektdaten-Import**: Durch das Einlesen eines Makefiles werden alle Projektdaten, wie Quellen, Includepfade, Compileranweisungen, usw. von Tessy importiert.
- **Testdaten-Import/-Export:** Entsprechend der Schnittstelle einer C-Funktion können Testdaten importiert bzw. exportiert werden. Hierbei kann es sich um Testdaten handeln, die beispielsweise in Excel bearbeitet werden oder aus anderen Tests stammen, usw.
- **Datenkonvertierung:** Beim Einsatz von Integer-Prozessoren müssen Float-Werte in Integer-Werte umgerechnet werden. Diese Datenkonvertierung wird von Tessy automatisch vorgenommen. Das System wandelt die Float-Werte vor jeder Testdurchführung auf einem Integer-Prozessor um. Umgekehrt werden die Istwerte nach der Testdurchführung wieder in Float-Werte umgerechnet.
- **Re-run/Retest:** Wiederholungsläufe bei unveränderter Schnittstelle auf Entwicklungsrechnern- und unterschiedlichen Zielplattformen.
- **Batch-Test:** Damit können alle in einem Projekt verwalteten Tests ohne Eingriffe des Benutzers vollständig wiederholt werden, um beispielsweise nach größeren Änderungen alle Funktionen auf Seiteneffekte hin zu überprüfen.
- **Back-to-Back Test:** Dies unterstützt den Vergleich von Testergebnissen, die beispielsweise aus einem Testlauf in einer Simulation (Matlab/Simulink) erfaßt wurden, mit Testergebnissen aus einem Testlauf in der Zielumgebung (ab Version 3.0).
- **Zeitüberwachung:** Für jeden Testfall/ Testschritt wird die reale Ausführungszeit des Testobjekts auf dem Zielsystem gemessen und dokumentiert.
- **Laufzeittest:** Der Nachweis, definierte Zeitschranken (min./max. Laufzeit) einzuhalten, spielt bei eingebetteten Systemen eine immer wichtigere Rolle. Der evolutionäre Test unterstützt bereits für eine Reihe ausgewählter Steuerungsalgorithmen die vollautomatische Ermittlung der kürzesten bzw. längsten Laufzeit.
- **Zufallstest/Lasttest:** Vollautomatische Tests, bei denen die Testdaten entweder über einen Zufallsgenerator generiert werden oder von einem Lastprofil (Protokoll) abgeleitet sind (ab Version 3.0).
- **Test von CORBA-Applikationen:** Auf Basis der IDL (Interface Definition Language) können Applikationen unterschiedlicher Programmiersprachen auf beliebigen Hardwareplattformen/Betriebssystemen mit Tessy getestet werden.
- **Test von OSEK-Applikationen:** (Version 4.0).

Target Test

Für den Test von eingebetteten Systemen (Target Test) sind eine Reihe von Anforderungen durch ein Testsystem zu erfüllen. Hierzu zählt die Berücksichtigung von **Echtzeitbedingungen**. Um diese zu überprüfen, leitet der Tester Testfälle zur Feststellung der kürzesten und längsten Ausführungszeit aus dem Programmcode ab. Mit der Zeitüberwachung bietet Tessy die Information über die Laufzeit zu jedem Testfall/Testschritt. Die manuelle Ableitung ist sehr zeitaufwendig. Sie beansprucht einen gut ausgebildeten Programmierer, je nach Größe der Funktion, einige Stunden bis zu mehreren Tagen. Durch den in Tessy integrierten **evolutionären Test** wird die kürzeste bzw. längste Ausführungszeit vollautomatisch ermittelt. Entsprechende Suchalgorithmen sind beispielsweise für die Motorsteuerung entwikkelt worden. Damit kann zum jeweiligen Testziel, beispielsweise längste Ausführungszeit, der entsprechende Testfall ermittelt werden.

Weitere Anforderungen werden durch das Zielsystem (Target) gestellt. Welcher **Microprozessor** kommt zum Einsatz? Bei einem Integer-Prozessor müssen physikalische Daten (float-Werte), die während der Entwicklung verwendet wurden und nun im Test wiederverwendet werden sollen, in Integer-Werte umgewandelt werden. Tessy wandelt entsprechend vorgegebenen Konvertierungsalgorithmen vor dem Testlauf die Testdaten in Integer-Werte um. Nach dem Testlauf werden die Istwerte wieder in physikalische Werte konvertiert.

Eine weitere Anforderung ergibt sich aus dem **verfügbaren Speicherplatz**. Dieser ist in der Regel klein ausgelegt. Daraus folgt, daß der

Testtreiber möglichst wenig Speicherplatz in Anspruch nehmen darf. Um dies sicherzustellen, wird in Tessy eine **Client/Server-Architektur** verwendet (Bild 5). Tessy generiert zwei Testtreiber, einen Master und einen Slave. Der Testtreiber-Master läuft auf dem Entwicklungssystem (Host), auf dem auch Tessy arbeitet. Der Testtreiber-Slave läuft auf dem Target-System zusammen mit dem Anwendungsprogramm (Testobjekt). Beide Komponenten tauschen Daten über die jeweilig verfügbaren Kommunikationsmechanismen des Target-Systems aus. Die Größe des Slaves ist allein abhängig von der Breite der Schnittstelle und nicht von der Anzahl der auszuführenden Testfälle mit ihren Testdaten. Die Testdaten werden vom Master aus der Testdaten-Bank gelesen, die in der Entwicklungsumgebung abgelegt ist. Der Slave nimmt die Testdaten entgegen und ruft damit das Testobjekt auf. Nach dem Durchlauf werden die Istwerte vom Slave zum Master gesendet, der sie für die darauf folgende Testauswertung in der Testdaten-Bank ablegt. Mit dieser Client/ Server-Lösung ist es auch möglich unbegrenzt viele Testfälle in einem Testdurchlauf auszuführen.

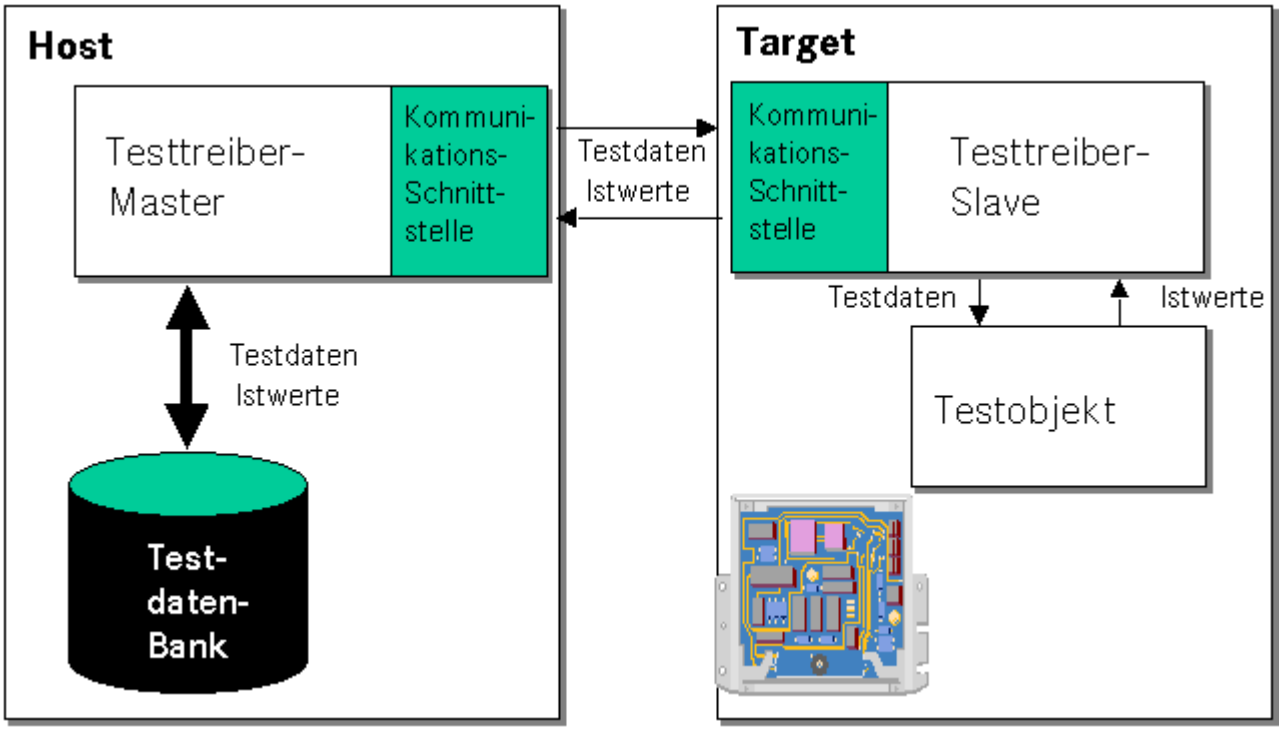


Bild 5: Client/Server-Architektur.

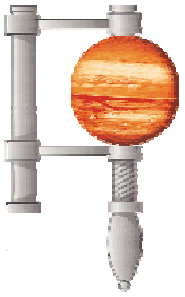
Das Testsystem Tessy unterstützt zur Zeit folgende Debugger, Emulatoren und Simulatoren:

- TRACE32, Lauterbach Datentechnik GmbH
- Fastview66 (Einsteckkarte und PCMCIA), PLS GmbH
- HiTOP, Hitex-Systementwicklung GmbH
- UDE, PLS GmbH
- WinIDEA, iSystem GmbH
- SingleStep Simulator, Wind River GmbH
- XRAY Simulator, Mercury Technologies, Inc.

Tessy kann um weitere Debugger, Emulatoren und Simulatoren schnell erweitert werden. Dies gilt auch für C-Compiler unterschiedlicher Hersteller. Aktuell werden folgende C-Compiler unterstützt:

- National CR16, National Semiconductor Corp.
- Tasking C166, Tasking GmbH
- Tasking Tricore
- DiabData PPC555, Wind River GmbH
- Cosmic HC12, Cosmic Software GmbH
- Cosmic HC08
- Metrowerks HC12, Metrowerks GmbH
- Keil C166, Keil Elektronik GmbH

Kontaktadressen:



DaimlerChrysler AG
Research and Technology, FT3/SM
Alt-Moabit 96a, D-10559 Berlin

R. Pitschinetz Tel.: 030 399 82 231
Dr. Grochtmann Tel.: 030 399 82 229
Dr. Wegener Tel.: 030 399 82 232

Internet:www.systematic-testing.com

Distribution Germany & Support:
Razorcat Development GmbH
Schwarzkopffstr. 6, D-10115 Berlin
Tel. +49 30 536 357 0
Fax. +49 30 536 357 60
Email: info@razorcat.com
Internet:www.razorcat.com/development

Distribution World Wide:
Hitex Development Tools
Greschbachstr. 12, D-76229 Karlsruhe
Tel. +(49) 7 21 / 96 28 – 125
Fax. +(49) 7 21 / 96 28 –149
Email: info@hitex.de
Internet:www.hitex.de/perm/tessy.htm