Schlechte Strukturierung und Dokumentation der Software kann zukünftige Erweiterungen so verteuern, dass die langfristige Wettbewerbsfähigkeit des Produktes gefährdet wird und die Wahrscheinlichkeit von Folgeaufträgen sinkt.

6.1 Teststrategie

Beim Test von Embedded-Software ist die Teststrategie entscheidend. Sie betrifft Modultest, Softwareintegration, Integrationstest und den Software-Anforderungstest. Je nach Aufgabenstellung werden auch Software-spezifische Themen für HW-/SW-Integrationstest, Komponententest und Systemtest in der Teststrategie berücksichtigt.

In der Teststrategie werden folgende Aspekte des Testens betrachtet:

- Sowohl Kunden als auch firmeninterne Qualitätsvorgaben stellen Anforderungen an den Test und beeinflussen Art und Umfang der Tests (Abschnitt 6.1.1).
- Vor der Entwicklung der Tests selbst sollten Sie die Voraussetzungen für die Tests klären, die Tests planen, die Integrationsstrategie festlegen und bestimmen, was genau getestet wird. Legen Sie Struktur und Inhalt der Testspezifikationen und Testprotokolle fest (Abschnitt 6.1.2).
- Definieren Sie auch die Testumgebung für die Embedded-Software sowie die Testwerkzeuge und legen Sie fest, wie geprüft wird, dass die Testeinrichtung zuverlässig und fehlerfrei arbeitet (Abschnitt 6.1.3).
- Zur Durchführung der Tests gehören neben individuellen Testvorbereitungen (Abschnitt 6.1.4), dem Test selbst (Abschnitt 6.1.5) auch Kriterien zum Beenden bzw. Abbruch der Tests, die Prüfung der Testergebnisse und die Freigabe der Software (Abschnitt 6.1.6).

Bei der Entwicklung von Büroanwendungen ist die Teststrategie oft überschaubar und von der Entwicklungsumgebung vorgegeben. Bei Embedded-Software müssen Sie viele zusätzliche Aspekte beim Erstellen der Teststrategie berücksichtigen. Einige dieser Themen sind:

- Hardwareabhängigkeiten
- Echtzeit-Randbedingungen
- Einschränkungen bei Testumgebungen
  - Funktionalität
  - Verfügbarkeit

- Störungen
  - Bitfehler in digitalen Signalen
  - Unterbrechung von Signalen

6.1.1 Anforderungen an den Test

Oft werden vom Kunden viele Anforderungen an die Verifikation der Software gestellt. Gerade bei Embedded-Software, die in Investitionsgüter oder teure Massenverbrauchsgüter integriert wird, ist die Qualität der Software wichtig. Meist kann die Software nur mit erheblichem Kostenaufwand ausgetauscht werden, wenn Geräte bereits an Kunden ausgeliefert wurden.

Teststrategie mit dem Kunden abstimmen

Manche Kunden von Embedded-Systemen bevorzugen eine Abstimmung der Teststrategien mit den Lieferanten. Diese Kunden haben erkannt, dass eine Abstimmung der Teststrategien von Embedded-Systemen und den Systemen oder Anlagen, in die Embedded-Systeme integriert werden, hilft, Aufwände beim Test insgesamt zu reduzieren und damit (meist beim Kunden) Kosten zu sparen. In diesem Zusammenhang stehen auch die Forderungen der Kunden, Testprotokolle als Teil der Produktdokumentation zu liefern. Einige Kunden beschränken sich auch darauf, die Testprotokolle bei Bedarf einsehen zu können.

Viele Kunden machen teilweise detaillierte Vorgaben zur Verifikation von Embedded-Systemen. Diese Vorgaben betreffen dabei nicht nur die Verifikationsmaßnahmen an sich, sondern oft auch die Prozesse, also die Vorgehensweise beim Test insgesamt. Die Prozessvorgaben orientieren sich meist an Normen und Standards, wie den Sicherheitsnormen IEC 61508, ISO 26262 oder EN 50128, aber auch an Qualitätsstandards wie SPICE oder CMMI.

Qualitative Anforderungen an die Tests

Qualitative Anforderungen betreffen die Vorgehensweise beim Erstellen der Testunterlagen, die Vorgehensweise beim Testen und die Dokumentation dazu und werden gestellt an:

- Testspezifikationen
- Testfall-Implementierung
- Testanleitungen
- Testwerkzeuge
- Testpersonal

Quantitative Anforderungen

Aus diesen Vorgaben leiten sich dann auch quantitative Anforderungen an die Tests ab. Diese Anforderungen an den Umfang der Tests betreffen:

- Testabdeckung
- Testtiefe
- Regression

Die Testabdeckung macht eine Aussage darüber, in welchem Umfang der Quellcode getestet wurde (vgl. Kap. 1), die Testtiefe betrifft die Anzahl der Testfälle pro Funktion, und Regression bezieht sich auf die Anzahl der Wiederholungen, die dann wichtig sind, wenn Zeitbedingungen, Unterbrechungen oder Hardwareeinflüsse eine Rolle spielen.

6.1.2 Testvoraussetzungen

Machen Sie sich beim Erstellen der Teststrategie Gedanken, welche Vorbereitungen getroffen und welche Bedingungen erfüllt sein müssen, damit die Softwaretests erfolgreich und gemäß den Vorgaben durchgeführt werden können.

Testplanung

Wie alle Prozess-Schritte innerhalb des Entwicklungszyklus sollten Sie auch die Testphase planen. Auch wenn Sie für ein Embedded-System nur den Software-Modultest für eine einfache Funktion durchführen müssen, sollten Sie sich Gedanken über den Ablauf machen. Planen Sie eine komplette Testphase, müssen Sie viele Details im Zeitplan berücksichtigen:

- Überarbeiten und Freigeben aller nötigen Testspezifikationen, Anleitungen und Protokollvorlagen
- Prüfen und Freigeben aller Testwerkzeuge und Testumgebungen
- Abstimmen der Entwicklung der Testspezifikationen und Testfälle mit der Design- und Implementierungsphase
- Zusammenstellen der zu testenden Software (Baseline)
- Zeit zum Einrichten der Testumgebung vor dem Test
- Reservieren kritischer Ressourcen wie
  - Messgeräte
  - Testeinrichtungen
  - Lizenzen für die Testsoftware
- Schulen oder Einweisen des Testpersonals

- Nachbereiten der Tests
  - Review der Testprotokolle
  - Dokumentieren gemachter Erfahrungen
  - Initiieren von Prozessverbesserungen

Integrationsstrategie

Eine Integrationsstrategie ist wichtig, wenn sich Embedded-Software aus mehreren Modulen zusammensetzt. Theoretisch können Sie alle Module einzeln testen, dann Modul um Modul zusammenzusetzen und jeweils die teilintegrierte Software testen, bis die gesamte Software integriert ist.

In der Realität orientiert sich die Reihenfolge der Tests an den Gegebenheiten des jeweiligen Projekts. Ausgangspunkt ist ein fertiggestelltes Modul, das einem Modultest unterzogen wird. Danach ist zwar die Funktion des Moduls verifiziert, ob aber das Modul mit anderen in der gewünschten Weise zusammenarbeitet, stellt sich erst beim Integrationstest heraus, der genau dieses Zusammenwirken prüft. Die Module, die integriert und dem Integrationstest unterzogen werden, müssen vorher geprüft worden sein. Dadurch ist eine teilweise Parallelisierung der Testaktivitäten möglich.

Bei der Integration mehrerer Module ist es schwierig, eine überschaubare Testmenge zu bestimmen, welche die Anforderungen ausreichend prüft. Versuchen Sie, aus den Anforderungen und den Einsatzszenarien der Software entsprechende Testfälle für die teilintegrierte Software abzuleiten.

Für den Test einzelner Module gilt:

- Werden im Projekt Softwaremodule neu erstellt, sollten Sie diese separat testen.
- Sicherheitskritische Module müssen Sie separat testen. Bei sicherheitskritischen Modulen ist im Vergleich zu nicht-sicherheitsrelevanter Software ein höherer Testaufwand erforderlich. Oft müssen Sie dann auch zusätzliche Testwerkzeuge einsetzen, etwa zur Bestimmung der Testabdeckung.
- Bei bereits bestehenden und aus anderen Projekten übernommenen Modulen können Sie oft auf eine erneute komplette Prüfung des Moduls verzichten und sich auf einen Test der Schnittstellen und der tatsächlich verwendeten Funktionalität beschränken.
- Von Dritten beigestellte Software wird meist nicht separat getestet. Im Rahmen der Integration der Software sollten Sie aber die im Projekt benutzte Funktionalität prüfen. Auch sollten Sie die Dokumentation der Software auf Einschränkungen und bekannte Fehler

und die vorgeschlagenen Maßnahmen zur Umgehung dieser Fehler hin durchlesen.
- Hardwarenahe Software können Sie meist nur sinnvoll in Verbindung mit der Hardware testen, also auf dem Zielsystem oder einer dem Zielsystem möglichst ähnlichen Hardware.
- Sind zwei Softwaremodule über viele Schnittstellen miteinander verbunden, ist ein gemeinsamer Test dieser Module, also eine Teilintegration, weniger aufwendig als ein separater Test jedes Moduls.

In welchen Schritten Sie die Integration und den Integrationstest der Module durchführen, hängt vom Zusammenwirken der Module ab.

Hat das Embedded-System viele verschiedene Funktionen, die wenig miteinander zu tun haben (beispielsweise Diagnosefunktion und Regelungsfunktion), empfehle ich, zuerst die Module der Applikation zu integrieren, die für die wichtigste Funktion unbedingt gebraucht werden, dann die restlichen. Dabei sollten Sie in der Richtung des Datenflusses vorgehen, also zuerst das Datenaufnahmemodul mit dem Datenverarbeitungsmodul integrieren, danach das Datenausgabemodul hinzufügen.

Meist können Sie Applikationssoftware und die hardwarenahe Basissoftware (vgl. Kap. 3) getrennt integrieren und testen. Diese Arbeiten lassen sich auch parallel ausführen, wie viele Modultests übrigens auch. In welchem Umfang Sie parallelisieren, hängt meist von der Anzahl der verfügbaren Tester ab. Rechnen Sie aber auch mit weiteren Einschränkungen der Parallelisierung wie:

- Anzahl der Testsoftware-Lizenzen
- Anzahl der Testumgebungen, z. B. bei hardwarenahen Tests

Testobjekte

Aus der Integrationsstrategie können Sie ableiten und festlegen, wie viele und welche Testobjekte (Module, teilintegrierte Software) einem Test unterzogen werden müssen. Sie wissen dann auch, welche Integrationstests von welchen Modultests abhängen und damit nach diesen erfolgen müssen. Aus diesen Abhängigkeiten ergibt sich eine Sequenz von Tests. Je mehr Ressourcen (Testumgebungen und Testpersonal) Sie haben, desto mehr können Sie die in der Sequenz enthaltenen parallelen Pfade nutzen, um Zeit zu sparen und die Testdauer zu minimieren.

Testspezifikationen

Unabhängig vom jeweiligen Testobjekt müssen Sie die Testspezifikation vor Beginn des Tests fertiggestellt haben. Zur Fertigstellung gehört auch, dass diese Spezifikation von den Entwicklern und Testern einem Review unterzogen wurde. Beim Test von Embedded-Software müssen Sie viele Randbedingungen beachten (Hardware, Zeitverhalten, Testumgebung), so dass die Gefahr besteht, beim Spezifizieren den einen oder anderen Aspekt zu übersehen.

Bauen Sie die Testspezifikationen deshalb, wenn möglich, nach einem einheitlichen Schema auf. Beschreiben Sie zu Beginn die Testobjekte und deren Testumgebungen. Denken Sie dabei auch an Kleinigkeiten, wie spezielle Kabel und Geräteeinstellungen. Nur weil Sie beim Konfigurieren einer Testumgebung ein Oszilloskop in einem bestimmten Betriebsmodus vorgefunden haben, können Sie nicht davon ausgehen, dass dies auch beim nächsten Mal so sein wird.

Wenn Sie die einzelnen Testfälle beschreiben, stellen Sie für jeden Testfall diese Informationen bereit:

- Vorbedingungen:
  Was muss vor dem Test alles eingestellt oder verändert werden?
- Nachbedingungen:
  Was muss nach dem Test alles zurückgesetzt oder in den vorherigen Zustand versetzt werden?
- Soll-Ergebnis:
  Welches Ergebnis soll der Test liefern, um als »bestanden« gewertet zu werden?
- Link zu den Anforderungen, die mit dem Testfall verifiziert werden.

Risiken für Testhardware und Testumgebung

Klären Sie auch mögliche Risiken für die Testhardware und die Testumgebungen ab. Wenn Sie etwa Modultests auf der Zielhardware durchführen, muss dazu die Software oft ins Flash geladen werden oder Daten in EEPROMs geschrieben werden. Diese Bausteine lassen nur eine beschränkte Anzahl von Schreibzyklen zu. Zwar wird die Hardware so ausgelegt, dass die maximale Zahl der Schreibzyklen später im realen Betrieb nicht überschritten wird, doch im Testlabor herrschen andere Bedingungen. Rechnen Sie sicherheitshalber nach, in welchem Umfang die Tests die Testeinrichtung belasten und abnutzen.

Schalten Sie in der Testeinrichtung Signale und Spannungen, dann prüfen Sie, ob durch eine falsche Reihenfolge von Schaltvorgängen Schäden an der Ausrüstung verursacht werden könnten. Oft muss eine bestimmte Verbindung zuerst aufgetrennt werden, bevor ein Kontakt

mit einer Testspannung verbunden wird. In diesem Fall sollten Sie in der Dokumentation entsprechende Gefahrenhinweise angeben.

Testfälle und ihr Bezug zu den Anforderungen

Listen Sie bei der Beschreibung der einzelnen Testfälle alle Anforderungen auf, die mit den Testfällen verifiziert werden sollen. Tragen Sie die Testfälle auch in die Anforderungsverfolgungsmatrix ein. Dann können Sie leicht feststellen, ob die Testfälle alle Anforderungen abdecken oder ob noch Testfälle fehlen. Die Anforderungen sind auch eine unabdingbare Voraussetzung zum Beurteilen der Wichtigkeit eines Tests und damit eine Voraussetzung, um die Priorität eines Tests festzulegen.

Sicherheitsanforderungen fallen damit in die Kategorie »unbedingt durchführen«.

Wenn es Anforderungen gibt, die sich auf optionale Features beziehen, müssen diese möglicherweise nicht immer getestet werden, und Tests, die diese Anforderungen verifizieren, können Sie in der Priorität zurückstufen.

Testprotokoll

Meine Erfahrungen haben gezeigt, dass Sie bei Tests von Embedded-Software möglichst viele Details protokollieren sollten. Nur so können Sie aus den Testergebnissen die richtigen Schlüsse ziehen.

Zweitrangig ist, ob Sie die Testergebnisse automatisch, manuell oder in einer Mischung aus beiden Protokollarten festhalten.

Diese Informationen sollten mindestens im Testprotokoll zu finden sein:

- Informationen über das Testobjekt
  - Bezeichnungen der getesteten Softwaremodule
  - Versionsnummern der getesteten Software oder die Bezeichnung der Baseline des Konfigurationsmanagementsystems, mit dem die Software verwaltet wird
- Informationen über die Testumgebung
  - Welche Testumgebung wurde verwendet?
  - spezielle Einstellungen der Geräte, soweit diese nicht in der Testspezifikation dokumentiert sind
- Informationen über die Testprogramme
  - Testsoftware inkl. Version oder ggf. Bezeichnung der Baseline
- Informationen über die Testdurchführung
  - Datum, ggf. Ort
  - Name der Person, die den Test durchgeführt hat
  - Ort der Ablage der Testprotokolle (URL)

- Informationen zu jedem einzelnen Test
  - Test-ID
  - Ist-Ergebnis
  - Bewertung
  - Bemerkungen

Bewertung der Testergebnisse

Zur Bewertung verwenden Sie *bestanden*, *nicht bestanden* oder *nicht durchgeführt*. Sind Sie nicht sicher, ob der Test bestanden wurde, weil das Testergebnis zwar vom Soll-Ergebnis abweicht, andererseits die Funktion Ihrer Ansicht nach i.O. ist, dann kommentieren Sie das Testergebnis in Feld »Bemerkungen« entsprechend.

Gerade die Spalte »Bemerkungen« ist in einem Testprotokoll wichtig. Die dort notierten Auffälligkeiten liefern Hinweise auf:

- Lücken in der Testspezifikation
- unvollständige Testbeschreibung
- nicht eindeutig beschriebene Soll-Ergebnisse
- allgemeine Mängel des Tests wie:
  - Zeitverhalten ist von Test zu Test verschieden.
  - fehlerhafte oder abgebrochene Datenübertragungen von Code auf das Testsystem

Besonders beim Testen sicherheitsrelevanter Software dient das Testprotokoll auch zum Nachweis, dass Sie alle erforderlichen Anstrengungen unternommen haben, um die Software zu prüfen. Dies könnte bei Fragen der Produkthaftung wichtig werden.

Testfälle

Automatische Tests

Erstellen Sie Testfälle für automatische Tests, so unterziehen Sie diese Testsoftware einem Review. Eventuell ist auch ein Kurztest sinnvoll, mit dem der Testablauf geprüft wird. Idealerweise ersetzen Sie die zu testenden Module durch leere Dummy-Module und weisen damit nach, dass alle Tests als Ergebnis *nicht bestanden* liefern.

Manuelle Tests

Leider lassen sich nicht alle Softwaretests automatisieren. Spielt die Hardware eine Rolle, wird die Automatisierung schwierig. Die Schnittstellen des Prozessors können Sie oft mit einer steuerbaren Testumgebung prüfen, welche die Schnittstellen mit geeigneten Testsignalen beschickt. Wollen Sie Software testen, die Hardwarefehler, etwa im RAM, erkennen soll, bleibt Ihnen nur der Einsatz eines Emulators, Debuggers oder anderer zusätzlicher Werkzeuge, um in den Programmablauf einzugreifen.

Beschreiben Sie den Ablauf dieser manuellen Tests detailliert. Falls diese Werkzeuge über Skripte steuerbar sind, nutzen Sie diese Möglichkeit. Je mehr Sie implementieren und automatisch ausführen, desto weniger anfällig ist der Test für Fehlbedienungen und desto schneller ist er durchgeführt. Dazu gehört auch, Messgeräte nicht von Hand einzustellen, sondern die Einstellungen ins Messgerät zu laden und auch die Messergebnisse, wenn möglich, auszulesen und automatisch auszuwerten und zu speichern.

Robuste Tests

Achten Sie darauf, dass Sie bei Tests, an denen Hardware beteiligt ist, die Testsignale und deren Timing so wählen, dass Sie einerseits die Funktion des Testobjekts eindeutig auf Korrektheit prüfen können, andererseits aber an die Testsignale nicht zu hohe Anforderungen stellen müssen. Das würde die Testumgebung erheblich verteuern. Je robuster die Tests, desto besser.

Ist ein Test nicht robust genug und liefert unterschiedliche Ergebnisse, obwohl die getestete Funktion korrekt ist, dann prüfen Sie alle Vorbedingungen des Tests. Meist waren diese bei der Testfallentwicklung nicht in vollem Umfang bekannt oder haben sich zwischenzeitlich geändert.

6.1.3 Testumgebung

Oft wird für einen Test die Testumgebung je nach Verfügbarkeit aus verschiedenen Werkzeugen und Geräten zusammengestellt. In einer solchen Umgebung reproduzierbare und vergleichbare Testergebnisse zu erzielen ist schwierig.

Auch für eine Testumgebung gilt: Beschreiben Sie diese ausführlich. Dazu gehören auch Gerätetypen, Hardwareversionen und Vorgaben zu Software-Werkzeugen inklusive deren Versionsnummern, Konfigurationsdateien.

Bei Embedded-Software spielen spezielle Hardwareschnittstellen, Sensoren und Aktoren eine wichtige Rolle. Versuchen Sie, die Originalteile zu bekommen. Manchmal ist dies nicht möglich, da die Original-Aktoren zu teuer sind, zu groß für das Labor oder weil sie ohne zusätzliche Geräte (etwa zur Kühlung) nicht betrieben werden können. In diesen Fällen ist beim Test besondere Vorsicht geboten. Versuchen Sie, eine möglichst gute elektrische Nachbildung der Sensoren oder Aktoren zu erhalten. Mit einem Ersatzwiderstand statt der Spule eines Ventils kann sich die Regelungs-Software ganz anders verhalten als erwartet. Die fehlende Induktivität des Ersatzwiderstands kann zu einem anderen zeitlichen Verhalten des Steuerstroms und damit der Regelgröße führen.

Spezielle Testumgebungen für Embedded-Software

Zum Testen von Embedded-Software werden unterschiedliche Testumgebungen eingesetzt.

Für den Test der Applikationen bieten sich Simulationsumgebungen an. Damit können insbesondere auch Regelsysteme ausführlich getestet werden. Für einfache Applikationen können Sie ein Simulationsprogramm natürlich selbst schreiben. Bei komplexeren Anwendungen sollten Sie auf Simulationsumgebungen zurückgreifen, wie etwa MATLB/Simulink. Bauen Sie die Simulationsumgebungen bei Regelungen so auf, dass Sie diese als geschlossene Regelschleife (*closed loop*) und als offene Regelschleife (*open loop*) betreiben können. Die offene Regelschleife können Sie mit Messdaten aus der realen Umgebung beschicken und so die Reaktion der Software prüfen. In einer Simulationsumgebung wie MATLAB/Simulink können Sie Softwaremodelle prüfen, aus denen Sie später Code generieren (Model in the Loop = MIL). Wenn Sie den generierten Quellcode statt des Modells einbinden, können Sie in der Simulationsumgebung dessen Verhalten mit dem des Modells vergleichen (Software in the Loop = SIL, nicht zu verwechseln mit dem Safety Integrity Level, der genauso abgekürzt wird). Damit prüfen Sie gleichzeitig die Qualität des generierten Codes. Speziell wenn der Code mit anderer Rechengenauigkeit arbeitet als das Modell, ist dieser Vergleich interessant. Sie können statt generiertem Code auch Handcode in diese Simulationsumgebung einbinden und so testen.

Mit MIL und SIL können Sie komplexe Einsatzszenarien für die Software testen, aber auch einzelne Funktionalitäten der Software untersuchen.

Eine Reihe von Herstellern bietet auch Testumgebungen an, die auf Embedded-Software zugeschnitten sind. Sie können aber auch auf Open Source zurückgreifen. Für C-Code bietet sich hier das Werkzeug CUnit an (vgl. Abschnitt 6.2). Diese Testumgebungen sind für den Modultest geeignet, wobei einige Werkzeuge sowohl den Test auf dem Entwicklungsrechner (PC) als auch auf dem Zielsystem unterstützen.

Gerade bei hardwarenaher Software, wie Schnittstellentreibern, können einige Funktionalitäten nur in Verbindung mit der Hardware getestet werden. Dazu muss die Schnittstellenhardware mit Signalen stimuliert werden. Die Testaufbauten werden meist aus handelsüblichen Geräten zusammengestellt. Trotzdem sind oft einige Teile konfektioniert, etwa spezielle Kabel. Sorgen Sie deshalb dafür, dass diese Testaufbauten und ihre Verkabelung vollständig dokumentiert werden. Geräte werden ausgeliehen oder dringender woanders gebraucht und der Aufbau zerlegt. Ohne Dokumentation müsste der Aufbau aus

dem Gedächtnis rekonstruiert werden. Speziell angefertigte Kabel sollten Sie detailliert beschreiben, damit sie neu hergestellt werden können, wenn sie nicht mehr auffindbar sind.

Nach der Integration der Software und der Hardware können Sie das Embedded-System in einem Testbett prüfen, das alle Eingangssignale liefert und alle Ausgangssignale aufnimmt. Dabei simuliert das Testbett das Zielsystem, in welches das Embedded-System integriert wird. Ist dieses Testsystem programmierbar und berechnet aus den Ausgangssignalen und einem Modell des Gesamtsystems die Eingangssignale für das Testobjekt, spricht man auch von einem Hardware-in-the-Loop-System (HIL).

Testwerkzeuge

Neben den üblichen Software-Entwicklungswerkzeugen wie Compiler und Debugger werden für die Prüfung von Embedded-Software weitere Werkzeuge eingesetzt.

Codeanalyse-Werkzeuge

Der Code selbst wird mit Analysewerkzeugen untersucht. Diese Werkzeuge können nicht nur die Syntax des Codes prüfen, wie ein Compiler, sondern helfen, Codezeilen zu identifizieren, die zu möglichen Fehlern im Verhalten der Software führen können. Dazu zählen Wertebereichsüberschreitungen und Datenverluste bei Zuweisungen, Fehler bei der Adressrechnung und das Auffinden von Rekursionen.

Werkzeuge für den Modultest

Für den Modultest kommen neben Werkzeugen, die helfen, Testvektoren zu bestimmen, auch solche zum Einsatz, welche die Testabdeckung bestimmen. Damit erhalten Sie eine wichtige Aussage darüber, in welchem Umfang und in welcher Tiefe der Code durch die vorhandenen Testfälle geprüft wird.

Weitere Werkzeuge unterstützen den Test des Codes auf dem Zielsystem (Lade- und Kommunikationssoftware).

Signalquellen und Analysegeräte

Zu einer Testumgebung für hardwarenahe Software gehören neben der Zielhardware auch Signalgeneratoren und Anzeigegeräte, wie Multimeter, Oszilloskop, Logikanalysator oder Busanalysator.

Oft scheitert der Softwareentwickler an so einfachen Dingen wie der Polarität von Signalen. Mit einem Multimeter wissen Sie sofort, welche Spannungen wo anliegen. Mit Schaltplan oder ausführlicher Hardwarebeschreibung wird dann auch schnell erkennbar, wie der Signalwert an die Software weitergereicht wird.

Den zeitlichen Verlauf von Ein- und Ausgangssignalen des Embedded-Systems können Sie mit einem Oszilloskop untersuchen.

Logikanalysatoren helfen Ihnen weiter, wenn Sie den Datenfluss auf parallelen Schnittstellen beobachten wollen oder den zeitlichen

Zusammenhang zwischen mehreren digitalen Signalen, welche die Software auswertet, untersuchen müssen.

Busanalyse-Werkzeuge verwenden Sie, wenn nicht klar ist, wann welche Datenpakete vom Embedded-System über einen Datenbus geschickt werden oder empfangen werden. Für Ethernet ist eine Vielzahl freier Werkzeuge verfügbar. Wollen Sie den Datenverkehr auf Bussen wie CAN oder FlexRay beobachten, werden Sie für gute Werkzeuge einige Tausend Euro investieren müssen.

Dokumentieren Sie in jedem Fall die oft umfangreichen Einstellungen der Messgeräte und Analysatoren, die für die jeweiligen Tests relevant sind. Wenn ein Gerät ausgeliehen wurde, kommt es meist in anderer, für die Tests nicht verwendbarer Konfiguration wieder zurück. Falls Sie Ihre Einstellungen im Gerät speichern, auf Datenträger kopieren oder an den Bedienrechner der Testeinrichtung übertragen können, nutzen Sie diese Möglichkeit unbedingt. Manche Geräteeinstellungen werden aus Versehen gelöscht, manchmal auch bei einem Rücksetzen des Geräts in den Lieferzustand, etwa im Rahmen einer Kalibrierung.

Denken Sie daran, dass Messgeräte auch regelmäßig kalibriert werden müssen. Nur dann sind die abgelesenen Werte verlässlich. Als Softwareentwickler müssen Sie sich darum oft nicht selbst kümmern, aber bei der Vorbereitung von Tests sollten Sie darauf achten, dass die verwendeten Geräte alle kalibriert sind. Diese Information können Sie meist Aufklebern auf den Geräten entnehmen.

Prüfen der Testumgebung

Soweit vorhanden, führen Sie Tests, mit denen Sie die Testeinrichtung auf ihre Funktionstüchtigkeit prüfen können, vor Beginn der Software- oder Systemtests durch. Viele Fehler gehen oft auf Fehler in der Testumgebung zurück und nicht auf Fehler in der Software oder der Implementierung des Tests selbst.

Ich empfehle, die Testeinrichtung vor jedem Test eines neuen Release der Software zu prüfen. Führen Sie häufig Tests durch, ist meist die Prüfung der Teile der Testeinrichtung ausreichend, die beim letzten Mal Schwächen gezeigt haben.

Kümmern Sie sich auch um Wartung und Support der Testsoftware und Testhardware. Oft sind Testgeräte auch geliehen, oder die Lizenz der Testsoftware läuft nur eine begrenzte Zeit. Wenn kurz vor der Auslieferung der Software die Testeinrichtung nicht benutzt werden kann, weil eine Lizenz ungültig geworden ist, ist dies äußerst ärgerlich.

6.1.4 Testvorbereitung

Wenn Sie Tests durchführen wollen, prüfen Sie, ob alle Voraussetzungen für die Testdurchführung erfüllt sind:

- Ist die Testumgebung funktionstüchtig?
  Sind alle Komponenten (Geräte, Kabel, Software usw.) vorhanden und einsatzbereit?
- Ist die Software, die getestet werden soll, für die Tests freigegeben?
  Wurde verifiziert, dass in der Software alle Anforderungen umgesetzt sind, und wurden bereits bekannte Fehler und Einschränkungen dokumentiert?
- Sind die Testspezifikationen und Protokollvorlagen freigegeben?
- Wurde der Testumfang gemäß der Teststrategie festgelegt?
  Je nach Aufgabenstellung sind die Testumfänge verschieden. Für interne Versuche im Zielsystem sind vermutlich weniger Tests erforderlich als für die Freigabe zur Serienproduktion.

Legen Sie fest, welche Tests zu welchem Zweck durchgeführt werden müssen und dokumentieren Sie dies in der Testspezifikation. Markieren Sie bei jedem Test, für welche Teststufe dieser durchgeführt werden muss.

Zur Vorbereitung des Tests gehört auch das Ausfüllen eines Protokolls, in dem die Testdurchführung dokumentiert wird. Was alles im Testprotokoll festgehalten werden sollte, finden Sie in Abschnitt 6.1.2.

6.1.5 Testdurchführung

Wenn Sie einen Softwaretest durchführen, verwenden Sie die aktuellen und freigegebenen Unterlagen. Halten Sie sich an die dokumentierten Arbeitsschritte. Dokumentieren Sie das Ergebnis eines einzelnen Testschritts eindeutig mit *bestanden* oder *nicht bestanden*. Nicht verwenden sollten Sie Häkchen, Striche, Kreuzchen oder dergleichen. Fehlinterpretationen bei der Auswertung der Protokolle sind die Folge.

Notieren Sie Messergebnisse, damit später festgestellt werden kann, wo innerhalb der Grenzen des von der Testspezifikation vorgegebenen Soll-Ergebnisses das Testergebnis liegt.

Markieren Sie Tests, die Sie ausgelassen haben, mit *nicht durchgeführt*. Notieren Sie die Gründe dafür.

Wenn Sie feststellen, dass sich der Test so nicht durchführen lässt, ist das bei Embedded-Software, insbesondere wenn Hardware und Signalquellen verwendet werden, keine Seltenheit. Oft wird etwas übersehen, oder manchmal wurde auch eine Einstellung an den Geräten oder

der Hardware verändert: Ein Jumper sitzt nicht korrekt, oder eine Adresse wurde nicht korrekt eingestellt.

Mein Tipp:

Ändern Sie die Einstellungen ab, so dass Sie den Test durchführen können, und dokumentieren Sie Ihre Änderungen im Testprotokoll. Werten Sie nach dem Test das Protokoll aus und optimieren Sie die Testspezifikationen.

Dokumentieren Sie alle Auffälligkeiten. Notieren Sie auch zusätzliche oder geänderte Einstellungen an Instrumenten, die Sie vorgenommen haben. Keine Spezifikation ist vollständig, weder die Testspezifikation noch die Vorgaben, die der Softwareentwickler hatte.

6.1.6 Ende des Tests

Kriterien für das Testabbruch

Legen Sie fest, wann ein Test abgebrochen wird. Eine Testsequenz bis zum Ende durchzuführen, wenn die Fortsetzung der Testsequenz nicht mehr sinnvoll ist und die Testergebnisse keine weiteren verwertbaren Informationen liefern, ist unrentabel und ineffizient. Mögliche Abbruchkriterien sind:

- Anzahl der fehlgeschlagenen Tests
- Fehlschlag eines (sicherheits-)kritischen Tests

Testauswertung

Werten Sie am Ende der jeweiligen Testphase (Modultest, Software-Integrationstest, HW-/SW-Integrationstest, Systemtest) die Testprotokolle auswerten.

Automatisierte Protokollbewertung

Protokolle von automatisierten Tests können meist auch automatisch ausgewertet werden. Ob sich die automatisierte Auswertung lohnt, hängt vom Aufwand im Verhältnis zu der Anzahl der durchzuführenden Tests ab.

Protokollbewertung mit Checklisten

Erstellen Sie eine Checkliste, in der Sie festlegen, wonach Sie in den Testprotokollen schauen. Die Testauswertung bekommt dadurch den Charakter eines Reviews. Dokumentieren Sie die Ergebnisse der Auswertung.

Die Ergebnisse der Auswertung sind eine wichtige Quelle zur Verbesserung der Testspezifikationen, der Tests und der Testumgebung.

Auswertung fehlgeschlagener Tests

Sind Tests fehlgeschlagen, sollten Sie zuerst die Ist-Ergebnisse mit den Soll-Ergebnissen vergleichen. Liegt kein Spezifikationsfehler vor und auch kein Fehler in der Testumgebung, haben Sie höchstwahrscheinlich einen Softwarefehler gefunden. Zur Sicherheit sollten Sie überprüfen, wie die ursprüngliche Anforderung an die Software lautet und ob diese und der Testfall zusammenpassen. Die Anforderungen, welche mit dem Testfall verifiziert werden sollen, finden Sie über die Anforderungsverfolgungsmatrix. Spiegelt der Testfall die Anforderung korrekt wider, kann aus dem Testergebnis oft schon geschlossen werden, welcher Fehler in der Software vorliegt.

Wurden Tests nicht durchgeführt, prüfen Sie die Begründungen. Tests können etwa wegen fehlgeschlagener anderer Tests übersprungen worden sein.

Wurden Fehler in der Testumgebung gefunden und behoben, müssen Sie überlegen, ob Sie nur den Test neu durchführen, bei dessen Ausführung Sie den Fehler in der Testumgebung gefunden haben oder ob Sie alle Tests nochmals durchführen. Die Änderung der Testumgebung kann Auswirkungen auf andere Tests haben.

Am Ende der Auswertung sollte Sie auflisten, welche Tests nach Behebung der gefundenen Fehler in Software, Testumgebung und Testspezifikation nochmals durchgeführt werden.

Teststatistik

Führen Sie eine Statistik und erfassen Sie für jede Testphase, wie viele Tests jeweils *bestanden*, *nicht bestanden* oder *nicht durchgeführt* wurden. Dokumentieren Sie auch die Werte aus der Code-Überdeckungsanalyse der Tests. Dies hilft, einen Eindruck von der Reife der Software und der Tests zu bekommen.

Freigabe

Nach dem Auswerten der Testprotokolle wird die Software freigegeben. Diese kann am einfachsten im Review-Protokoll der Testergebnisse dokumentiert werden. Wichtige Freigaben, etwa bei Übergabe an den Kunden oder an die Serienfertigung, sind aufwendiger. Dabei prüfen Sie meist zusammen mit dem Qualitätssicherungsbeauftragten, ob alle festgelegten Freigabekriterien erfüllt sind. Neben dieser schriftlichen Bestätigung ist die Liste der bekannten Mängel und Einschränkungen das wichtigste Dokument. Dieses unterziehen Sie im Rahmen der Freigabe einem Review. Bei diesem Review muss auch jemand teilnehmen, der die Auswirkungen der Einschränkungen und Mängel auf die weiteren Tests oder Arbeiten im Projekt beurteilen kann.