Trend Guide

Embedded Quality

MICRO CONSULT

No.1 in Developer Training

Inhalt

Embedded Quality

Vorwort

Klaus-Peter Rosenthal
Product Manager Software Entwicklung
E-Mail k.rosenthal@microconsult.de

Liebe Leser,

wo liegt unsere Zukunft, fragen sich die deutschen Unternehmen nach dem Ende des High-Tech-Booms der 90er Jahre. Unentschlossene, eher hilflose Antworten geben auch führende Analysten wie Gartner. Deren aktuelle Vorhersagespanne für 2003 reicht beispielsweise im Halbleitermarkt von 12 Prozent Umsatzwachstum bis 8 Prozent Rückgang. Dazwischen liegen wirtschaftlich gesehen wirklich Welten.

Positive Impulse kommen momentan eher aus dem asiatisch-pazifischen Raum und dort vor allem aus dem Reich der Mitte: So gut wie alle internationalen Chip-Hersteller sind mittlerweile in China mit Fabriken vertreten. Dies nicht nur wegen guter Absatzperspektiven. So lockt das Niedriglohn-Niveau Deutsche, Amerikaner, Japaner und Taiwanesen gleichermaßen. Frühere Qualitätsvorbehalte gegenüber den Billigproduzenten treten auch bei technisch anspruchsvollen Produkten immer mehr in den Hintergrund.

Wo entsteht die Wertschöpfung zukünftig im vergleichbar kleinen Deutschland, das über lange Zeit mit guter Qualität gute Margen erzielen durfte?

Zählt das eigene Know-how im internationalen Wettbewerb nicht mehr? Wir meinen doch und denken hier konkret an die Embedded Welt.

In Zeiten, in denen der Wert der Hardware rasant auf Talfahrt geht, Embedded Systeme aber gleichzeitig weltweit die unterschiedlichsten Gerätemärkte erobern, entsteht die Wertschöpfung klar aus der Software. Oder sie könnte daraus entstehen, denn noch fehlt es in vielen deutschen Entwicklungsunternehmen an einer umfassenden Qualitätsstrategie.

Dieser Trend Guide wird Ihnen Impulse geben, über Qualität und ihre Merkmale, über Entwicklungsprozesse und -methoden nachzudenken. Er wird Ihnen dabei helfen, den individuellen Weg für Ihr Unternehmen zu finden.

Wollen Sie auch unsere weiteren Trend Guides wie „Embedded Future“ kostenlos beziehen? Dann schicken Sie uns eine E-Mail an **trendguide@microconsult.de** oder ein Fax an **+49 (0) 89 / 45 06 17-18.**

Und nun viel Spaß beim Lesen!

Klaus-Peter Rosenthal

Trend Spots

Fatale Folgen

„Immer mehr technische Pannen, Rückrufaktionen, verschobene Markteinführungen. Schlampt die deutsche Autoindustrie bei der Qualität?“

Quelle: Handelsblatt, 25.09.2002

Ein kleines Unterprogramm wurde der Ariane 5 zum Verhängnis: Für den ausschließlichen Einsatz am Boden bestimmt, löste es im Flug eine unheilvolle Fehlerkette aus.

Horrende Haftung

Ein Software-Haus kann, wenn überhaupt, nur dann der Produkthaftung entgehen, wenn es nachweisen kann, dass nach dem Stand von Wissenschaft und Technik Software-Fehler nicht erkannt werden konnten. Dazu muss eine einwandfreie Qualitätssicherung, insbesondere durch eine lückenlose Dokumentation, nachgewiesen werden.

Quelle: RA Dr. Rupert Vogel, Bartsch und Partner Rechtsanwälte, 2003

Wertvolle Werkzeuge

Der Markt für Tools im Bereich verteilter, automatisierter Software-Qualität wird bis 2006 jährlich um 20 Prozent anwachsen. Dies resultiert aus dem wachsenden Bedarf an Qualität in immer komplexeren, verteilten Anwendungsumgebungen.

Quelle: Gartner, Juli 2002

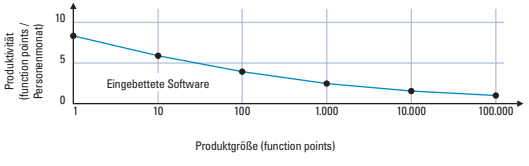
■ Heutige Herausforderungen

Die Komplexität von Embedded Projekten steigt durch mehr Funktionalität, größere Variantenvielfalt, einen hohen Grad an Vernetzung und kürzere Reaktionszeiten.

Wenn der Test das Maß aller Dinge ist, dann steht der Fehler im Zentrum und nicht die Qualität.

Gute Qualität ist teuer, schlechte noch viel mehr.

Qualitätsfalle Software



Quelle: BMBF-Verbundprojekt EQUAL 2002

Mit steigendem Software-Anteil fallen die Produktivität und Qualität eines Embedded Systems.

Die Herausforderung: Qualität als Strategie

Wenn das Navigationssystem im Automobil ausfällt, ist das ärgerlich. Wenn der Motor bei 200 Stundenkilometern versagt, vielleicht tödlich. In den letzten Monaten rückte eine ganze Industrie ins Rampenlicht der Medien: auf den Glanz neuer Edelkarossen fiel der Schatten frappierender Qualitätsmängel. Laut Kraftfahrt-Bundesamt hat sich die Zahl der Rückrufaktionen von Automobilherstellern in den letzten fünf Jahren mehr als verdoppelt. Mittlerweile fragen sich mehr und mehr Kunden, ob ihr Vertrauen in die Automobilbranche noch gerechtfertigt ist.

Speziell die Luxusklasse ist vollgestopft mit Elektronik, die an Komplexität mittlerweile schwer zu überbieten ist. Und gerade in diesem Bereich verlagert sich die Produktentwicklung verstärkt von der Hardware hin zur Software. So hängt die Qualität technischer Produkte zunehmend von Embedded Systemen mit hoch integrativer Software ab. Dies nicht nur im Automotive-Bereich, sondern in praktisch jeder Branche.

Zur Qualitätsproblematik trägt nicht nur die Komplexität der Systeme bei, sondern auch der hohe Verteilungsgrad.

Es scheint, als könne das Qualitätsbewusstsein der Entwicklungsunternehmen nicht mit dem Tempo der technologischen Neuerung und wachsenden

Komplexität Schritt halten. Je größer der Software-Anteil ist, desto schwieriger lässt sich hohe Produktqualität erzielen. Dies mit fatalen Folgen gerade bei kritischen Embedded Systemen. Die Palette reicht von Reputationsverlust und hohen Haftungskosten über Umweltschäden bis hin zur Gefährdung des menschlichen Lebens.

Klare Linie finden

Doch woraus resultieren die Mängel von Embedded Systemen konkret? Nicht zuletzt liegt die Misere in den mangelnden Ausbildungssystemen von Universitäten und Fachhochschulen begründet. Die heutigen Defizite bestehen vor allem in den Kenntnissen, Qualität umfassend zu beschreiben. Am Beginn eines Projektes fehlen häufig klare Entscheidungen, welche Eigenschaften für ein Produkt wichtig und welche verzichtbar sind. Meist beschränkt sich die Spezifikation auf funktionale Anforderungen, wobei Qualität in vielen Fällen einen ausgeprägt nichtfunktionalen Charakter besitzt. Selbst dem Auftraggeber ist oft nicht bewusst, dass er diesen mit der Anforderungsspezifikation konkret einfordern muss.

Wenig weiß man auch, wie Qualität zu erreichen und zu kontrollieren ist. Zudem mangelt es an dem Bewusstsein, dass sie Ressourcen kostet, ihr Feh-

len aber wesentlich mehr zu Buche schlägt. So kann sich ein spät entdeckter Fehler aufgrund mangelnder Qualitätssicherung immens auf Aufwand und Termine auswirken.

Ein Projektleiter trifft heute Entscheidungen über qualitätssichernde Maßnahmen meist aus dem Bauch. Selbst wenn er richtig läge, kann er weder seinen Chef, noch das Team adäquat überzeugen, warum ein bestimmtes Tool oder eine neue Methode wirklich Früchte tragen soll. Auch die Rahmenbedingungen eines Projektes lassen sich mit der eigenen Intuition nur unzureichend konkretisieren.

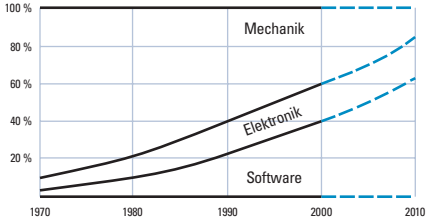
Qualität als Kernkompetenz

Qualitätsziele sollten auf Basis der Firmenstrategie festgelegt werden und damit durch die Produkte und Märkte bestimmt sein, die das Unternehmen bedienen will. Schließlich resultiert daraus ein Kundenpotenzial mit spezifischen Anforderungen.

Letztendlich muss das Management Qualität als Wettbewerbsfaktor und Kernkompetenz in Zeiten begreifen, in denen sich ein Software-Haus durch Funktionalität alleine nicht auf dem Weltmarkt behaupten kann. Geschäftsprozesse beeinflussen Entwicklungsprozesse. Je größer das Qualitätsbewusstsein der Führung ist, desto mehr ist es auch

in den Köpfen der Mitarbeiter verankert. Ein gemeinsames Verständnis von Entwicklungsabläufen ist zentral, gerade wenn ein Unternehmen in vielen Bereichen aktiv ist. Ist es nicht vorhanden, trägt das Endprodukt eher zufällige Qualitätsmerkmale und wird in heutigen Märkten langfristig scheitern. Die unternehmerische Verantwortung entbindet den einzelnen Entwickler jedoch nicht von der Eigeninitiative. Letztendlich ist er dem zukünftigen Produkt am nächsten und kann damit den Wandel zu mehr Qualität selbst anregen und mitgestalten.

Software als Wertschöpfungsfaktor



Zusammensetzung der Investitionen in Produktionsanlagen

Quelle: Glas/Kohen/McKinsey 2002

Beispiel Investitionen in Produktionsanlagen: Bei gleichbleibendem mechanischem und steigendem elektronischem Kostenanteil lässt sich über die Software die größte Wertsteigerung erzielen.

Qualitätsmerkmale: Weniger ist mehr

Wer Entwickler nach ihrem Verständnis von Qualität befragt, wird wahrscheinlich höchst unterschiedliche Antworten erhalten. Missverständnisse bestehen nicht nur bezüglich des Begriffes an sich, sondern betreffen auch die Bewertung und Gewichtung ihrer Merkmale. Dieser Umstand kann das Unternehmen schnell teuer zu stehen kommen.

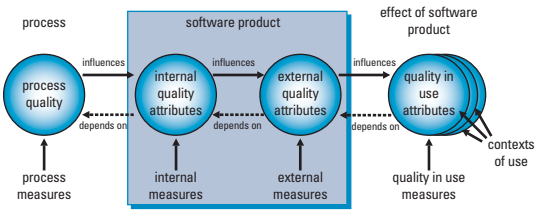
Der Qualitätsbegriff beinhaltet heute zwei wesentliche Aspekte:

- **Qualität des Prozesses:** Wie gut sind die Entwicklungsabläufe, mit denen Embedded Software erstellt wird?
- **Qualität des Produktes als Resultat der Prozesse:** Sie wird vor allem von der Eignung für den späteren Einsatz bestimmt.

Je komplexer Systeme sind, desto mehr wird sich eine hohe Prozessqualität auf die Produktqualität auswirken. Die sinnvolle Gestaltung der Prozesse schafft Termin- sowie Budgettreue und bestimmt letztendlich die Kundenzufriedenheit. Prozessqualität bewährt sich auch in der Akquise-Politik: Wer seinem Interessenten den Weg zum Ziel verständlich machen und Zwischenziele stecken kann, schafft Vertrauen. Gerade bei kritischen Embedded Systemen (Reputationsverluste, Umweltschäden, Verletzte oder Tote) rückt dagegen die Produktqualität verstärkt in den Vordergrund.

Aus den Anforderungen an Prozesse und das Produkt lassen sich interne und externe Qualitätsmerkmale ableiten. Die internen werden direkt am System gemessen, beispielsweise ob der Leiterbahnabstand auf einer Platine ausreichend groß ist. Externe Merkmale, wie beispielsweise Zuverlässigkeit oder Funktionalität, sind dagegen durch die äußere Umgebung bestimmt.

Axiom des Software-Engineering



Quelle: Fenton, Norman E.: Software Metrics, 1991

Norman E. Fenton konstatiert, dass ein direkter Zusammenhang zwischen internen Strukturen und externen Qualitätsmerkmalen besteht.

Die scheinbare Trennung von interner und externer Qualität relativiert das „Axiom des Software-Engineering“ von Norman E. Fenton. Danach bildet eine gute interne Qualität die Voraussetzung für eine gelungene externe. Dies umso mehr, je länger

die Lebensdauer einer Software ist, die mit wechselnden Anforderungen einem ständigen Wandel unterliegt.

Die Norm ISO/IEC 9126 von 1991 legt folgende interne und externe Qualitätsmerkmale fest:

- **Funktionalität:** Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmäßigkeit, IT-Sicherheit
- **Zuverlässigkeit:** Reife, Fehlertoleranz, Wiederherstellbarkeit
- **Benutzbarkeit:** Verständlichkeit, Erlernbarkeit, Bedienbarkeit
- **Effizienz:** Zeitverhalten, Verbrauchsverhalten
- **Änderbarkeit:** Analysierbarkeit, Modifizierbarkeit, Stabilität, Prüfbarkeit
- **Übertragbarkeit:** Anpassbarkeit, Installierbarkeit, Konformität, Austauschbarkeit

Unternehmen müssen bei der Umsetzung von Qualität auf ein optimales Kosten-Nutzen-Verhältnis achten: Wie kann ich dem Kunden für das Geld, das er bereit oder fähig ist zu zahlen, das Produkt anbieten, das seinen Wünschen am nächsten kommt?

Deshalb liegt der Schlüssel in der Gewichtung und Beschränkung der Qualitätsmerkmale. Oder provokativ gefragt: Würde ein Automobilhersteller jemals einen Formel 1-Wagen mit einem Verbrauch von drei Litern produzieren, der Container transportieren soll? Fokussiert sich ein Projektteam auf

maximal vier bis sechs Merkmale, gelingt ihm erfahrungsgemäß der Spagat zwischen Kundenanforderungen und Entwicklungskosten bravourös. Gegen einen Rundumschlag in Sachen Qualität spricht auch, dass sich einige Merkmale gegenseitig ausschließen oder zumindest teilweise widersprechen, beispielsweise die Zuverlässigkeit und harte Zeitanforderungen an ein System. Bei ersterer zählen gute Strukturen, bei letzteren sind diese weitgehend zu ignorieren.

Gewichtung von Merkmalen

Quelle: MicroConsult 2003

Da Qualität kostenintensiv ist, müssen sich Entwicklungsunternehmen auf bestimmte Merkmale fokussieren.

Einige Beispiele sollen die zentrale Bedeutung der Gewichtung von Merkmalen noch einmal konkret verdeutlichen:

- **Zuverlässigkeit:** Sie lässt sich am besten in Software realisieren, da diese nicht dem physikalischen Fehler ausgesetzt ist. Während bei der Hardware die Zuverlässigkeit anhand der Zeit gemessen wird, ist es bei der Software die Einsatzart und wie dadurch Design-Fehler auftreten.

- **Benutzbarkeit:** Dieses Qualitätsmerkmal ist dann wenig ausgeprägt, wenn ein Anwender lange braucht, um ein System in den Griff zu bekommen. Eine schlechte Ausprägung dieses Merkmals könnte jedoch Sinn machen, wenn ein Produkt nur selten bedient wird.

- **Effizienz:** Sie wird heute in vielen Projekten überbewertet, beispielsweise im Hinblick auf Speicher oder Prozessorauslastung. Daraus resultieren häufig fehlende Ressourcen für die Integration mehrerer Systemkomponenten in einem Gesamtsystem. Ist beispielsweise ein gutes Zeitverhalten wirklich unumgänglich, muss diese Anforderung in der Spezifikation bewusst formuliert sein.

Die Jahr 2000-Umstellung als Beispiel für ein Zuviel an Funktionalität mit fatalen Folgen hat sicherlich jeder Programmierer noch vor Augen. Heute haben die Unternehmen aus diesem und ähnlichen Fällen gelernt und vermeiden Überimplementierungen weitgehend. Dafür spezifizieren sie schriftlich, welche Funktionen das System nicht erfüllen darf. ■

Entwicklungsprozess: Konsequenter Start

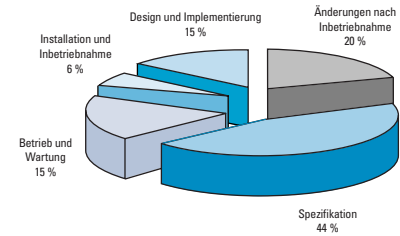
Wer gut sattelt, reitet gut, sagt ein Sprichwort. Bei Projektbeginn sollten deshalb Hersteller wie Betreiber gemeinsam die zentralen Qualitätseigenschaften festlegen. Im Projektverlauf ist deren Erfüllung zu berücksichtigen, und Standards unterstützen den gesamten Prozess der Qualitätssicherung.

Die Grafik auf Seite 18 zeigt anschaulich, dass die meisten Fehler ihre Wurzeln in der Spezifikationsphase haben. Dies ist umso fataler, als die Entwicklungs- und Produktionskosten exponentiell mit dem Zeitraum bis zu ihrer Beseitigung ansteigen. Wer heute noch Qualität ausschließlich über Tests im späteren Entwicklungsprozess realisieren will, überschreitet deshalb schnell den Kostenrahmen. Den Königsweg schlagen hier Unternehmen ein, die systematisch vorgehen und ihre Qualitätsziele von Beginn an konsequent verfolgen.

Merkmale auswählen

So bildet die gezielte Auswahl von Qualitätsmerkmalen bereits eine wichtige Basis für die Anforderungsspezifikation, aus der in der folgenden Analyse die Architektur abgeleitet wird. Diese bestimmt wiederum die Rahmenbedingungen für das Verhalten des späteren Produkts in Bezug auf das Betriebssystem und die Hardware (Design) und schafft damit die Voraussetzungen für die Implementierung.

Fehler im Projekt-Lebenszyklus



Quelle: CATS, G. Glöe 2002

Die Spezifikation ist die Hauptquelle aller Probleme. Diese treten meist spät zutage und sind dann wesentlich aufwändiger zu beseitigen.

Die unterschiedlichen Testverfahren (siehe auch ab Seite 24) beziehen sich auf die Requirement-Analyse und sichern die dort aufgestellten Anforderungen. Im Rückschluss bedeutet dies, dass das Testing ohne eine vernünftige Anforderungsanalyse nicht oder nur durch Zufall zum Erfolg führt.

Entwickler von Embedded Systemen kämpfen allerdings verstärkt mit dem Problem, dass aus einer Anforderung nicht deterministisch die Umsetzung folgt. Dies gilt beispielsweise für zeitkritische Systeme wie Fertigungsmaschinen, die intensiv mit ihrer Umgebung interagieren. Hier gibt es keine klare Wirkungskette für das Realisieren der angestrebten

Quality of Services (QoS) von der Analyse über die Architektur bis zum fertigen Code. Deshalb nähert sich das moderne Engineering über inkrementelle Vorgehensmodelle wie S.P.E.E.D™ oder ROPES den QoS schrittweise an (siehe auch ab Seite 20). Dabei werden die Zwischenergebnisse nicht nur für einzelne Bereiche, sondern vor allem im Gesamtzusammenhang des Projektes betrachtet und bewertet.

Internationale Norm

Die alle Prozesse begleitende IEC 61508 wurde zwar ursprünglich auf das V-Modell zugeschnitten, lässt dem Entwickler jedoch die freie Wahl seines (zeitlichen) Vorgehens, wenn er dabei alle festgelegten Schritte durchführt. So listet die IEC 61508 ganze 450 Merkpunkte für die konkrete Umsetzung auf. Die Norm ist seit 1998 international eine klare Vorgabe, wie ein Embedded System hardware- und softwareseitig zu bauen ist. Und es existieren von ihr mittlerweile auch anwendungsspezifische Ableitungen, beispielsweise für die Eisenbahntechnik, chemische Industrie oder Kerntechnik. In ihrem ersten Teil nennt die IEC 61508 die Anforderungen und die vier Safety Integrity Level für Embedded Systeme. Im Teil drei sind die Bauvorschriften für die Konstruktion der Software festgelegt. ■

Beispiel S.P.E.E.D: Inkrementell zum Ziel

Moderne Vorgehensmodelle wie S.P.E.E.D™ des Entwicklungs- und Beratungsunternehmens Berner & Mattner nähern sich inkrementell dem fertigen Release an und sind gerade für komplexe Embedded Systeme mit harten Echtzeitanforderungen prädestiniert. Dafür planen die Entwickler einfach ein paar Schleifen beziehungsweise Prototypen mehr ein. Das böse Erwachen, wenn sich beispielsweise die Software am Ende partout der Hardware verweigert, ist damit Geschichte.

Generell sollte jeder Entwickler mit dem Modell arbeiten, das die Geschäftsleitung von ihm fordert. Die unternehmensweite Entscheidung für eine Vorgehensweise wird dabei durch den Markt bestimmt, in dem sich der Anbieter bewegt. Wer eher an herkömmlichen Methoden festhalten möchte, sollte seine Einstellung jedoch überdenken: S.P.E.E.D™ nimmt dem Entwickler viel Arbeit ab, weil er den Prozess nicht von Grund auf neu entwickeln muss. Die höheren Weihen von S.P.E.E.D™ lassen sich problemlos in drei Stufen erklimmen:

- Grundmodell für alle Entwicklungsarten
- Branchenspezifisches Modell
- Firmenspezifisches Modell

So lässt sich S.P.E.E.D™ auf einer soliden Basis Schritt für Schritt nach klaren Vorgaben anpassen und erweitern. Inkrementelle Modelle, zu denen auch ROPES (Rapid Object-Oriented Process for

Embedded Systems) zählt, revolutionieren die Entwicklung im Embedded Bereich, weil sie im Gegensatz zur klassischen Wasserfall-Methode eine schrittweise Annäherung an das fertige Release ermöglichen. Jedem Inkrement, also jedem Entwicklungsschritt mit einer klaren Zielsetzung, ist ein eindeutiges Entwicklungsziel mit einer klaren Zeitvorgabe zugeordnet. Folgende Workflows werden dabei mehrfach durchlaufen:

Anforderungsanalyse

Die Anforderungsanalyse spezifiziert die Ein- und Ausgaben eines Systems für eine Umgebung und ihre Randbedingungen. Dies mündet in den funktionalen und nichtfunktionalen Anforderungen.

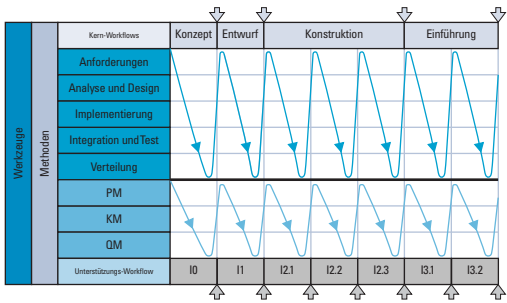
Analyse

Die Architekturanalyse teilt ein System in Subsysteme mit klaren Schnittstellen auf. Sie legt die Bestandteile der Module sowie die lose Kopplung zu anderen Subsystemen fest und achtet auf die Abgeschlossenheit der jeweiligen Untereinheit. Die Objektanalyse analysiert den fachlichen Anteil eines Systems, beispielsweise die Sensorarten. Daraus resultieren sogenannte „Domains“, die den Subsystemen zugeordnet werden und den Schlüssel für einen hohen Wiederverwendungsgrad bilden.

Design

Im Design werden die Quality of Services, wie Sicherheit oder Performance, aus der Anforderungsanalyse berücksichtigt. Das Architektur-Design setzt die Anforderungen der Architekturanalyse an Qualitätsmerkmale und Hardware so um, dass die Architektur auf dem System ablauffähig ist. Das mechanistische Design analysiert die Anforderungen an die Subsysteme, das detaillierte Design behandelt die Klassen in einem Subsystem. Die Sichtbarkeit von Attributen und Operationen wird festgelegt und die Algorithmen sowie Datenstrukturen werden bestimmt.

Workflows am Beispiel des S.P.E.E.D-Modells



Quelle: Berner & Mattner 2002

Jeder Durchlauf der verschiedenen Prozessphasen im inkrementellen Modell liefert in kurzer Zeit einen neuen Prototypen beziehungsweise Versionskandidaten, auf den sich die weitere Entwicklung konkret beziehen kann.

Implementierung

Sie dient dem Erzeugen von Source-Code-Modulen, die dann zu ablauffähigen Software-Komponenten zusammengesetzt werden. Hier können die konstruktiven Diagramme der UML für Klassen, Zustände und Aktivitäten zur Code-Generierung verwendet werden. Der Vorteil: in kurzen Abständen ausführbare und damit testbare Binärkomponenten. Außerdem bleiben Modell und Source-Code synchron.

Test

Die Entwickler erstellen einen Software-Integrations- und -Testplan, der unter anderem die Testvoraussetzungen und die dafür benötigte Hardware konkretisiert. Eine Spezifikation listet zudem die Testfälle auf, die das komplette Programm inklusive aller Anforderungen abdecken. Danach sind die ablauffähigen Software-Komponenten dem Komponententest zu unterziehen. Die im Implementierungs-Workflow schrittweise integrierten Produktversionen kommen sukzessive in den Systemtest. Das nach den Vorgaben der Testspezifikation lauffähige Software-Produkt wird dann gegen die Benutzeranforderungen verifiziert. ■

Das zackige V: Schlauer Testen

Trotz neuer inkrementeller Vorgehensweisen wie S.P.E.E.D™ oder ROPES hat das klassische V-Modell noch lange nicht ausgedient. Es legt fest, wann das Entwicklerteam welche Ergebnisse im Projekt erzeugen muss und wann es diese testet oder prüft. Normalerweise wird es nur einmal durchlaufen, im inkrementellen Ansatz jedoch n-mal. MicroConsult hat S.P.E.E.D™ und V zum „zackigen V-Modell" vereint – für die schlauere Art des Entwickelns und Testens.

„You can't test what you can't execute", meint Bruce Paul Douglass in seinem Klassiker für Entwickler „Doing Hard Time" (siehe Buch-Tipps Seite 37) und veranschaulicht damit den Unterschied zwischen einem Test- und einem Prüfvorgang. So setzt Testen immer ausführbaren Code voraus, während die Prüfung auch anhand von Dokumenten, wie der Anforderungsspezifikation oder dem Architekturmodell, erfolgen kann. Die Methoden dafür sind beispielsweise Review-Techniken (siehe Seite 30) oder statische Analyse.

Das nacheinander abzuarbeitende V-Modell zeigt auf seinem absteigenden Ast die Workflows Requirement-Analyse, Analyse, Design und Implementierung. Der aufsteigende Ast widmet sich ausschließlich den Bereichen Testen und Prüfen. Seine vier Workflows korrelieren dabei mit den gegenüberliegenden im absteigenden Ast, also beispielsweise der Abnahmetest mit der Requirement-Analyse.

Vier Test-Workflows

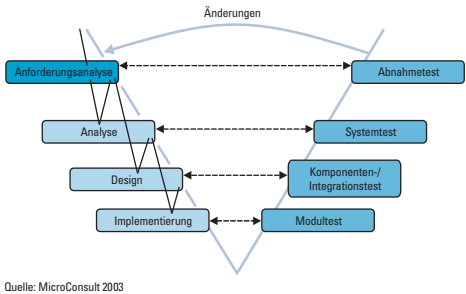
Der Modultest beschäftigt sich mit der konkreten Implementierung, also mit der Funktionalität des Systems. Dagegen nimmt der Komponenten- und Integrationstest die Schnittstelle einer Komponente unter die Lupe und untersucht, ob sich diese gemäß der Spezifikation verhält. Der Systemtest des Herstellers ist im Grunde ein vorweggenommener Abnahmetest durch den Kunden und widmet sich der Funktion des Systems auf Basis der Anforderungsspezifikation. Im Gegensatz zum Abnahmetest kann hier auf genaue Kenntnisse der Systeminterna zurückgegriffen werden.

Das zackige V-Modell

Das klassische V-Modell verträgt sich gut mit dem modernen S.P.E.E.D™, weil auch dort jedem Workflow ein Test-Workflow zugeordnet ist, also beispielsweise die Implementierung dem Modultest. Die modellbasierte, objektorientierte Entwicklung erlaubt zudem heute das frühzeitige Ausführen des Modells und damit das Prüfen oder Testen in den Anfangsphasen. Die Zacken im V-Modell von MicroConsult verweisen auf diese Möglichkeiten in den Workflows Anforderungsanalyse, Analyse und Design. So lassen sich die Ergebnisse der Requirement-Analyse mittels semiformaler oder informeller Verfahren validieren, weil dort bereits Code

erzeugt und ausgeführt werden kann. Zu den informellen Verfahren zählen die Review-Techniken, semikonstruktiv ist dagegen beispielsweise die Code-Generierung aus der Requirement-Analyse. Mit diesen Methoden stellt der Hersteller gemeinsam mit dem Kunden die Konsistenz und Vollständigkeit der Anforderungsanalyse sicher.

Zackiges V-Modell



Quelle: MicroConsult 2003

Die Kombination von V-Modell und inkrementellen Verfahren wie S.P.E.E.D™ oder ROPES macht Testen und Prüfen in frühen Projektphasen möglich. Diese Optionen sind durch die Zacken im absteigenden Ast symbolisiert.

Auch in der Analysephase lassen sich die Architektur und fachlichen Inhalte mit semiformalen wie informellen Verfahren prüfen. Die Design-Phase dient der Adaption von Architektur und fachlichen Teilen auf Betriebssystem und Hardware. Dieser

Workflow untersucht, ob das Konzept stimmig ist und ob die nichtfunktionalen Anforderungen der Requirement-Spezifikation im Design-Modell Berücksichtigung fanden. Hier bieten die erwähnten Review-Techniken weitreichende Unterstützung.

Aufbau des Workflows

Die Test-Workflows basieren auf klar definierten Zielen. In der Vorbereitungsphase müssen die Testbedingungen inklusive Umgebung und Szenarien festgehalten werden. Dazu zählen die Sollwerte eines Prüflings und die Situation, in der er diese liefern soll. Letztendlich führt ein Testszenario zu manuell geschriebenem oder automatisch generiertem Code.

Außerdem ist ein klares Testabbruch-Kriterium für den Fall zu definieren, dass der Prüfling keine brauchbaren Ergebnisse liefert. Das Testende-Kriterium legt fest, wann der Test erfolgreich abgeschlossen ist. Aus Motivationsgründen sollte sich der Tester jedoch in seinem abschließenden Fehlerreport nicht auf Ja-Nein-Angaben beschränken, sondern die Entwicklung konstruktiv unterstützen. Diese muss nämlich die Testergebnisse in einem weitergehenden Report interpretieren und effektiv die gefundenen Defekte beseitigen. Bei dieser Analyse werden auch Metriken für das Qualitätsmanagement generiert, die den Gesamtzustand des Systems beschreiben, zum Beispiel „60 % fertiggestellt“.

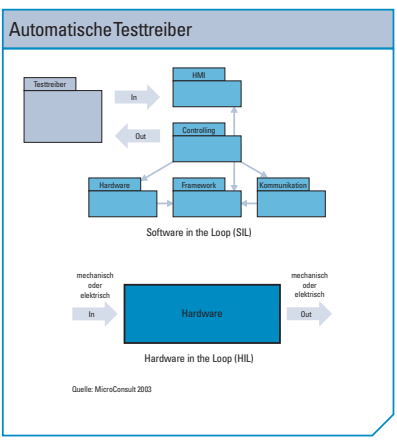
Verfahren auswählen

Jedem Test-Workflow wird ein Testverfahren zugeordnet. Welche Methode mit welchem Werkzeug zum Einsatz kommt, hängt von dem Test- und Wirtschaftlichkeitsziel des Systems ab. Die Testziele für jedes Inkrement ergeben sich wiederum aus dessen Entwicklungszielen. Die objektorientierte, modellbasierte Entwicklung nutzt dabei weiterhin klassische Verfahren wie White- oder Black-Box-Test und eröffnet gleichzeitig neue Perspektiven.

Weil Hersteller mit modernen Modellen nicht mehr einzelne Verfahren betrachten, sondern über die Prozess-Workflows das Gesamtsystem im Auge behalten, lässt sich die Wirtschaftlichkeit der Systementwicklung gut austarieren. So können beispielsweise Qualitätsmerkmale und Testverfahren in einen Zusammenhang gebracht werden, der Projekte wirtschaftlicher macht. Beispielhaft soll hier das Zusammenstellen der Verfahren für die einzelnen Workflows näher beleuchtet werden.

Für den Modultest bietet sich die statische Analyse an, die den Code auf Basis von Codier-Richtinien mit Werkzeugen wie PC-LINT prüft. Diese Methode bereitet weitere Testvorgänge vor, da der Hersteller die Qualität des Codes bereits in einem frühen Projektstadium gut einzuschätzen lernt. Tools für White- oder Grey-Box-Tests sind dann einzusetzen, wenn die Funktionalität der einzelnen Methoden oder Funktionen über Parameter zu testen ist.

Der Workflow für Komponenten- und Integrationstest prüft die Komponenten auf Basis der implementierten Funktionalität im Inkrement und nutzt dazu im Black-Box-Test nur die Schnittstelle. In dieser Phase werden die verschiedenen Komponenten im Zusammenspiel dahingehend untersucht, ob in dem weiterentwickelten System die vormals spezifizierten und implementierten Funktionalitäten nach wie vor erfüllt sind. Hier ist ein Regressionstest die erste Wahl, dem beim inkrementellen Vorgehen eine Schlüsselrolle zukommt. Er weist die Kompatibilität der weiterentwickelten Teile nach. Zudem lässt er sich für jedes Inkrement wirtschaftlich durchführen, wenn das Unternehmen systematisch Testtreiber entwickelt, die sich automatisch aufrufen lassen (siehe Abbildung). ■



Bei Software in the Loop (SIL) wird Software mit Software getestet, hier dargestellt anhand eines Paketdiagramms der UML. Die Pakete sind Modulen gleichzusetzen. Hardware in the Loop (HIL) „injiziert" Eingangswerte mechanisch oder elektrisch in das System. Ein HIL-Teststand kann also Signale an die Klemmen des Embedded Systems bringen, die Ausgangssignale aufzeichnen und mit den Sollwerten vergleichen.

Review-Techniken: Der Mensch als Prüfer

Wenig fortschrittlich zeigen sich die Deutschen in Sachen Reviews: nicht einmal zehn Prozent der Embedded Entwickler nutzen die informellen Techniken der Software-Prüfung heute professionell. Rasches Umdenken ist hierzulande gefragt angesichts des wachsenden Kostendrucks und effizienterer Entwicklungsmodelle wie S.P.E.E.D™ oder ROPES.

In den bedeutenden ersten Projektphasen kann nur ein geschulter Reviewer Fehler wirklich gezielt aufspüren. Eine Stunde Aufwand ist durchschnittlich für das Auffinden eines Fehler inklusive seiner Korrektur nötig, bei software-gestützten Testverfahren sind es bereits drei bis acht Stunden. Der anteilige Aufwand für Review-Techniken in einem typischen Software-Projekt sollte sich heute bei fünf bis 15 Prozent einpendeln. Im Vergleich dazu: die Testphasen verschlingen immerhin durchschnittlich 40 bis 50 Prozent des Projektaufwands.

Informelle Nachweisverfahren wie Reviews erfordern zwar weniger Zeitaufwand als Tests, spüren aber nur durchschnittlich 60 Prozent der Fehler auf. Sie sind deshalb erst in Kombination mit anderen Prüfmethoden sinnvoll einsetzbar.

Das inkrementelle Vorgehen basiert dabei ganz wesentlich auf modernen Notationen wie der UML, die sich dem klassischen Software-Test natur-

gemäß entzieht. Der Markt hat generell heute noch so gut wie keine Test-Tools zu bieten, die sich prozessübergreifend anwenden lassen. Debugger und Tracer helfen zwar punktuell bei der Verifizierung, letztendlich steht und fällt die übergreifende Systemprüfung jedoch mit der Erfahrung des Entwickler-Teams. Drei informelle Nachweisverfahren unterstützen die moderne Projektarbeit:

- Management-Review,
- Walk-Through und
- klassische Inspektion.

Das Management-Review bezieht in der Anfangsphase das Management und den Kunden beziehungsweise Produkt-Manager als Repräsentanten des Kunden mit ein und stellt übergreifende Fragen nach Projektzielen sowie Anforderungen an das Release. Im darauf folgenden Walk-Through erläutern die Autoren ihre auf Basis des Management-Reviews erstellten Dokumente. Hier geht es vor allem darum, Design-Alternativen gegeneinander abzuwägen.

In fast allen Phasen des Spiralmodells findet sich die klassische Inspektion, welche die Textdokumente Wort für Wort und den Code Zeile für Zeile überprüft. Bei der Code-Inspektion gilt es, die sogenannte „optimale Inspektionsrate" einzuhalten. Sie liegt im Bereich von 100 bis 150 NLOC (non-commentary Lines of Code) in der Stunde. Auch bei

Textdokumenten gibt die Fachliteratur optimale Inspektionsraten an, wobei die Meinungen hier differieren. Einige Autoren setzen drei bis fünf Seiten pro Stunde an, Altmeister Tom Gilb empfiehlt dagegen nur ein bis zwei Seiten pro Stunde.

Reviews in allen Phasen

Letztendlich verfolgen Review-Techniken den Top-Down-Ansatz: in den frühen Phasen dienen sie der strategischen Orientierung sowie groben Ausrichtung auf Management-Ebene. Je weiter der Entwicklungsprozess fortgeschritten ist, desto tiefer reicht die Prüfung und desto mehr Dokumente sind einzubeziehen. Dabei darf jedoch gleichzeitig der Blick für das Wesentliche nicht verloren gehen.

Vor Fehlern oder Mängeln schützen auch ergänzende Check-Listen, die sich der Entwickler aus dem Internet laden kann. S.P.E.E.D™ hat diese bereits umfassend und für alle Projektphasen integriert. Den Königsweg selbst erstellter Check-Listen beschreiten heute noch zu wenige Unternehmen, obwohl die Methode „Lerne aus Deinen eigenen Fehlern“ eigentlich nahe läge. Das Team bestimmt dazu beispielsweise die zehn problematischsten Fehler des letzten Jahres und formuliert daraus zehn Check-Fragen. Treten weitere Mängel auf, lässt sich die Liste jederzeit einfach erweitern.

Anforderungsanalyse: Ziele finden

Als besonders wirksam haben sich Reviews bei Anforderungsanalyse, Design, Codierung und Testspezifikation erwiesen. Die Vorteile des Management-Review zeigen sich bei der Anforderungsanalyse, da hier generelle Ziele formuliert werden, die in der Anforderungsspezifikation münden. Nur was in dieser festgelegt ist, lässt sich in späteren Prozessphasen gegenprüfen. Wird ein Fehler erst in der Testphase offensichtlich, wird es meist teuer. Pro „Major Defect" ist dann im Industriedurchschnitt ein Bearbeitertag anzusetzen, wobei der Embedded Bereich aufgrund seiner wachsenden Komplexität und Kundenanforderungen mehr Zeit einkalkulieren muss.

In der Anforderungsspezifikation häufig unzureichend oder nicht festgelegt sind die nicht-funktionalen Anforderungen, also die Quality of Services (QoS). Dazu zählen unter anderem Laufzeitverhalten, Sicherheit und Performance, bei hohen Stückzahlen auch Prozessoren und Speicher. Sie spielen in (Real-time) Embedded Systemen eine zentrale Rolle, entziehen sich aber leicht dem Auge des Reviewers.

Analyse und Design: Diskussionen

Die ersten Reviews in den Analyse- und Design-Phasen sind häufig von hitzigen Debatten begleitet, geht es doch darum, zwei bis drei große Alternati-

ven im Walk-Through auf ihre Tragfähigkeit hin zu überprüfen. Erst nach der Entscheidung für eine Design-Richtung können die Autoren ihre Dokumente erstellen und diese dann für die klassische Inspektion bereitstellen. Beim Design liegt ein besonderes Augenmerk auf den Quality of Services. Alle in der Anforderungsspezifikation beschriebenen Quality of Services müssen im Design berücksichtigt worden sein. Leicht offenbaren sich an dieser Stelle die Fehler, weniger leicht die Lücken in der Spezifikation, und die oben erwähnten Check-Listen werden sicher eine wertvolle Bereicherung für Folgeprojekte erfahren.

Codierung: mühevolle Kleinarbeit

In der Source-Code-Inspektion nehmen die Autoren Zeile für Zeile unter die Lupe und prüfen das Programm oder Teile davon gegen die Design-Dokumente, welche die Anforderungen an den Code konkretisieren. Diese Knochenarbeit können statische Analyse-Tools wie LINT oder QA-C dem Entwickler zwar nicht völlig abnehmen, sie leisten in diesem Stadium dennoch wertvolle Unterstützung. Die meiste Zeit müssen die Code-Inspektoren vor der eigentlichen Review-Sitzung investieren. 80 Prozent der Fehler lassen sich in dieser Vorbereitungsphase ausfindig machen, die restlichen 20 Prozent während des Meetings.

Test: 99,9% abdecken

In der klassischen Software-Entwicklung ist es nicht selten, dass der Testüberdeckungsgrad einer Testspezifikation gerade einmal bei 50 Prozent liegt. Natürlich kann dieser Wert keine Orientierung für den sicherheitskritischen Embedded Bereich liefern. Hier wäre eine Abdeckung von 99,9 Prozent aufgrund der gegebenen Randbedingungen durchaus erstrebenswert. Reviews sind in der Testphase ein ideales Hilfsmittel, um sicherzustellen, dass die Testspezifikation tatsächlich alle benötigten Testfälle enthält und den erforderlichen Testüberdeckungsgrad erreicht.

Wie gut Review-Techniken in den verschiedenen Stadien des Projektzyklus greifen, lässt sich sofort anhand der Stunden und gefundenen Fehler ermitteln. Fortschrittliche Unternehmen führen sogar umfassende Statistiken, um festzustellen, mit welcher Prüfmethode sie welche Major Defects in welcher Phase gefunden haben. Damit optimieren sie laufend das Zusammenspiel von Review, statischer Analyse und dem klassischen Ablauftest. Folgen die Entwicklungszyklen im inkrementellen Vorgehensmodell beispielsweise sehr kurz aufeinander, kann der Kosten-Nutzen-Effekt von Reviews in den späteren Phasen zu gering sein und der Verzicht auf eine detaillierte Code-Inspektion nahe liegen. Letztendlich gilt also für Prüfverfahren wie so häufig im Entwickleralltag: auf die richtige Mischung kommt es an. ■

Vergleich informeller Nachweisverfahren

Kriterien	Management-Review	Walk-Through	Inspektion
Zusammensetzung	Projektteam, Auftraggeber, Auftragnehmer	Autor, Fachkollegen, Prüfer	Moderator, Autor, Tester, Anwender
Teilnehmer	5 bis 15	2 bis 6	3 bis 6
Dauer	1 bis 2 Tage	max. 2 Stunden	max. 2 Stunden + Vorbereitung
Objekte	Phasenprodukte, Projektpläne, Projektberichte, Problembereich	Dokumente, Software-Grob- und -Feinentwurf	Dokumente inkl. Vertrag und Code
Ziel	Ermitteln Projektstatus, Problemlösung, Maßnahmen	Fehler und Unvollständigkeiten feststellen	Prüfen von Dokumenten
Zeitpunkt	Phasenende, Meilenstein	Dokumente erstellt	Dokumente erstellt
Durchführung	Tagesordnung	Gedankliches Ausführen vor Zuhörern	Checkliste
Ergebnis	Protokoll mit Beschlüssen	Protokoll mit Fehlern und Unvollständigkeiten	Protokoll mit inspizierten Dokumenten und eventuell Neuinspektion

Info Pool

Buch-Tipps

Binder, Robert V.: Testing Object-Oriented Systems. Addison Wesley 2000, 1.191 Seiten, ISBN 0-201-80938-9
Das Buch ist ein wichtiges Standardwerk für das automatisierte, modellbasierte Testen von objektorientierten Anwendungen mit vielen praktischen Beispielen.

Broekman, Bart; Notenboom, Edwin: Testing Embedded Software. Addison Wesley 2003, 320 Seiten, ISBN 0-32115-986-1
Die Autoren adaptieren Projekterfahrungen aus dem strukturierten Testen kommerzieller Software auf die Anforderungen von Embedded Systemen mit Fokus auf die praktische Organisation des Testprozesses.

Douglass, Bruce Powel: Doing Hard Time. Developing Real-Time Systems with UML, Objects, Frameworks and Patterns. Addison Wesley 1999, 749 Seiten, ISBN 0-20149-837-5
Das Werk bietet unter anderem eine aufschlussreiche Einführung in den Rapid Object-Oriented Process for Embedded Systems (ROPES).

Gilb, Tom; Graham, Dorothy: Software Inspection. Addison Wesley 1993, 471 Seiten, ISBN 0-20163-181-4
Das erste und umfangreichste Lehrbuch ist noch immer das Standardwerk für alle Reviewer und solche, die es werden wollen.

Niemann, Ralf: Hardware/Software Co-Design for Data Flow Dominated Embedded Systems. Kluwer Academic Publishers 1998, 223 Seiten, ISBN 0-79238-299-4
Niemann gibt eine Einführung in Systemspezifikation, Hardware/Software Partitioning, Co-Synthese und Co-Simulation.

Sommerville, Ian: Software Engineering. Addison Wesley 2001, 712 Seiten, ISBN 3-8273-7001-9
Der Klassiker bietet eine breit angelegte Übersicht über alle Aspekte des Software-Engineering inklusive der Standardtechniken zur Entwicklung ausgedehnter Software-Systeme.

Thaller, Georg Erwin: Software-Qualität. VDE-Verlag 2000, 247 Seiten, ISBN 3-80072-494-4
Der Autor beschreibt die gängigen Methoden wie Verifikation und Validierung, Test, Debugging, Regression Testing, Inspektionen, Walk-Throughs und Prozessmodelle, aber auch fortschrittliche Techniken für das Management wie Tailoring oder Clean-Room.

Thaller, Georg Erwin: Software-Test. Verifikation und Validation. Heise 2002, 394 Seiten, ISBN 3-88229-198-2
Die Techniken und Methoden beim Test werden an einem durchgehenden Beispiel in C erläutert, so dass sich auch weniger erfahrene Leser leicht einarbeiten können.

Wiegers, Karl E.: Peer Reviews in Software. A Practical Guide. Addison Wesley 2001, 256 Seiten, ISBN 0-20173-485-0
Mehr noch als Gilb und Graham geht Wiegers in seinem Werk auf typische Review-Problemfälle ein. Der Leser erfährt beispielsweise, wie er besonders umfangreiche Dokumente per Stichprobenverfahren prüfen kann.

Web-Tipps:

www.afm.sbu.ac.uk/safety
Linkliste zu Informationen über sicherheitskritische Systeme, unter anderem Veröffentlichungen, Newsgroups und Organisationen

www.asq.org
Die Site der American Society for Quality

www.bms.de/speed
Alle Infos rund um S.P.E.E.D™ auf der Site von Berner & Mattner, dort auch ein Link zu einem hilfreichen Whitepaper

www.embedded-quality.de
Das BMBF-Verbundprojekt EQUAL zur Erforschung und Entwicklung von Embedded Quality

www.icstest.com
International Conference on Software Testing, findet in mehreren Ländern statt, in Deutschland in Köln

www.reviewtechnik.de
Dort finden Entwickler unter anderem auch eine der umfassendsten Linksammlungen zu Review-Checklisten.

www.soft.com/Institute/HotList
Linkliste zu weltweiten Sites über Software-Qualität

www.stickyminds.com
Online-Magazin für Software-Manager und -Tester, viel Wissenswertes rund um Testing, Configuration Management oder Requirements

Kurse zum Thema

High Quality Requirement-Analyse (SW/REQ)

Teilnehmer:.....Requirement- und System-Engineers sowie Software-Entwickler von Embedded Systemen; Entwickler und Manager mit Testverantwortung

Vorkenntnisse...Projekterfahrung in der technischen Software-Entwicklung

Ziel.....Alle wesentlichen Schritte von der Anforderungsspezifikation und -analyse über das Referenzmodell bis hin zur Validierung und Verifikation

Inhalt.....Einordnung in den modellbasierten Entwicklungsprozess mit seinen Technologien und Werkzeugen; Erstellen von High Quality Anforderungsspezifikationen und Referenzmodell; automatisiertes Testen der realen Implementierung; Reuse-Aspekte von durchgängigen, toolunterstützten Ansätzen

Dauer.....5 Tage

Preis.....1.990 Euro + Ust.

Software-Qualität für Embedded Systeme (SW/Q)

Teilnehmer.....Software-Entwickler von Embedded Systemen mit Qualitätsbewusstsein; Entwickler und Manager, die für die Software-Qualität verantwortlich sind

Vorkenntnisse...Projekterfahrung mit Software-Systemen

Ziel.....Systematisches Vorgeben der angestrebten Qualität von Software; Kenntnisse über Stand zur Qualität von Software; Bestätigung der erreichten Qualität

Inhalt.....Einführung in die Qualität von Entwicklungsprozessen; Qualität der Produkte Rechner und Software; Bestätigung der Produktqualität; praktische Übungen zum Ableiten von nicht-funktionalen Anforderungen aus Standards, Prüfen von Code (C51, C167) sowie Bewerten von Prüfergebnissen

Dauer.....5 Tage

Preis.....1.890 Euro + Ust.

Testen im Embedded Markt (SW/TEST)

Teilnehmer.....Projektleiter, Testmanager, Software-Entwickler, Qualitätsmanager, QS-Beauftragte, Software-Tester

Vorkenntnisse...Grundkenntnisse in einer höheren Programmiersprache (beispielsweise C/C++)

Ziel.....Umfassender Einblick in das professionelle Testen und die Qualitätssicherung; Tests planen und bewerten können

Inhalt.....Entwicklung eines Testkonzeptes; Teststrategie, -methoden und Software-Lifecycle; Testmanagement; Schritte innerhalb des Testprozesses; Software-Quality-Management und -Testen

Dauer.....5 Tage

Preis.....1.990 Euro + Ust.

Reviews erfolgreich durchführen (REVIEWT)

Teilnehmer.....Software-Entwickler, Projektleiter, QS-Beauftragte, Software-Tester

Vorkenntnisse...Grundkenntnisse in einer höheren Programmiersprache (idealerweise C), branchenübliche Englischkenntnisse

Ziel.....Erhöhen der Software-Qualität; Steigern der Produktivität von Software-Projekten; als Moderator ein Review planen und leiten sowie die Ergebnisse quantitativ wie qualitativ bewerten

Inhalt.....Einführung in die Review-Techniken; 15 kritische Erfolgsfaktoren; Nutzen von Reviews; Videoszenen von guten und schlechten Reviews; Übungsbeispiel; Einführung in das Lotus Inspection Data System (LIDS); Arbeitshilfsmittel

Dauer.....2 Tage

Preis.....1.150 Euro + Ust.

Objektorientierte Analyse für Embedded Systeme (OOA/UML-RT)

Teilnehmer.....Software-Entwickler von Embedded Systemen

Vorkenntnisse...Projekterfahrung mit Embedded Systemen

Ziel.....Richtiges Einschätzen des Einsatzes der UML für Embedded Systeme; professionelle Nutzung von Frameworks; effizientes Entwickeln von Architekturen in Realtime-Systemen

Inhalt.....UML und Embedded Systeme; Vorgehensmodelle; grundlegende Software-Architekturen; Modellierung von Echtzeitsystemen

Dauer.....5 Tage

Preis.....1.990 Euro + Ust.

Die aktuellen Kurstermine: **www.microconsult.de**

Weitere Informationen

Andreas Munk – Ihr Vertriebskontakt

Tel.: +49 (0) 89 /45 06 17 - 42

E-Mail: a.munk@microconsult.de

Zum guten Schluss

MicroConsult beschreitet aktiv viele Wege, um Kunden und Interessenten frühzeitig über die wichtigen Themen der Zeit zu informieren – mit Events, Trend Guides, Fachartikeln und natürlich Trainings. Der Erfolg dieser Projekte liegt im wesentlichen in der Teamarbeit begründet.

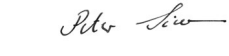
Viele kreative Köpfe bei MicroConsult und in Partnerunternehmen haben fundiertes Wissen, inspirierende Ideen und unterschiedliche Sichtweisen in den Trend Guide „Embedded Quality“ eingebracht.

Wir danken:

- unseren Kollegen für inhaltliche und organisatorische Mitarbeit,
- Dr. Günter Glöe, Leiter Softwarezertifizierungsstelle, TÜV Nord, Peter Rösler, QS-Beauftragter Airports/Airlines, Softlab und Siegfried Hörfarter, Fachgebietsleiter Prozess- und Methodenberatung, Berner & Mattner Systemtechnik, für inhaltliche Unterstützung,
- Eva Schulz, actimedia, für die redaktionelle Umsetzung
- den Sponsoren ELEKTRONIKPRAXIS und iSYSTEM, die mit dazu beigetragen haben, dass dieser Trend Guide für Sie kostenlos ist

Der nächste Trend Guide ist bereits in Arbeit. Lassen Sie sich überraschen!

Sabine Häring
Marketing Manager



Peter Siwon
stellv. Geschäftsführer

Partnerverzeichnis

Fokus	Firma
UML-Tools	ARTiSAN Software Tools GmbH Eupener Str. 135-137 D-50933 Köln
Entwicklung, Beratung, Werkzeuge	Berner & Mattner Systemtechnik GmbH Otto-Hahn-Str. 34 D-85521 Ottobrunn
Software-Engineering and Design	HERMES SoftLab Litijska 51 1000 Ljubljana, Slovenia
Development Tools	Hitex Development Tools GmbH Greschbachstr. 12 D-76229 Karlsruhe
UML-Tools	I-Logix Deutschland GmbH Bahnhofstr. 39 85591 Vaterstetten
Entwicklungswerk-zeuge & -umgebung, Emulatoren, Debugger	iSYSTEM GmbH Carl-Zeiss-Str. 1 D-85247 Schwabhausen
Automatic Test- and Validation-Tools	OSC - Embedded Systems AG Industriestr. 11 26121 Oldenburg
On Top Solutions for System on Silicon Debugging	pls Programmierbare Logik & Systeme GmbH Technologiepark D-02991 Lauta

Info	Kontakt
www.artisansw.com	Christiane Kapteina Tel. +49 (0)2 21/4 85 22-60 christiane.kapteina@artisansw.com
www.bms.de	Georg Zimmermann Tel. +49 (0) 89/60 80 90-1 60 georg.zimmermann@bms.de
www.hermes-softlab.com	Martin Weiss Tel. +43/1/9 94 96 50-21 martin.weiss@hermes-softlab.com
www.hitex.de	Christiane Simon Tel. +49 (0)7 21/96 28-1 46 sales@hitex.de
www.ilogix.com	Wolfgang Leimbach Tel. +49 (0) 81 06/37 96 60 wleimbach@ilogix.com
www.isystem.com	Marian A. Wosnitza Tel. +49 (0) 81 38/69 71-51 marian.wosnitza@isystem.com
www.osc-es.de	Guido Sandmann Tel. +49 (0)4 41/3 50 42-3 30 guido.sandmann@osc-es.de
www.pls-mc.com	Heiko Riessland Tel. +49 (0)3 57 22/3 84-0 info@pls-mc.com

Trend Guide Medienpartner:

www.elektronikpraxis.de

MicroConsult GmbH • Schule für MicroElektronik & Informationstechnologie
Rosenheimer Straße 143 b • D-81671 München • Tel.: (089) 45 06 17- 71
Fax: (089) 45 06 17- 17 • www.microconsult.de