Gepubliseerd in: *PT-Embedded*, Mei/Juni 2002

Gestructureerd testen van embedded software

Erik van Veenendaal / Rob Hendriks
(Improve Quality Services BV)

De risicofactor bij embedded software is veelal erg groot. De economische, juridische of zelfs menselijke schade kan hoog oplopen als gevolg van een software fout. Een gestructureerd testproces van goed niveau is derhalve noodzaak. In dit artikel beschrijven de auteurs vanuit hun praktijkervaringen een aantal "test practices" die gebruikt kunnen worden om het embedded software testproces op een hoger niveau te brengen. Hierbij is de testaanpak TMap als uitgangspunt gehanteerd. Met name wordt ingegaan op testtechnieken en -tools aangezien op deze onderdelen de grootste verschillen kunnen worden onderkend tussen het testen van embedded software en het testen van informatiesystemen..

Het belang van software in de maatschappij neemt steeds toe. Software is niet meer gelimiteerd tot het domein van administratieve informatiesystemen, in allerlei industriële en consumentenproducten stijgt de hoeveelheid software exponentieel (Rooijmans *et al*,1996). Ondanks goede resultaten met diverse kwaliteitsmethodieken, is de IT-industrie nog steeds ver verwijderd van foutloze software. Testen is én blijft een belangrijk onderdeel van het softwareontwikkelings- en onderhoudsproces. Veelal neemt testen zo'n 30% tot 40% van het totale budget voor haar rekening. Zowel het toenemend belang van software in de maatschappij als de kosten die testen met zich mee brengt, geven de noodzaak aan voor een gestructureerde testaanpak.

Een veelgebruikte testaanpak is de Test Management approach (TMap®) (TMap, 1995). TMap heeft zich in de laatste jaren ontwikkeld tot de defacto standaard ten aanzien van testen in Nederland. De meeste banken, verzekeringsinstellingen, pensioenfondsen en overheidsorganisaties bedienen zich op enigerlei wijze van TMap. TMap is in eerste instantie ontwikkeld voor het testen van informatiesystemen en wordt soms beschouwd als een aanpak die niet kan worden toegepast in technische c.q. embedded softwareomgevingen. TMap is echter een generiek model dat bestaat uit vele onderling samenhangende bouwstenen. Op basis van expertise, ten aanzien van het toepassingsgebied en kennis en kunde met betrekking tot het gestructureerd testen, dienen uit de grote hoeveelheid beschikbare bouwstenen de juiste te worden geselecteerd. In de afgelopen jaren heeft TMap zich ook in de embedded software industrie bewezen als een zeer bruikbaar standaard.

De auteurs identificeren een aantal bouwstenen die meestal van toepassing zijn bij het testen van embedded software. De selectie is gebaseerd op praktijkervaringen met betrekking tot het toepassen en implementeren van gestructureerd testen in diverse embedded software projecten. Door de juiste bouwstenen te selecteren en terminologie te hanteren, kunnen testmethodes en –technieken zoals TMap, van grote toegevoegde waarde zijn bij het testen van technische c.q. embedded software producten.

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

In dit artikel wordt de volgende definitie van embedded product gehanteerd:

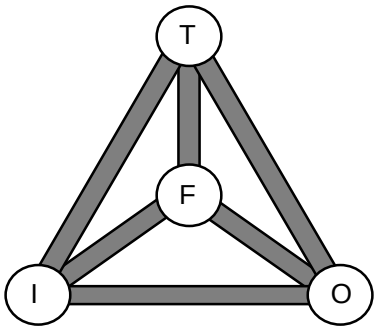
Embedded product Een autonome fysieke eenheid, bestaande uit hardware (elektronica) met daarin software, waartussen een interactie plaatsvindt via een specifieke interface. (Van Solingen en Rodenbach, 1996)

Een embedded product vervult derhalve een zodanige autonome functie dat het voor de gebruiker van dat product niet duidelijk is of bepaalde functionaliteit is gerealiseerd in de vorm van hardware of software.

Gestructureerde testaanpak

Een gestructureerde aanpak steunt op een aantal aan elkaar gerelateerde pijlers. Op basis van testliteratuur en praktijkervaring wordt binnen TMap een viertal aandachtsgebieden onderkend:

- een met de ontwikkelingscyclus samenhangende *fasering* van testactiviteiten;
- een goede *organisatorische* inbedding;
- bruikbare *technieken* voor de uitvoering van de testactiviteiten.
- de juiste *hulpmiddelen* (tools) en *infrastructuur*;



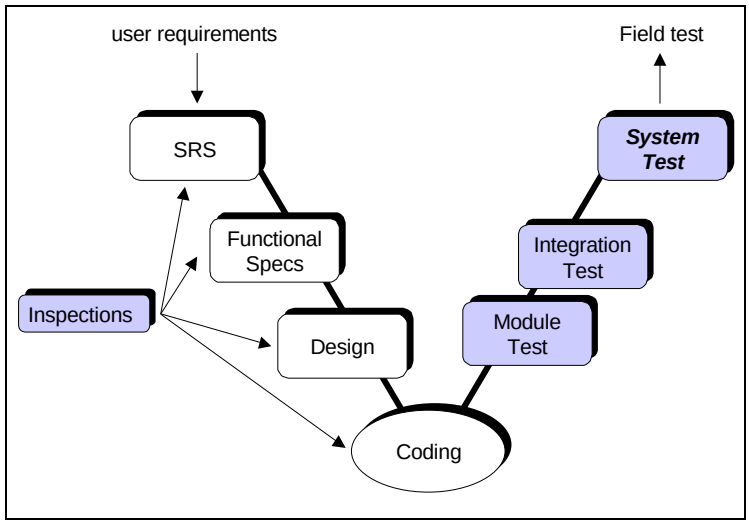
Figuur 1: De vier pijlers onder de TMap testaanpak

Fasering

Een veelgebruikte wijze om de testfasering te structuren is het zogenaamde V-model (zie figuur 2). Aan de linkerkant staan de fasen waarin het systeem wordt gebouwd en aan de rechterkant de te onderkennen testfasen (oftewel testsoorten). In uitgebreide varianten van het V-model, zoals figuur 2, worden tevens de voorgenomen inspecties aangegeven. Naast de ontwikkeldocumenten kunnen uiteraard ook inspecties plaatsvinden van de testspecificaties. Het V-model is een raamwerk dat per project c.q. organisatie dient te worden geconcretiseerd. Per testsoort dienen de doelstellingen, verantwoordelijkheden, te testen kwaliteitsattributen, entry en exit criteria etc. te worden vastgesteld. Het V-model is uiteraard ook van toepassing bij technische toepassingen. Het definiëren van de moduletest is hierbij vaak veel duidelijker algevolg van de meer traditionele ontwikkeltalen (C, C^{++}, Java). Bij dit soort ontwikkeltaken zijn de modules beter te identificeren dan bij 4 of 5GL omgevingen die vaak worden toegepast bij het ontwikkelen van informatiesystemen. Het definiëren van de hogere testsoorten (alpha- c.q. systeem- en betatest) blijkt echter in de praktijk vaak vele malen moeizamer. Tijdens deze tests dient niet slechts de software te worden getest maar ook de parallel nieuw ontwikkelde hardware en mechanica. Bij het vaststellen van de verantwoordelijkheden en doelstellingen zijn dan meerdere partijen betrokken. In de embedded software praktijk komen zelfs vaak

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

faseringen voor die gebruik maken van geneste V-modellen met een hoog complexiteitsgehalte.



Figuur 2: V-model

Het V-model lijkt in eerste instantie alleen van toepassing bij traditionele ontwikkelmethoden op basis van het waterval model, echter ook bij incrementele en evolutionaire ontwikkelingmethoden is dit model uitermate bruikbaar. Bij incrementele ontwikkeling wordt de "V" geheel of gedeeltelijk doorlopen voor ieder increment. Ook bij evolutionaire ontwikkelingmethoden zijn de testfases uit het V-model van toepassing, echter aangezien de ontwikkeldocumentatie is het begin globaal is, en pas op het einde wordt gedetailleerd heeft dit gevolgen voor het testtraject. Bij het opzetten van testgevallen wordt initieel veelal gebruik gemaakt van informele technieken (bijv. use cases) en pas in een later stadium van het ontwikkeltraject van formele technieken (bijv. beslissingstabellen).

Organisatie

Een testproces wordt uitgevoerd door mensen en behoeft daarom organisatie. Enerzijds is er de organisatie binnen het testteam, waar ieder zijn taken en verantwoordelijkheden moet krijgen en de inbedding van het testteam in de (project)organisatie. Anderzijds dient aandacht te worden besteed aan de implementatie van gestructureerd testen binnen de organisatie. Voor iedere testsoort moet beschreven zijn wie hem uitvoert, wie er verantwoordelijk voor is, wie controleert dat de test goed verloopt en aan wie de resultaten worden gerapporteerd. Op organisatorisch niveau dienen testfuncties te zijn gedefinieerd met bijbehorende taken en opleidingseisen alsmede teststandaards. Hierin dient uiteraard ook aandacht te worden besteed aan de testactiviteiten van de software engineer. Voor de organisatorische pijler geldt dat er niet veel verschillen zijn tussen het testen van administratieve en embedded softwaresystemen.

Echter, bij gelijktijdige hard- en softwareontwikkeling verdient het maken van duidelijke afspraken maken over de verantwoordelijkheden voor het testen van het grensgebied tussen hardware en software voldoende aandacht. Door goede afspraken te maken met de hardware discipline kan worden voorkomen dat delen van het systeem dubbel of erger nog, niet worden getest.

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Daarnaast is een verschil te onderkennen in de eisen die aan de tester worden gesteld. Naast de algemene opleidingseisen op het gebied van testen zal een embedded software tester niet kunnen zonder een ruime dosis technische kennis, met name kennis ten aanzien van het hardware platform, de ontwikkelomgeving, ontwikkeltools en de programmeertaal. Een black-box tester "pur sang" is binnen de technische automatisering nauwelijks aanwezig. Op het gebied van opleidingen is naast de reguliere TMap training (binnen het Philips opleidingscentrum is zelf een specifieke "TMap embedded software" training ontwikkeld), ook het ISEB testcertificatie programma uitstekend geschikt gebleken voor software- en testengineers werkzaam in een embedded omgeving.

Technieken

Testen kan met behulp van vele technieken. Er zijn technieken ter ondersteuning van het bepalen van de teststrategie, het opstellen van een testbegroting (o.a. testpuntanalyse (Van Veenendaal, 1995)) en het beoordelen van de documentatie. Bij het bepalen van de teststrategie dient te worden vastgesteld wat en met welke diepgang getest gaat worden. Er dienen keuzes te worden gemaakt aangezien het onmogelijk is om een softwareproduct volledig te testen. Binnen de fase voorbereiding wordt de documentatie (bijv. de software requirements specification of het global design) getoetst op kwaliteit en testbaarheid. De meest effectieve en efficiënte technieken voor het toetsen c.q beoordelen van documentatie zijn reviews en inspecties (Gilb en Graham, 1993). Met name binnen de technische automatisering worden inspecties, of een variant daarvan, veelvuldig toegepast. Echter, de belangrijkste groep testtechnieken wordt gevormd door de zogenaamde testspecificatietechnieken.

Testspecificatietechniek
een gestandaardiseerde manier om vanuit uitgangsinformatie testgevallen af te leiden. (TMap, 1995)

Op basis van de "uitgangsinformatie" moeten testgevallen worden bepaald. Voor het testen van de verschillende soorten kwaliteitseisen dienen verschillende testspecificatietechnieken met een verschillende diepgang beschikbaar te zijn. In de literatuur is een zeer groot aantal testspecificatietechnieken beschikbaar (Beizer, 1990, BS7925-1, 1998 en TMap, 1999). Vervolgens worden een aantal technieken beknopt beschreven die door de auteurs uitermate geschikt zijn bevonden voor het maken van testgevallen binnen technische automatisering.

Test coverage

Voor module testen worden veelal geen formele testspecificaties vervaardigd. Om desondanks de kwaliteit c.q. volledigheid van de uitgevoerde testen te kunnen bewaken, kan gebruik worden gemaakt van test coverage metingen (BS7925-1, 1998). Test coverage meet het percentage van de code dat is uitgevoerd tijdens het testen. Aangezien bij de module test de interne structuur van de code centraal staat, sluit het begrip test coverage goed aan bij deze test. Op basis van de resultaten van de test krijgt de ontwikkelaar/tester een beeld van welk percentage van de code is uitgevoerd tijdens de test. Dan kan ook worden bepaald of de test uitgebreid dient te worden en waar de zwakke punten (niet geraakte code) van de test liggen. Veelal wordt gestreefd naar een coverage percentage van 70% tot 80% tijdens module testen. Hoger percentages zijn veelal erg moelijk te behalen en duur, met name als gevolg van de "error-handling" code.

Beslissingstabellentechniek

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Veelal is het belang van een “foutloos” eindproduct binnen de technische automatisering erg groot, bijv. een software fout in een TV-range met 5.000.000 apparaten wereldwijd kan niet “zo maar” worden opgelost. Tijdens het testen dient derhalve een hoge dekkingsgraad te worden gehaald. Met de beslissingstabellentechniek (BTT) (Pol *et al*, 1999) kan dit worden gerealiseerd. In sommige projecten is hiermee zelfs een statement coverage van 95% gehaald (Van Veenendaal, 1999). Met de BTT worden een groot aantal situaties (combinaties van input parameters) afgedekt. Tevens is de BTT uitermate geschikt voor state-testing. (Voor state transition testing is ook een specifieke techniek beschikbaar, zie BS7925-2, 1998.) De BTT techniek is oorspronkelijk ontwikkeld voor white-box testen, maar wordt inmiddels ook veelvuldig toegepast tijdens het black-box testen.

Identificatie tabel					
Logische testkolom	1	2	3	4	5
Trigger	1	1	1	1	1
A > 0	1	0	0	0	1
B = 6	0	1	0	0	1
C = 2 EN D = 4 EN E =5	0	0	1	0	1
F < 0 OF G > 0	0	0	0	1	1
Resultaat 1	X				X
Resultaat 2		X	X	X	X

Figuur 3: Voorbeeld beslissingstabel

Equivalence Partitioning
Aangezien het veelal bijna onmogelijk is om alle delen van een softwareproduct op de meest uitgebreide wijze te testen, bijv. met behulp van de BTT, kan gebruik worden gemaakt van de equivalence partitioning techniek (BS7925-2, 1998). De equivalence partitioning techniek richt zich op het reduceren van het testgevallen door deze onder te brengen in zogenaamde klassen die hetzelfde gedrag zouden moeten vertonen. Voor iedere klasse hoeft dan slechts één waarde te worden getest. Vaak wordt deze techniek toegepast in combinatie met boundary value analysis (BS7925-2, 1998). Hierbij worden testgevallen bepaald op en om de grenzen van een equivalentieklasse. Dit zijn testsituaties die in de praktijk veel fouten veroorzaken. De praktijk leert dat boundary value analysis een zeer effectieve techniek is voor module testen. De equivalence partitioning techniek is uitermate geschikt voor het testen van interfaces tussen modules (integratietest) en/of softwarecomponenten.

Use cases
De meeste testspecificatietechnieken zijn gebaseerd op de functionele of technische beschrijvingen van de te testen software. Een andere benadering is vanuit gebruikersoptiek kijken of het systeem aan de behoeften en verwachtingen voldoet. Hiervoor kan gebruik worden gemaakt van use cases (Collard, 1999). De workflow van de gebruiker wordt hierin stapsgewijs en op eenvoudige wijze beschreven. De use cases komen tot stand door aan de gebruikers te vragen hoe ze een bepaalde taak uitvoeren. De code dekkingsgraad van use cases is echter vrij laag. Use cases zijn uitermate geschikt voor het validatietesten en/of als onderdeel van een regressietest.

Field test

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Veel technische softwaresystemen worden in een groot aantal omgevingen gebruikt. Alleen testen in een laboratorium omgeving is vaak niet voldoende, aangezien veel realistische situaties niet of zeer moeilijk te simuleren zijn. Bovendien is het, qua geld en tijd, onmogelijk om alle omgevingen te simuleren en testen. Het product zal op z'n minst in de meest voorkomende en/of meest kritische omgevingen getest moeten worden, tijdens de zogenaamde fieldtest, om het risico op fouten in het vrijgegeven product te beperken. Het product kan eventueel ook bij klanten uitgezet worden, de zogenaamde customer trial, waar het in de uiteindelijke "productie" omgeving getest wordt. Eventuele resterende fouten kunnen op deze wijze nog voor de uiteindelijk vrijgave worden verwijderd.

Infrastructuur & tools

Om tests te kunnen uitvoeren is een testomgeving nodig. Deze omgeving moet stabiel, beheersbaar en representatief zijn. Verder moet deze omgeving zijn afgescheiden van andere omgevingen (zoals de ontwikkelomgeving). Alleen onder deze voorwaarden is het mogelijk om reproduceerbare tests uit te voeren. Om de tests bovendien efficiënt te kunnen uitvoeren zijn hulpmiddelen ("testtools") noodzakelijk. Testtools kunnen worden onderscheiden naar de activiteiten (en dus testfase) die ze ondersteunen. Tenslotte is ook de werkomgeving (bijv. PC's, telefoon, e-mail) van het testpersoneel in het kader van de infrastructuur van belang.

Binnen de technische automatisering is het gebruik van testtooling reeds behoorlijk ingeburgerd, maar het toepassen van deze tooling gaat vaak met specifieke problemen gepaard. De sterke hardware afhankelijkheid bemoeilijkt het gebruik van standaard tools en verhoogd daarmee de implementatie inspanning. Een overzicht van de specifieke tooling die binnen de technische automatisering in praktijk gebruikt wordt en de bijbehorende kenmerken wordt hierna gegeven. Het onderstaande overzicht is opgesplitst in testtools die met name worden gebruikt door de software engineer ter ondersteuning van het white-box testproces en tools die worden gebruikt voor de systeem- en acceptatietest, het black-box testproces.

White-box testtools

Static code analyzer

Een static code analyzer gebruikt programmacode als invoer en voert daarop allerlei statische analyses en controles uit. Het doel is het signaleren van zaken zoals "onveilig" programmeren en foutgevoelige of slecht onderhoudbare code. De functionaliteit van een statisch testtool is globaal onder te verdelen in een viertal aspecten:

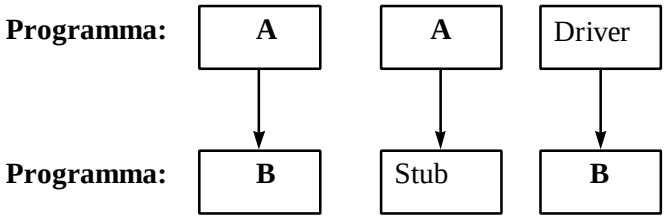
- Programmastructuuranalyse: door middel van deze structuuranalyses wordt een beoordeling gegeven van een programma, en/of subsysteem-architectuur;
- Coding standards: is de code ontwikkeld conform de geldende normen en standaards?;
- Style guide: voldoet de programmatuur aan de gehanteerde style guide (bijv. hoeveelheid commentaar en de wijze van inspringen)?;
- Code metrics: ten aanzien van bijvoorbeeld de complexiteit, omvang, aantal interfaces kunnen metrics worden gegenereerd die tesamen een kwaliteitsoordeel over de code geven.

In de praktijk wordt het meest gebruik gemaakt van de coding standards functionaliteit, bijv. voor C of C^{++}. Bij het toepassen van statische analyse tools dient rekening gehouden te worden met mogelijke specifieke embedded software varianten ten opzichte van de standaard programmeertaal. Deze varianten ondersteunen vaak processor specifieke functionaliteit die

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

weer niet door de static code analyzers wordt ondersteund. Met (arbeidsintensieve) aanpassingen in de software code en de juiste configuratie van het tool is dit probleem echter meestal op te lossen.

Stubs & drivers
Doordat bij technische systemen de hardware vaak gelijktijdig met de software wordt ontwikkeld, dient er vaak software code geschreven te worden die de nog ontbrekende hardware vervangt. Deze software code wordt een stub genoemd. Overigens worden stubs niet alleen gebruikt om ontbrekende hardware te vervangen, maar ze kunnen ook gebruikt worden om nog ontbrekende software componenten te vervangen. De stubs bieden doorgaans een beperkte functionaliteit aan, die voldoende moet zijn om de te testen software volledig te kunnen testen. Naast stubs dienen in deze context ook drivers te worden genoemd. Een stub wordt aangeroepen vanuit het te testen programma, een driver roept een te testen programma juist aan (zie figuur 4).



Figuur 4: Stubs en drivers

Test coverage tool
Om de eerder beschreven coverage techniek te ondersteunen zijn test (of code) coverage tools beschikbaar. Hier dient wel rekening mee gehouden te worden dat de meeste coverage tools zogenaamde ‘code instrumentatie’ toepast. Hierbij wordt door het tool extra software code toegevoegd om achteraf te kunnen bepalen welke software code uitgevoerd is en in welke mate. Voor technische software is dit echter vaak niet mogelijk, of zeer onwenselijk. Geïnstrumenteerde software code kan immers een ander gedrag gaan vertonen door mogelijke veranderingen in de timing of door de toevoegingen tegen een fysieke maximale code size grens aanlopen. De aanwezigheid van een maximale code size grens komt vooral voor bij embedded software (waarbij de software onderdeel is van het product) waar deze grens bepaald wordt door de gebruikte microcontroller. Voor deze embedded software zijn tools in omloop die de test coverage kunnen meten zonder gebruik te maken van code insertion, maar zij zijn dan wel microcontroller(-familie) specifiek.

Testontwerp
Een essentieel onderdeel bij het vervaardigen van testgevallen is het nadenken over het testontwerp. Wat moet er getest worden en hoe zal dat gaan gebeuren? Met name bij white box testen wordt het testontwerp vaak overgeslagen, omdat te arbeisintensief is en te veel documentatie oplevert. Een goed testontwerp is echter even belangrijk voor testen als een technisch ontwerp voor source code. Inmiddels zijn er een aantal goede testontwerp tools beschikbaar, veelal gekoppeld aan coverage tools. Een voorbeeld hiervan is xUnit-framework (http://www.xunit.org). Dit tool biedt de mogelijkheid om vanuit de ontwikkelomgeving testgevallen te specificeren in de vorm van source code en deze testen uit te voeren vanuit dezelfde omgeving. De testcode is in dit geval tevens de documentatie.

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Black-box testtools

Dynamische code analyzer
Om tijdens de testuitvoering het dynamisch gedrag van de software te analyzeren kan gebruik gemaakt worden van dynamische code analyzers. Problemen die met dit type tools gevonden worden, liggen op het vlak van geheugenproblemen (memory leaks, etc.) en problemen met de belasting (stress/load testen). Het nadeel van deze tooling is dat zij de prestaties van de software kan beïnvloeden, met name op het gebied van timing. Bij het testen van de software dient minimaal éénmaal zonder de aanwezigheid van het tool getest te worden om zo het werkelijke gedrag te kunnen beoordelen.

Record & playback tool
Binnen de administratieve softwaresystemen wordt veelvuldig gebruik gemaakt van een record & playback tool. Een dergelijk tool registreert de handelingen van de tester tijdens het uitvoeren van een test en kan deze handelingen naderhand, zonder dat de tester hierbij aanwezig hoeft te zijn, herhalen. Op deze manier kunnen regressietesten, in principe, op eenvoudige wijze worden uitgevoerd.
Bij technische software zijn deze tools vaak niet zonder meer toe te passen door het feit dat een duidelijke, generieke user interface vaak ontbreekt. Zelfs als een user interface aanwezig is, dient het betreffende tool veelal te worden aangepast aan de specifieke software en hardware omgeving. Het record & playback tool dient meestal aangepast te worden aan de omgeving waarin deze gebruikt zal gaan worden. Het is maar de vraag of de tijd die hierin gestopt moet worden, zal worden terugverdiend tijdens het automatisch uitvoeren van de tests. Veelal worden binnen de technische automatisering zelf tools ontwikkeld, die record & playback functionaliteit bieden.

Simulator
Met behulp van simulatoren kan niet beschikbare hardware door middel van software gesimuleerd worden. Het verschil met de eerder genoemde stubs en drivers is dat simulatoren de volledige functionaliteit, of op z'n minst een groot deel ervan, van de hardware aanbieden. Simulatoren worden veelal ook gebruikt om de testbeoordeling beter te kunnen uitvoeren. Dit gebeurt door de testresultaten met behulp van, eventueel extra toegevoegde functionaliteit, in detail te loggen. Bij hardware is deze logging vaak erg moeilijk, zoniet onmogelijk te realiseren.

Stand van zaken
Om een meer inzicht te geven in het gebruik van testtools in de praktijk, is enige data opgenomen die met betrekking tot de implementatiegraad van de verschillende tools. De data is afkomstig uit een toolonderzoek uitgevoerd door Improve in de periode 2000-2001 bij ruim 200 Europese IT-bedrijven werkzaam in de embedded c.q. technische automatisering.

Testtool	*Implementatie ratio*
Statische analyse	22%
Coverage tools	17%
Test ontwerp	18%
Dynamische analyse	19%
Record & playback	29%
Simulators	51%

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Tabel 1: Testtool implementatie status

Conclusie

De vier pijlers van gestructureerd testen zoals besproken in dit artikel, zijn duidelijk niet slechts van toepassing bij administratieve automatisering, maar tevens bij het testen van embedded software. Zelfs bij de concrete invulling kan veelvuldig gebruik worden gemaakt van beschikbare testmethoden, -technieken en -tools, zoals bijv. gedefinieerd door TMap. Bij het daadwerkelijk toepassen en implementeren dienen de juiste "practices" te worden geselecteerd die passen bij de specifieke embedded c.q. technische software omgeving. Deze laatste stap blijkt in de praktijk vaak moeizaam te verlopen omdat hiervoor zowel uitgebreide kennis van gestructureerd testen als van de technische omgeving noodzakelijk is. Veelal worden testers ook op het verkeerde been gezet door de terminologie die wordt gehanteerd. Door echter de diverse voorhanden zijnde testmethoden en -technieken gedetailleerd en gedegen te bestuderen kunnen de "practices" worden geselecteerd die van toegevoegde waarde zijn. Dit artikel beoogt een eerste aanzet te geven tot een selectie van bruikbare "practices". Aan de lezer de uitdaging om een en ander voor zijn/haar omgeving verder te verdiepen.

Literatuur

Beizer (1990), *Software Testing Techniques*, International Thomson Computer Press

BS 7925-1 (1998), *Software Testing – Vocabulary*, British Standards Institution

Collard, R. (1999), Testdesign, developing testcases from use cases, in*: Software Test & Quality Engineering*, July/August 1999

Gilb T. en D. Graham (1993), *Software Inspection*, Addison-Wesley, Harlow, England

Kit, E. (1995), *Software Testing in the real world*, Addison-Wesley, Reading, Massachusetts

Pol. M., R.A.P. Teunissen en E.P.W.M. van Veenendaal (1999), *Testen volgens TMap (2e druk)*, Tutein Nolthenius, 's Hertogenbosch

Rooijmans J., H. Aerts en M. van Genugten (1996), Software Quality in Consumer Electronic Products, in: *IEEE Software*, January 1996

Veenendaal, E. van (1999), Practical Quality Assurance for Embedded Software, in: *Software Quality Professional*, Vol. 1, Issue 3, June 1999, pp. 7-18

Veenendaal, E. van (1995), Test point analysis: a method for test estimation, in: *Computable*

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl

Auteursgegevens

Drs. Erik P.W.M. van Veenendaal CISA (Improve Quality Services BV) is reeds een groot aantal jaren werkzaam binnen het vakgebied software kwaliteit. Hij heeft daarbinnen een specialisatie ontwikkeld op het gebied van testen en is co-auteur van onder andere "Testen volgens TMap". Als testmanager en adviseur is hij betrokken geweest bij een groot aantal automatiseringsprojecten. Tevens heeft Erik bij een aantal omvangrijke organisaties meegewerkt aan teststructurering en test process improvement. Hij spreekt regelmatig op zowel nationale als internationale conferenties en is een internationaal gerespecteerd docent op het gebied van software kwaliteit en testen (o.a. ISEB geaccrediteerd). Momenteel voert hij de directie van Improve Quality Services BV, een dienstverlenende organisatie op het gebied kwaliteitsmanagement, usability, testen en inspecties (www.improveqs.nl).
Als universitair docent is Erik part-time verbonden aan de faculteit Technology Management van de TU-Eindhoven. Tevens is hij lid van de Nederlandse standaardisatie commissie ten aanzien van software kwaliteit en was hij mede-initiatiefnemer voor de oprichting van TestNet (de Nederlandse vereniging van testers).

Ing. Rob Hendriks heeft meer dan 6 jaar ervaring op het gebied van software kwaliteit, met de nadruk op software testen voor embedded en technische software systemen. De afgelopen jaren is hij werkzaam geweest als testcoördinator en -adviseur binnen projecten voor consumenten elektronica en professionele systemen. Hij verzorgt regelmatig cursussen op het gebied van inspecties en testen (o.a. ISEB geaccrediteerd). Rob is werkzaam als testconsultant binnen Improve Quality Services BV.

Improve Quality Services BV, Waalreseweg 17, 5554 HA Valkenswaard
Tel: 040-2089283, Fax: 040-2021450, E-mail: info@improveqs.nl