juni 2003

Nieuwsbrief van de vereniging TestNet

TNN

Van de redactie

Vakantie! Nog even en de meesten van ons gaan lekker een paar weken genieten van een welverdiende rust. Even niet aan testen denken maar gewoon genieten van zon, zee, water, natuur en rust. Maar ja er zal nog wel even een TestNet Nieuws doorgewerkt moeten worden voor de vakantie. De zachte kant van het testen staat in Erik's Column en Hans van Loenhoud besteedt aandacht aan de professionalisering van onze vereniging. Er zijn verslagen van een thema-avond (Agile methoden) en een stuk uit de Automatiserings Gids over het voorjaarsevenement met dank aan Freek Blankena. Dat is natuurlijk weer voldoende leesvoer zo vlak voor onze vakantie. En natuurlijk verwachten wij u op de algemene ledenvergadering van 26 juni in Nieuwegein.

In dit nummer

Van de Voorzitter

Door Hans van Loenhoud

Beter

TestNet gaat het beter doen! Dat is nog eens een pakkende opening van een stukje voor TestNet Nieuws. 'Hoezo beter?' vraagt u zich wellicht af, 'deed TestNet het dan niet goed?'

Zonder verwaand te zijn: ik vind dat TestNet het best goed doet. Onze evenementen en thema-avonden worden druk bezocht en goed gewaardeerd. De werkgroepen kennen een vaste kern van trouwe deelnemers en leveren regelmatig vernieuwende bijdragen aan het vakgebied. TestNet Nieuws wordt veel gelezen en onze site druk bezocht. Allemaal zaken die in de afgelopen zes jaar zijn opgebouwd en intussen een vaste plek in de wereld van software testen hebben veroverd.

Toch zijn er ook zaken die nog flink kunnen worden verbeterd. Daarbij doel ik met name op wat er zich achter de schermen afspeelt, onze back-office. Daar lopen we een beetje tegen de handicap op, dat al het werk moet worden gedaan door goedbedoelende amateurs, die het er even in de avonduren bij doen naast een drukke baan als tester: werkgroeptrekkers, TNN-redactie, commissieleden, bestuur en al die anderen die door hun actieve inzet TestNet hebben gemaakt tot wat het nu is. Ik ben enorm blij en dankbaar voor het vele werk dat wij als leden op die manier voor de vereniging en dus voor elkaar doen. En ik ga ervan uit dat wij dat ook de komende tijd voor elkaar willen blijven doen.
Tegelijkertijd valt te constateren dat back-office activiteiten nu niet direct zulke attractieve werkzaamheden zijn, dat er gelijk hele horden leden staan te dringen om ze aan te pakken. Een mailing voor een door TestNet ondersteunde conferentie verzorgen houdt in dat er:

- 600 enveloppen moeten worden geadresseerd;
- 600 brieven gevouwen;
- 600 folders in enveloppen gestopt;
- 1200 postzegels geplakt en;
- 2 dozen vol post naar het postkantoor wordt gesjouwd.

Leuk? En het permanent up-to date houden van de ledenadministratie is ook niet iets waar je op een familiefeestje de blits mee kan maken…

In het verleden, bij een kleiner TestNet, was het nog wel bij te houden. Door de groei van het ledenaantal (voor mij nog steeds het beste bewijs dat we op de goede weg zijn) wordt het echter steeds moeilijker om op een niveau te opereren dat kwaliteitsbewuste testers acceptabel vinden.
Gevolg: fouten bij de inning van de contributie, mailtjes die te lang onbeantwoord blijven, adreswijzigingen die spontaan kwijt raken en posters die later worden verzonden dan de bedoeling was. Mocht u daar in de afgelopen tijd last van hebben gehad: sorry, had natuurlijk niet mogen

gebeuren, maar ja, soms lopen dingen nu eenmaal niet precies zoals je het zou willen.

En – zo kom ik weer terug bij de opening van dit stukje – voortaan wordt het allemaal beter. Onlangs heeft TestNet een contract gesloten met een administratiekantoor, In Capo uit Best. In Capo gaat voor ons de activiteiten die betrekking hebben op ledenadministratie, contributie-inning, boekhouding en mailings verzorgen. Op basis van een gedetailleerde SLA is In Capo vanaf 1 juni voor ons aan de slag gegaan. Oké, het kost een paar centen, maar dan heb je ook wat: een professionele back-office voor TestNet.

Door het overnemen van deze activiteiten kunnen onze actieve redactie-, commissie-, werkgroep- en bestuursleden zich weer volledig concentreren op wat ze echt leuk vinden: vakinhoudelijk bezig zijn met software testen dus.
Let op mijn woorden, dat moeten jullie als leden gaan merken, zowel door een snellere en meer accurate afhandeling van allerlei administratieve zaken, als door een nog betere performance op vakinhoudelijk gebied. Laten we afspreken dat jullie me over een half jaartje of zo een mailtje sturen en dan aangeven of deze voorspelling is uitgekomen. Ik ben benieuwd.

Iets heel anders tot slot: voor velen staat de zomervakantie al weer voor de deur. Test daarin de camping, de tour-operator of de wandelschoenen. En vertel elkaar op de eerstvolgende TestNet avond de bevindingen, dan kunnen we er allemaal van meegenieten. Veel plezier toegewenst!

Van de secretaris

Door Marco Jansen van Doorn

Onze voorzitter heeft al ruim voldoende aandacht besteedt aan het outsourcen van onze back-office, dus daar zal ik geen woorden meer aan vuil maken. Maar er is natuurlijk nog een ander belangrijk punt n.l. de ALV.

Op 26 juni 2003 wordt in het NBC in Nieuwegein de Algemene Leden Vergadering georganiseerd. In deze vergadering zal het bestuur terugblikken op het jaar 2002 en vooruitblikken op (de rest van) 2003. InCapo zal ook aanwezig zijn om, waar nodig of interessant, toelichting te geven. De aanwezigheid van de leden op de ALV is uiteindelijk natuurlijk het belangrijkst, wij hopen dan ook u op de ALV te ontmoeten.

Testen in agile methoden

Door Harco op den Kelder

Inleiding
Op deze bijeenkomst konden de deelnemers kennismaken met een aantal bekende agile methoden en de rol van testen daarin. De opzet van de avond was geheel in stijl: **agile**. Na een korte inleiding werd de avond ingedeeld in blokken (iteraties). In elk blok werd een onderwerp gepresenteerd door Anko Tijman. Vervolgens werden in groepjes van 6 man/vrouw de verschillen tussen testen in traditionele ontwikkelmethoden ten opzichte van agile methoden in kaart gebracht. De resultaten werden voor de gehele groep teruggekoppeld. Oftewel: vertellen, vertalen, en verhalen. En alles in een strak tijdschema (time-boxing).

Agile manifesto
Wat is agile eigenlijk? Het agile manifesto is opgezet door “17 wijze mannen in een blokhut”. Ze staan een manier van software ontwikkelen voor die meer belang hecht aan:

- individuen en interacties dan processes en tools;
- werkende software dan begrijpelijke documentatie;
- samenwerking met klant dan contractonderhan-delingen;
- inspelen op veranderingen dan volgen van een plan.

Met andere woorden: Too much process will kill you.

Agile methoden
De volgende agile methoden zijn aan bod gekomen:

1. Extreme Programming (XP). Gericht op programmeren en testen (Test First). Het principe XP is opgebouwd rondom vier algemene waarden: Communicatie, Eenvoud, Terugkoppeling en Moed. Rollen in XP zijn klant en ontwikkelaar.
2. Crystal. Gericht op de inrichting van het project. Het is mens- en communicatiegericht, incrementeel en schaalbaar.
3. SCRUM. Gericht op projectleiderschap in agile omgevingen. In het kort: verzamelen van taken die in het project verricht moeten worden, plannen van taken voor de komende periode (meestal 1 maand), uitvoeren zonder tussentijdse wijzigingen, evaluatie + schone lei voor volgende periode.

Rol van de tester
Welke rol kan de tester zich

toe-eigenen in een agile omgeving? Uit de groepsdiscussie is naar voren gekomen dat de rol van de tester wat meer flexibel moet zijn. Er is een nauwere samenwerking tussen tester, klant en ontwikkelaar. De tester zal meer betrokkenheid en materiekennis nodig hebben. Bij XP zou de tester een facilitator of één van het pair moeten zijn, anders vervalt zijn rol.
Ook de inhoud van het testen is anders. De tester zal meer worden ingezet bij acceptatietests. Activiteiten richten zich als gevolg van de iteratieve werkwijze meer op herhaalbaarheid. Er zal meer testautomatisering gebruikt worden.
Door het gebruik van agile methoden is de functionaliteit variabel en krijgt de tester een adaptieve werkwijze die vooral geschikt lijkt voor nieuwe projecten, niet voor onderhoudsprojecten. De agile conventies laten een korte voorbereiding per iteratie toe. Volgens de deelnemers is de agile methode toepasbaar voor prototyping, time-boxing, prioritering en workshops.

Conclusie
Door de interactieve opzet van de avond hebben de leden een praktisch gevoel kunnen krijgen bij de achterliggende gedachte van agile. Inclusief de moeilijkheid van timeboxing. In de kleine groepen kon iedereen een actieve bijdrage leveren. Rest mij Anko te bedanken voor de voorbereiding en presentatie van deze avond. De avond vond ik zeer geslaagd en de interactieve opzet is zeker voor herhaling vatbaar.

Erik's Column

Soft Skills: de "vergeten" vaardigheden?

Door Erik van Veenendaal
eve@improveqs.nl

De inhoud van deze TestNet Column is met name ingegeven door een aantal recente praktijkervaringen. Steeds vaker (en terecht!) hoor je organisaties praten over professionalisering van het testvak, functieprofielen, testcarrièrepaden, ISEB testcertificering etc. Kortom, ons vakgebied wordt steeds meer volwassen en we worden erkend. Kijkend naar de meeste functieprofielen, opleidingsplannen en naar de inhoud van testopleidingen dan gaat het bijna altijd over testonderwerpen zoals strategie, planning, technieken (TMap) en automatisering (TestFrame). Een goede tester en testmanager hebben dan ook een gedegen kennis van dit soort methoden en technieken en gaat daarmee in de praktijk aan de slag.

In de meeste projecten wordt impliciet veel meer gevraagd van de testprofessional. Het wordt immers wel eens spannend aan het einde van een project; het project loopt uit, het budget wordt overschreden, de gebruikers zijn niet tevreden met de geboden functionaliteit, etc. Allemaal redenen voor enige frictie tussen de opdrachtgever (gebruikersorganisatie) en ontwikkeling. De tester staat dan vaak in het midden, soms ietwat meer aan gebruikerskant (acceptatietest), soms ietwat meer aan de ontwikkelkant (systeemtest). In dit soort situaties moet de tester op een 'goede' wijze kunnen communiceren over de probleemrapporten, dient een testrapportage te worden geschreven waarbij de woordkeuze van groot belang is, moet een presentatie worden gegeven over de stand van zaken ('vrijgave advies') aan de stuurgroep en moet worden deelgenomen aan politiek zeer beladen meetings. Allemaal activiteiten waarvoor bepaalde sociale en communicatieve vaardigheden uitermate noodzakelijk zijn. Dit zijn zeer zeker geen trivia, maar complexe omgevingen waarin de testprofessional moet kunnen 'overleven'.

Een goed en gedegen testopleidingsplan dient naast de testinhoudelijke zaken, ook ruim aandacht te hebben voor adviesvaardigheden, schriftelijk communicatie, presentatietechnieken, het schrijven van adviesrapporten, en algemene communicatieve vaardigheden. Mijn persoonlijke ervaring is dat deze aspecten, de zogenaamde 'soft skills' veruit onderbelicht zijn bij de meeste testcarrièrepaden. Ik bedoel hiermee dat structureel trainingen (2 à 3 dagen per onderwerp) worden gevolgd en coaching plaatsvindt ten aanzien van deze onderwerpen. Ook in de reguliere testopleidingen zou enige aandacht voor dit soort aspecten niet misstaan. Voor zover mij bekend, wordt hieraan alleen bij ISEB Practitioner enige aandacht (zo'n 4 uur) besteed.

Natuurlijk is het zo dat je bij een aanname beleid al kunt kijken wat voor karakteristieken iemand met zich mee draagt. Iemand die het testvak in wil, zou eigenlijk al enige aanleg moeten hebben voor goede communicatie en sociale vaardigheden. Het verder ontwikkelen hiervan is mijns inziens een stap die, gezien onze rol in projecten, de nodige aandacht moet (gaan) krijgen. Alleen als we dat goed beheersen, kunnen we onze goede inhoudelijke testkennis optimaal benutten en ervaart de opdrachtgever de meerwaarde van de testprofessional ten volle. In de praktijk gaat het helaas nog wel eens mis, en kan een hele hoop ellende worden voorkomen door een gedegen bagage van 'soft skills'. Op weg naar een verdere professionalisering van het testvak!

Voor vragen of een reactie kunt u mailen met Erik van Veenendaal

Testers voelen zich nog niet echt gewaardeerd

Door Freek Blankena
F.Blankena@WKTHS.NL
Automatiseringsgids 2003, week 16

Softwareontwikkelaars maken de stomste fouten en softwaretesters zeuren over dingen die er niet toe doen. Dat is het wederzijdse beeld dat beide beroepsgroepen van elkaar schetsen. Op een bijeenkomst van Testnet bleek in ieder geval dat testers zich nog niet echt serieus genomen voelen. Maar gebruikers, applicatiebeheerders en vooral directies ontdekken langzamerhand het belang van goed testen.

"Software ontwikkelen is een heel eenvoudig proces. Neem een goed idee, voeg bugs toe en lever het uit." Het was een semi-schertsende uitlating van testgoeroe Les Hatton op een bijeenkomst van TestNet. De professionals die zich wijden aan het testen van software voelen zich kennelijk genoodzaakt af en toe van zich af te bijten, want het zit nog niet helemaal goed met de waardering die ze krijgen, vooral van de zijde van de ontwikkelaars.
Hans van Loenhoud, voorzitter van de beroepsvereniging TestNet, bevestigt het beeld dat de testers zich niet helemaal serieus genomen voelen. "In het spanningsveld tussen ontwikkelen en testen acht de gemiddelde ontwikkelaar zich wat hoger gepositioneerd dan de tester, omdat de ontwikkelaar een creatief proces doormaakt waarbij hij iets nieuws maakt, terwijl de tester iets wat al bestaat aan de tand moet voelen. In de praktijk van alledag zie je dat bij veel organisaties de waardering voor softwaretesters vaak een niveautje lager ligt dan die voor ontwikkelaars. Kijk gewoon naar salarissen en dergelijke. TestNet vindt dat een volstrekt onterechte situatie."
Er zijn wel wat verschuivingen de laatste tijd, constateert Van Loenhoud. "Je ziet dat de business er steeds meer genoeg van heeft dat software de beloften niet waarmaakt die ontwikkelaars en verkopers van software doen. Bedrijven stellen steeds vaker de eis dat een ontwikkelaar of verkoper bewijst dat de software doet wat hij moet doen. De waardering voor het testvak groeit dus langzamerhand omdat bedrijven zekerheid willen dat software geen risico inhoudt."
In plaats van de gemankeerde ontwikkelaars die dan maar de nare testklusjes moesten opknappen, zijn testers nu professionals geworden die de bedrijfsrisico's van software beperken. Die professionalisering wordt veelal gestimuleerd door de gebruikers, meent Van Loenhoud. "Gebruikers, applicatiebeheerders en met name businessmanagers zeggen 'wij zijn het zat om software met bugs te krijgen'. Zeker in bijvoorbeeld de vliegtuigwereld is softwaretesten een gerespecteerd vak." En ook aan de embedded softwarekant komt men er volgens Van Loenhoud achter dat software vol met bugs zit.
"Hoe toon je het nut van testen aan?" was de vraag voor Henk van Merode van de KLM. Hij moest met zijn afdeling Test Management een nieuw systeem voor het luchtpostvervoer testen dat nog voor kerst 2002 moest draaien. "Het moest goedkoop en snel", zo luidde de opdracht. KLM moest de eerste gebruiker worden van een standaardproduct van een externe leverancier. Maar in De Merodes ervaring moesten de testers nogal vechten voor erkenning van het nut van hun werk. Slechts in kleine stappen kon er mankracht vrijgemaakt worden voor het testen, uiteindelijk 2,7 FTE voor langere tijd. De Merode spreekt van 'wederzijds wantrouwen tussen business en IT'. Aanvankelijk kregen de testers ook relatief meer de schuld van de opgetreden vertragingen. "Er gold een harde deadline, maar de business stelde niet genoeg middelen ter beschikking." Uiteindelijk ontstond er meer teamwerk. De Merode zegt te hebben

geleerd dat de complexiteit van een goed testproces doorgaans wordt onderschat door 'de business', de opdrachtgevers. "Je moet betrokkenheid krijgen van die klant, dus je moet wel eens dingen doen die niet bij je eigen werk horen." Dat kan betekenen dat testmanagers niet alleen testcases moeten bedenken, maar ook businesscases, iets waar ze minder affiniteit mee hebben. Daarnaast is het zaak te zorgen voor een paar 'quick wins', kleine succesjes waarmee het nut van de testprocedure is aan te tonen.

Inhaalslag

Dat de waardering voor testers achterblijft, vindt Van Loenhoud ook weer niet zo vreemd. "Het testen als professie bestaat eigenlijk nog maar kort. Het uitkomen van de methode TMap was het moment dat het testen voor het eerst als vak werd erkend. Dat is nog geen tien jaar geleden. De testers zitten in een inhaalslag en die is nog niet achter de rug."
Van Loenhoud hoort 'in de markt' van bedrijven als Sogeti, Logica CMG en Ordina wel dat testactiviteiten relatief in belang toenemen. "Verschillende grote generieke dienstverleners die ook een testpoot hebben, roepen allemaal 'onze testpoot doet het op dit moment relatief veel beter dan onze ontwikkelpoot'. De klanten achten het belang van testen steeds hoger." Hij denkt dat er relatief meer tijd wordt gestoken in het testen van een beperkter aantal ontwikkelde applicaties. Binnen ontwikkelprojecten verschuift de verhouding tussen ontwikkelen en testen. "In plaats van 10 of 20 procent van de tijd vergt die fase nu 30 of 40 procent van de tijd."

De Britse testwetenschapper Les Hatton betoogde op de bijeenkomst dat de meeste fouten in software 'langzame fouten' zijn. "Veel softwarefouten doen er erg lang over aan de oppervlakte te komen, of doen dat nooit." Juist die vertraging zorgt voor schade. Hij illustreert zijn stelling met een experiment met olie- exploratiesoftware, die door negen oliemaatschappijen wordt gebruikt om de succeskans van dure proefboringen in te schatten, op basis van seismische data. "Zelfs na vele gebruiksjaren blijken dezelfde algoritmes en dezelfde parameters te leiden tot uiteenlopende beslissingen over wel of niet boren, terwijl dezelfde seismische data gebruikt zijn."
Hatton heeft nog één tip voor testverantwoordelijken. "Als je software vrijgeeft, houd dan zelf niet op met zoeken naar fouten, want dat doe je beter dan je klanten. Bij client software is dat nog veel sterker het geval. Mensen raken steeds ongevoeliger voor GUI-fouten. We nemen het voor lief als de computer de stomste dingen tegen ons zegt."

Publicaties door leden

Recentelijk gepubliceerd door TestNet leden in Professional Tester

Making Quality Assurance work - A set of practical recommendations

Door Vivian Van Gansewinkel, Erik van Veenendaal en Mark van der Zwan

In practice quality assurance is often perceived as being theoretical and having little or no added value. This is especially true for the engineer's opinion. Vivian van Gansewinkel, Erik van Veenendaal and Mark van der Zwan (Improve Quality Services ltd.) have been working in the area of quality assurance for many years and share their practical experiences on lessons learned. They are convinced quality assurance does have added value provided the right approach is taken. The recommendation and practical experiences that are discussed in this paper are mainly derived from implementing TMM and the CMM key process area "Software Quality Assurance" in industrial organizations.

Het artikel is te downloaden via www.improveqs.nl / artikelen

When can the software be released? Software product development from an economic perspective

Door Hans Sassenburg
hsassenburg@se-cure.ch

Paper presented at the European SEPG, June 17th, 2003 (London)

Abstract

In this paper, a conceptual model is presented, offering a framework to steer software product development from an economic perspective. Through a series of mini-cases, in various organizations, it was found that the determination of a satisfactory moment to release a software product is a problem that most organizations struggle with. In the first place, the business case as the initial rationale for a project stays insufficiently aligned with the actual status of the project as it progresses. Secondly, at crucial decision

points, there are many uncertainties and comparing and evaluating the different alternatives is performed limited by the time available. Finally, there is a lack of proper evaluation of the business case results and the correctness of the various decisions made. Together, these prevent organizations from improving their capabilities in this area. The conceptual model presented combines existing methods and techniques into an overall framework. It helps organizations to build an economic case for their projects and control the progress of these projects. It supports the decision-making process to determine a satisfactory moment to release a software product.

Het complete artikel kunt u van onze site downloaden. Het artikel staat in **Bibliotheek > Rapporten**.

Thema-avond 12 juni Open Source en Testen

Door Brian Kemble
brian.kemble@mail.ing.nl

Op donderdag 12 juni j.l. was er een thema-avond georganiseerd over de inzet van Open Source Software en hoe als testers hiermee om te gaan. Gebleken is dat meerdere bedrijven Open Source toepassen of dit overwegen. Eén van de vragen die dan wel eens worden gesteld is of er wellicht (aangepast) testbeleid nodig is.
Het doel van de thema-avond was, antwoord te krijgen op de vraag of er ander/aangepast testbeleid nodig is voor het testen van Open Source Software.

Tijdens deze thema avond zijn er 2 presentaties verzorgd:

- Harrie Vollaard van RaboFacet over “Open Source Software binnen de Rabobank”
- Rix Groenboon van Reasoning over "Open-source Versus Commercial Software: A Quantitative Comparison "

Tijdens de presentatie van Harrie Vollaard werd stilgestaan bij hoe Open Source Software wordt ingezet bij de Rabo Bank. Op het gebied van Open Source Software is de Rabo Bank in de financiële wereld een van de voorlopers. De spreker stond kort stil bij de historie van Open Source Software
De voordelen en nadelen van het gebruik van Open Source Software ten opzichte van Vendor software kwamen aan bod.

Kenmerkend voor de Open Source Software is het feit dat iedereen bij de source code kan en modificaties hierop kan aanbrengen. Ook opvallend is de kennis en kunde die via de Internet community verkregen kan worden. In principe kan er dus 7 x 24 uur van de kennis van de community gebruik gemaakt worden.

De algemene claim is dat de Open Source Software van hogere kwaliteit is of zal blijken te zijn. Dit vanwege het feit dat de programmeurs die aan de software werken een hoger niveau hebben dan programmeurs die intern binnen software ontwikkelorganisaties werkzaam zijn.

Wat ook nadrukkelijk naar voren kwam is dat de kosten m.b.t. de licenties voor Open Source Software goedkoper uitvallen dan de kosten voor Vendor Software.

Binnen RaboFacet zijn er een aantal succesvolle pilots voor Open Source Software uitgevoerd. Een hiervan is open LDAP. De gemodificeerde source code is door RaboFacet weer aan de community aangeboden.

Met betrekking tot het testen kwam naar voren dat er in principe weinig verschil is tussen het testen van Vendor software en het testen van Open Source Software. Wel is het zo dat er met name tijdens de ontwikkeling van/ met behulp van Open Source Software, meer nog dan bij de ontwikkeling van Vendor Software, sprake is van peer reviews.

De volgende ervaringen zijn door de testers bij de Rabobank opgedaan;

- goede en snelle ondersteuning van de Internet community;
- risico van zelf programmeren;
- makkelijk te installeren en benaderen;

De tweede spreker Rix Groenboom, werkzaam bij Region, borduurde voort op het verhaal van de Harry Vollaard. Region, is met name gespecialiseerd in het leveren van diverse testservices.
Een van de services is het geautomatiseerd inspecteren van source code. Hierdoor kunnen er kwaliteitsvergelijkingen gemaakt worden tussen Vendor software en Open Source Software
Tijdens deze presentatie werden er cijfers van de diverse inspecties op Open

Source Code en Vendor source code getoond.

Conclusie: Interessante presentaties die meer inzicht verschaffen in het begrip Open Source Software. Het testbeleid zelf bleef echter wat onder belicht. Tussen de regels door kwam wel naar voren dat het testproces op zich niet ingrijpend veranderd. Wel dient er meer aandacht aan reviews en inspecties besteed te worden.

How Open-Source and Commercial Software Compare

A Quantitative Analysis of TCP/IP Implementations in Commercial Software and in the Linux Kernel

By Rix Groenboom
rix.Groenboom@reasoning.com.

Introduction
Reasoning Inc. provides an automated software inspection service that is used by leading commercial software vendors to identify defects and provide metrics regarding the quality of the inspected code. Our inspection service is based on a combination of technology and a repeatable process, and enables us to maintain a database of metadata about code quality. This database provides a unique opportunity to independently assess the quality of software.

What is Open Source and why might it be better?
Most commercial software vendors distribute their products in the form of executable or object code. Their customers do not acquire a license to use the source code, so they cannot change or extend the functionality of the executables, except by specific arrangements with the vendor. They are generally prohibited from redistributing a changed or extended version to others. With few exceptions, customers of commercial software vendors must rely on the vendor to make changes and extensions.
Open source software represents a fundamentally different way in which software is developed, sold, and maintained. For example, the source code can be modified by many people without the need for those people to be employed by the same software vendor.
Open source proponents believe that, for important pieces of software, the open source model encourages several activities that are not common in the development of commercial code:

- Many users don't just report bugs, as they would do with commercial software, but actually track down their root causes and fix them;
- Many developers are reviewing each other's code, if only because it is important to understand code before it can be changed or extended. It has long been known that peer review (inspection) is the most effective way to find defects;
- The open source model encourages programmers to organize themselves around a project based on their contributions. The most effective programmers write the most crucial code, review the contributions of others, and decide which of these contributions are incorporated into the next release;
- Open source projects don't face the same type of resource and time pressures that commercial projects do. Open source projects are rarely developed against a fixed timeline, affording more opportunity for peer review, and usually offer extensive beta testing before "release."

For these reasons, open source enthusiasts claim that the open source model produces better quality software than commercial software development.

Software inspection
Software inspection - the process of examining source code to identify defects – is a standard practice in development organizations and is widely recognized as the best way to find defects. Inspection is hardware-independent, does not require a "run able" application nor a suite of test cases, and does not affect code size or execution speed. But until recently, it has been a manual process – very slow, and very costly - or tools-based and hard to implement effectively.
The majority of code inspections are performed manually. Although a human reading the code line-by-line can theoretically uncover the greatest number of defects, the process is slow, painstaking, and fraught with inconsistency. Also, this approach does not scale to handle today's multi-million line applications. As a code base grows, the cost of a complete manual inspection becomes prohibitive and the volume of code is intimidating to developers. In practice, manual inspections are only

performed on subsets of the source code.
Inspection tools are able to perform only a portion of the inspection process, requiring significant further manual review. The inspections tools generate a large volume of defect 'warning messages' many of which are, in fact, false positives. The inspection tool "thinks" it has found a defect, but a deeper manual analysis of the context shows that the reported issue is not actually a defect. This false positive problem is very severe. Frequently, the rate will exceed 50 false positives to each true positive; in other words, only 2% of the warning messages represent defects.

Reasoning's automated software inspection service
Our automated software inspection service provides many of the benefits of a manual code review in significantly less time and at dramatically lower cost than manual inspection or internal use of inspection tools. With the service, in-house resources are not diverted from current development projects. We identify defects that cause application crashes and data corruption, and provides actionable reports. The error classes in C and C++ include:

- Memory leak: Reference to allocated memory is lost;
- NULL pointer dereference: Expression dereferences a NULL pointer;
- Bad deallocation: Deallocation is inappropriate for type of data;
- Out of bounds array access: Expression accesses a value beyond the array;
- Uninitialized variable: Variable is not initialized prior to use.

The output of the inspection is a set of reports that:

- Make defect analysis fast and simple by identifying the location and describing the circumstances under which the defects will occur;
- Identify the parts of the code with the greatest risk, enabling the development organization to focus QA and testing resources where they are most needed;
- Compare the customer's code quality with a benchmark (related to other inspections done by us).

The Study and Methodology
Of the thousands of open source applications available, we chose the Linux® operating system. This general-purpose operating system has been under development for nearly a decade, is widely used and is actively maintained and enhanced by a community of thousands of programmers. However, comparing the quality of several entire operating systems is a challenge, primarily because the size, scope and goals can be so different. Instead, we chose a common function implemented by all the projects in our study, the TCP/IP network protocol "stack". There were several reasons for this decision. This protocol is usually in the operating system "kernel", the lowest level software in the system; thus defects can have a major impact, including inability to communicate, system crashes, network outages, and security violations.
Each project was inspected using our standard automated software inspection process.

The commercial projects
We have conducted inspections of five different commercial TCP/IP implementations, including implementations from both general-purpose operating systems and embedded applications. Four of the five implementations are considered mature, having been in commercial use for over ten years (although the TCP/IP code is under active development). The fifth is relatively young: it was started about three years ago.
The size of these projects ranges from 64 KLSC to 269 KLSC. For reasons of client confidentiality, we cannot disclose further information about these projects.

The Linux inspection
We inspected the TCP/IP implementation in version 2.4.19 of the Linux kernel. We chose this version because it was the latest "stable" release at the time of the study. The TCP/IP code was inspected in isolation from the rest of the kernel, using the exact same process we use for customer projects.
The open source TCP/IP implementation includes 166 source files with just under 82 thousand lines of source code (KLSC) in size, not including user include files, header files, blank lines and comments. We found 8 defects, resulting in a defect density of 0.10 defects/KSLC.
Comparison Results
The table below summarizes the results for the five inspection classes.

Error class:	Com-mer-cial:	Open Source:
Memory leak	43	1
NULL pointer dereference:	128	3
Bad deallocation:	0	0
Out of bounds array access:	9	3
Uninitialized variable:	132	1
Total:	312	8

Note that there are no bad deallocations. However, since the applications are generally fairly mature and all are written in C rather than C++, this is not particularly surprising. Bad deallocations that occur in C are generally beginner's mistakes (much more so than the other defect classes), and tend to happen on the path the code is intended to take, so there is a large likelihood they get caught quickly.
In light of the relative maturity of the commercial applications and the open source model of the Linux code, perhaps one should be surprised that any defects remain at all. Given the amount of testing that all the code bases had undergone before the inspections, this also confirms that testing is not enough: inspection finds defects that escape testing.
Of course, the five commercial applications together contain much more source code than the one open source application. Therefore it makes much more sense to look at defect densities.

Feedback from the developers on Linux inspection
We submitted the details to people on the kernel networking list and have received the following feedback so far:

- The memory leak is a real defect. Independently of this inspection, it has been fixed in version 2.4.20;
- The out of bounds array accesses are not real defects, because the kernel would not work if they were;
- The uninitialized variable is not a defect. This is code implementing a tiny interpreter, and the uninitialized variable represents variables in the interpreted language. These variables have random values when the interpretation starts, and it is the responsibility of the interpreted program to initialize the variables before they are used;
- We have not received definitive feedback on any of the null pointer dereferences.

In summary: one defect is real, 4 defects are not real, and 3 are undecided.

Defect repair comparison
Since those most familiar with the application are best able to determine the need to repair any individual defect, the most reliable metric is which defects need to be fixed according to the developers or maintainers of the code.
The table below reflects the reported defects, the repaired defects, and the defect density (defects/KLSC) for the commercial projects and the open source project. Since we have not yet received feedback on many of the defects reported to the Linux kernel maintainers, the real number for the open source code may be higher.

	Com-mer-cial	Open Source
Reported:	312	8
Repaired:	235	1
Size (KLOC):	568	81.9
Reported / Size:	0.55	0.10
Repaired / Size:	0.41	0.013

On average, both the reported and the repaired defect densities are higher for the commercial implementations compared to the open source implementation.

Conclusions
This study compares five commercial implementations of TCP/IP, the fundamental protocols underlying the Internet, with the TCP/IP implementation in version 2.4.19 of the Linux kernel, an open source general-purpose operating system.
The open source implementation of TCP/IP in the Linux kernel exhibits significantly lower defect density when compared to the five commercial applications and falls within the "Best Third" of source code projects inspected by us.

About Reasoning Inc
Our company is a leading provider of automated software inspection services that helps development organizations reduce the time and cost involved in finding software defects. The company's business is focused on organizations that develop C and C++ applications. Reasoning is headquartered in Mountain View, CA, USA.
The full Linux inspection report can be downloaded from URL
http://www.reasoning.com/dow

nloads/inspectionreport.html.
The full comparison paper is available at http://www.reasoning.com/downloads/opensource.html. For further discussion about these results and /or the our service, please contact us.

Testing Humor

Just in case you ever got the two mixed up. This should make things a bit more clear

IN PRISON...you spend the majority of your time in an 8X10 cell.
AT WORK...you spend the majority of your time in a 6X8 cubicle.

IN PRISON...you get three meals a day.
AT WORK...you only get a break for one meal and you pay for it.

IN PRISON...you get time off for good behavior.
AT WORK...you get more work for good behavior.

IN PRISON...the guard locks and unlocks all the doors for you.
AT WORK...you must carry around a security card and open all
the doors for yourself.

IN PRISON...you can watch TV and play games.
AT WORK...you get fired for watching TV and playing games.

IN PRISON...you get your own toilet.
AT WORK...you have to share with some idiot who pees on the seat.

IN PRISON...they allow your family and friends to visit.
AT WORK...you can't even speak to your family.

IN PRISON...all expenses are paid by the taxpayers with no work required.
AT WORK...you get to pay all the expenses to go to work and then they
deduct taxes from your salary to pay for prisoners.

IN PRISON...you spend most of your life inside bars wanting to get out.
AT WORK...you spend most of your time wanting to get out and go inside
bars.

IN PRISON...you must deal with sadistic wardens.
AT WORK...they are called managers.

Now, get back to work! Go Testing!

Testing Vocabulary

Source www.testingstuff.com

Here is a list of types of testing that we'd like *not* to see:

- AGGRESSION TESTING: If this doesn't work, I'm gonna kill somebody.
- COMPRESSION TESTING: []
- CONFESSION TESTING: Okay, okay, I did program that bug.
- CONGRESSIONAL TESTING: Are you now, or have you ever been a bug?
- DEPRESSION TESTING: If this doesn't work, I'm gonna kill myself.
- EGRESSION TESTING: Uh-oh, a bug... I'm outta here.
- DIGRESSION TESTING: Well, it should work, but let me tell you about my truck...
- EXPRESSION TESTING: #@%^&*!!!, a bug!
- OBSESSION TESTING: I'll find this bug if it is the last thing that I do.
- OPPRESSION TESTING: You will test this, now!
- POISSION TESTING: Alors! Regardez le poission!
- REPRESSION TESTING: It's not a bug, it's a feature.
- SUCCESSION TESTING: The system is dead. Long live the new system!
- SUGGESTION TESTING: Well, it works but wouldn't it be better if...
- PRESIDENTIAL TESTING: Using the definition of testing as defined in the affidavit...

Evenementen

Algemene LedenVergadering

PLAATS NIEUWEGEIN
GEBOUW NBC
DATUM 20 JUNI 2003
TIJD 18:00 - 22:00
Belangrijk:
Aanmelden uiterlijk 23 juni
E-mail: evenementen@testnet.org

AsiaSTAR

PLAATS SYDNEY
GEBOUW
DATUM 14-16 JULY 2003
TIJD -
Informatie:
URL:
http://www.testingconferences.com

SOFTWARE TEST AUTOMATION CONFERENCE & EXPO FALL 2003

PLAATS BOSTON
GEBOUW SHERATON BOSTON HOTEL
DATUM 19 – 22 AUGUST 2003
TIJD -
Informatie: Still the only conference focused exclusively on software test automation issues
URL:
http://www.sqe.com/testautomation/

The International Conference on Practical Software Testing Techniques PSTT 2003 North

PLAATS MINNEAPOLIS, MN
GEBOUW RADISSON HOTEL & CONFERENCE CENTER
DATUM 8-12 SEPTEMBER 2003
TIJD 8:30 - 16:30.
Informatie:
URL: http://www.psqtconference.com

Conquest 2003

PLAATS NÜRNBURG
GEBOUW
DATUM 17-19 SEPTEMBER 2003
TIJD -
Informatie:
URL: http://www.asqf.de

ICSPI 2003 The 2nd International Conference on Software Process Improvement

PLAATS WASHINGTON DC
GEBOUW
DATUM 17-21 NOVEMBER 2003
TIJD -
Informatie: Whether you are adopting a formal assessment model or interested in more informal process improvement approaches, ICSPI 2003 promises to keep the same practical theme in all of its presentations and tutorials. The program will feature presentations by world leading authorities in software process improvement.
URL: www.icspi.com

EuroSTAR 2003

PLAATS AMSTERDAM
GEBOUW RAI
DATUM 1-5 DECEMBER 2003
TIJD -
Informatie: This conference is aimed at those responsible for ensuring software quality in medical technology (incl. IVD) and pharmaceutical companies as well as at hospital technical centres.
URL: www.csvhc.de

Colofon

BESTUUR

Hans van Loenhoud	Voorzitter & 2e penningmeester
Han Toan Lim	Vice-voorzitter
Astrid Freericks	Penningmeester
Marco Jansen van Doorn	Secretaris & Ledenadministratie
Frank van Elsdingen	Algemene Zaken
Bob van de Burgt	Marktverkenning Informatievoorziening & Beheer
Elise Greveraars	Evenementen & Thema-avonden

MARKTVERKENNING, INFORMATIEVOORZIENING EN BEHEER
Bob van de Burgt (T)

TESTNET WEB
Rob Hendriks
Gerrit de Munck
Meile Posthuma (T)

TESTNET NIEUWS
Meile Posthuma (T) (Redactie)
Milo van der Kruis (Redactie)
Rob Hendriks
E-mail: tnn@testnet.org

EVENEMENTEN & THEMA-AVONDEN
Elise Greveraars (T)

TESTNET THEMA
Mark Paap (T)
Egbert Egberts
E-mail: cie-ce@testnet.org (algemeen)
E-mail: evenementen@testnet.org (aanmelden)

TESTNET EVENEMENT
Egbert Egberts (T)
Mark Paap
Fred Weber

LID WORDEN
U kunt lid worden door een e-mail te sturen naar de ledenadministratie of door op onze Internet site het online registratieformulier in te vullen.
Internet site: www.testnet.org

LEDENADMINISTRATIE
Marco Jansen van Doorn
E-mail: ledenadministratie@testnet.org

TESTNET NIEUWS©
TestNet Nieuws verschijnt eenmaal per kwartaal. Kopij aanleveren per e-mail aan de redactie
Het is niet toegestaan om de nieuwsbrief of delen eruit zonder bronvermelding over te nemen.

Legenda: (T) = Trekker aandachtsgebied