Software zonder fouten

De ene professor voorspelt met welke problemen Vista te maken krijgt, de ander gebruikt formele wiskundige methodes om fouten in software juist te voorkomen. Beiden hebben echter een gemeenschappelijk doel: software zonder fouten.

Door Mark van Sommeren en Manfred Flohr

De weddenschap loopt. Ergens in een virtuele brandkast liggen gegevens die het management van Microsoft slapeloze nachten zouden moeten bezorgen. Hierin staan concrete cijfers voor het aantal verwachte problemen in Windows Vista, de opvolger van Windows XP. Andreas Zeller, de geestelijk vader van deze voorspelling, is zelfs zo overtuigd van de nauwkeurigheid. Hiervan dat hij hierover een weddenschap met Microsoft heeft afgesloten. Maar hoewel de weddenschap vooral een sportief karakter heeft, gaat het voor Microsoft uiteindelijk om miljarden dollars. Voor professor Zeller staat zijn eer als een van de beste debuggers ter wereld op het spel.

De informaticus, die software engineering doceert aan de universiteit van Saarbrücken, zegt dat zijn specialisatie "niet datgene is, wat hij zich bij een rijk sociaal leven had voorgesteld." Maar de professor heeft alle reden tot lachen, want waar een doorsnee programmeur vaak toevallig tegen een bug in de software aanloopt, gebruikt hij een zelfontwikkeld programma dat automatisch naar fouten zoekt en zelfs bij ingewikkelde problemen nooit het overzicht verliest. "Vergelijk het maar met een auto, waarvan de eigenaar in de garage klaagt dat die rammelt als hij er 80 kilometer per uur mee rijdt," licht hij zijn tactiek toe. Zijn programma doet hetzelfde als de monteurs in de garage. De verschillende onderdelen worden een voor een getest om de bron van het probleem te vinden. Naarmate er meer gecontroleerd wordt, kunnen er ook meer oorzaken uitgesloten worden. Eigenlijk is het zoeken naar een programmafout of 'bug' makkelijker dan het zoeken naar de oorzaak van de rammelende auto. "We starten de test gewoon telkens weer opnieuw om het programma steeds grondiger te doorzoeken." Modules worden één voor één uitgeschakeld, net zo lang tot het gedeelte met de fout gevonden is. De 'Delta Debugger' van Zeller trekt de strop rond de gezochte bug dus automatisch steeds strakker aan.

Vindt fouten

Professor Andreas Zeller heeft een programma ontwikkeld dat volautomatisch de code in computerprogramma's herkent die fouten veroorzaakt.

Zelf vindt hij het veel interessanter om met dit programma nog onbekende bugs in nieuwe programma's op te sporen dan om al bekende bugs eruit te halen. Zeller kan trouwens nog veel meer, zoals het voorspellen van fouten wat hij nu voor Windows Vista doet. Uit zijn onderzoeken en bevindingen kan hij in ieder geval al concluderen dat áls er al veel fouten in een programma gevonden worden, er gegarandeerd nog veel meer in zitten. "Het is eigenlijk net zoals bij vissen", zegt hij. "Daar waar ze bijten, vang je er altijd meer."

Wanneer Zeller in water vist waarvan nog niemand weet hoe diep het is, kijkt hij altijd eerst naar oudere programmaversies. Op die manier krijgt hij alvast een idee van het aantal te verwachten bugs en kan hij kritieke modules identificeren. Hierbij gaat hij ook na welke componenten van het oude programma nog in het nieuwe programma gebruikt worden en dus nog voor problemen kunnen zorgen.

Zeller heeft overigens nog een aardige vuistregel: "Hoe meer ervaring een programmeur heeft, hoe meer fouten hij maakt." Daar is ook een simpele verklaring voor. Softwarebedrijven zetten immers hun beste programmeurs op de moeilijkste onderdelen van een programma. Om hun sporen door de programmacode te kunnen volgen, bevat Zellers tool een onderdeel dat lijkt op de module die webwinkels zoals Amazon gebruiken om aanbiedingen te doen. De Delta Debugger weet bijvoorbeeld dat programmeurs die een bepaalde functie hebben veranderd meestal ook een andere functie hebben aangepast.

Weddenschap met Microsoft

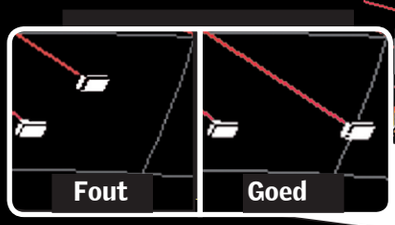
De mogelijkheid om met dit soort als-dan-constructies verbanden en waarschijnlijkheden aan te tonen en daarmee weer voorspellingen te doen over foutgevoelige locaties in code, is ook aan softwaremakers niet ongemerkt voorbijgegaan. Per slot van rekening kosten programmeerfouten jaarlijks miljarden dollars. Dan hebben we het nog niets eens over de miljoenen geïrriteerde consumenten die dagelijks met software werken die het vaak op kritieke momenten laat afweten.

Microsoft benaderde Zeller dan ook met het verzoek om hun bugdatabase in Redmond eens systematisch te doorzoeken. Hiermee was hij de eerste wetenschapper met toegang tot de ontelbare foutmeldingen van Windows-gebruikers – dus tot alle berichten die gebruikers naar Microsoft versturen wanneer een programma is gecrasht en ze uit frustratie op de knop "Foutbericht versturen" drukken om hun probleem te melden.

Het werk van Zeller maakte grote indruk op de softwarereus. Vooral het vooruitzicht dat producten al van tevoren grondig met een 'softwarekeuring' onderzocht kunnen worden, liet hen niet onverschillig. Daarom werd hij meteen op een groot prestigeobject gezet: Windows Vista. De onderzoeker nam de uitdaging aan en ging een weddenschap aan over het aantal fouten dat in het besturingssysteem zou zitten. Als basis voor zijn voorspellingen gebruikte hij zijn ervaringen met Windows XP, de foutmeldingen uit de bug-database van Microsoft en de bèta's van Vista, waarmee Zeller toen al enkele maanden werkte. Volgens hem gaat het net zoals bij het weerbericht. "Je gebruikt de gegevens van eergisteren om te controleren of je het weer van gisteren goed had. Vervolgens gebruik je de nieuwste gegevens om een prognose te maken." Mocht hij zijn weddenschap met Microsoft winnen, dan kon er voor de informaticus nog wel eens een leuke adviserende rol bij Microsoft uit komen rollen.

Waar is de bug?

Dit plaatje brengt de complexiteit van moderne software in beeld. Professor Zeller heeft deze toestand geëxtraheerd uit een draaiend programma: één van de 40.000 verbindingen (oranje) tussen twee variablen veroorzaakt een crash. Maar welke?



Het testprogramma voert veel automatische runs uit waarin het foutieve programma met probleemloze versies vergeleken wordt. Het aantal mogelijke oorzaken wordt steeds beperkter, totdat het programma een afwijkende verbinding (hier herkenbaar aan de kortere lijn) als foutbron gelokaliseerd heeft.

Wiskunde is goedkoper dan handmatig zoeken

Hoewel het zoeken naar fouten een relatief jong vakgebied is, is Zeller zeker niet de enige die zich daar professioneel mee bezighoudt. In Nederland gebeurt dit bijvoorbeeld door professor Frits Vaandrager, hoogleraar Informatica voor Technische Toepassingen aan de Radboud Universiteit Nijmegen. Maar hij pakt het probleem op een totaal andere manier aan dan Zeller. Waar die een statistische benadering heeft gekozen om het aantal gemaakte fouten te voorspellen, probeert Vaandrager juist op wiskundige wijze te bewijzen dat programma's correct zijn en dus helemaal geen fouten bevatten. Het is zijn ambitie om fouten al in een zo vroeg stadium te vinden. Zijn vakgroep probeert bugs te voorkomen door al van tevoren te beredeneren waar je fouten in kritieke onderdelen kunt verwachten. Hiervoor maken ze eerst een model, waarin ze precies beschrijven wat de software doet. Aan de hand van dit model rekenen ze vervolgens alle mogelijkheden door: "Als we bepaalde invoer geven gaat het goed, als we andere invoer geven gaat het fout."

Een computerprogramma is een complex systeem bestaande uit code en modules. Welke parameters je invoert of de volgorde waarin je iets invoert kunnen al bepalend zijn of een programma crasht of niet. Ofschoon het aantal mogelijke toestanden daarvan eindig is, zijn er wel astronomisch veel mogelijkheden.

Wanneer je een programma met honderd variabelen hebt die allemaal een waarde van 0 tot en met 9 kunnen hebben, is het aantal mogelijke toestanden waarin dat programma zich kan bevinden in principe 10^{100}. Dat is een getal dat googol wordt genoemd en dat groter is dan alles wat astronomen tegenkomen. Dan hebben we het alleen nog over een eenvoudig programma. Met elke extra variabele neemt het aantal toestanden explosief toe.

Door een systeem in deelsystemen op te breken en die vervolgens te onderzoeken, kun je wiskundig bewijzen dat ze onder bepaalde randvoorwaarden feilloos werken. Daarbij gebruikt de vakgroep slimme algoritmen om de toestandsruimte van computerprogramma's te doorzoeken.

Vaandrager legt dit uit aan de hand van een voorbeeld: de eenvoudige wiskundige formule $(x+y)(x-y) = x^2-y^2$ geldt voor elke x en y die je invult. Dus zowel voor eenvoudige parameters als $x = 4$ en $y = 2$ die $(4+2)(4-2) = 4^2 - 2^2 = 12$ opleveren, als voor grotere parameters zoals $x = 1024$ en $y = 2049$ die $(1024+2049)(1024-2049) = 1024^2 - 2049^2$ opleveren. Deze 'symbolische' som vervangt dus alle 'concrete' sommen. De taak van Vaandrager en zijn vakgroep bestaat uit het bedenken van nieuwe algoritmes waarmee ze de moeilijkste deelsystemen op fouten kunnen controleren. Dat doen ze niet met een losse rekensom per keer, want daar heb je er oneindig veel van, maar op een hoger niveau en dan met meerdere sommen tegelijk. "En dát is nu de formele wiskundige analyse", zegt Vaandrager.

Voorkomt fouten

Professor Frits Vaandrager analyseert programma's met formeel wiskundige methodes en verifieert zo of er fouten in zitten.

In de formele toestandsanalyse praten ze ook niet over individuele toestanden, maar over toestanden van een bepaald type. "Wanneer je in een toestand van dit type zit, kom je bijna altijd in een toestand van dat type en zo ga je door. Je probeert dan te bewijzen, door gewoon stug door te redeneren, waardoor een slechte toestand veroorzaakt wordt." De computers die de voorbeelden doorrekenen zijn daar vaak uren mee bezig.

Geen fouten?

Wanneer er een bug in een besturingssysteem zoals Vista zit, kun je hooguit wat data kwijtraken of is je systeem minder veilig. In de praktijk is het belangrijker en ook interessanter om bugs te voorkomen in software voor auto's, vliegtuigen of raketten. Daar is het letterlijk van levensbelang.

Op de vraag of het mogelijk is om software te ontwikkelen die honderd procent foutloos is, moet Vaandrager even nadenken. "In theorie is het mogelijk om 100% foutloze software te schrijven, maar dat zou ontzettend veel tijd kosten en erg kostbaar zijn, dus dat doe je niet." Tegen de tijd dat alle berekeningen klaar zijn, is het product niet meer op de markt. Bovendien is de software afhankelijk van de hardware waarop het programma draait.

De vakgroep heeft ook niet de ambitie om van een programma te kunnen bewijzen dat het volledig werkt. Wel kleine onderdelen daarvan, die in een groter geheel zijn ingebed en waarin allerlei fouten kunnen ontstaan. Het liefst proberen ze fouten al in een heel vroeg stadium te detecteren, terwijl mensen nog nadenken over wat globaal de opzet moet zijn. Dan maken ze eerst de benodigde modellen en gaan ze aan het rekenen. Het rekenen aan die modellen gaat steeds sneller, omdat enerzijds de onderzoekers steeds slimmer en beter leren rekenen en anderzijds de computers steeds krachtiger worden. Voor bepaalde verificatietaken worden parallelle computers of grid-technologie gebruikt.

> »Het vinden van een fout is veel makkelijker dan bewijzen dat een programma goed is.«
>
> **Frits Vaandrager**

De leden van de vakgroep bedenken de algoritmes en daarna bouwen ze tools en maken ze software, waarmee ze die algoritmes loslaten op allerlei voorbeelden zoals auto's, vliegtuigen, treinen et cetera. Het doel daarbij is om fouten te vinden. In het model simuleren ze alle toestanden, dus eigenlijke alle waardes die de parameters kunnen innemen. Op het moment dat er een fout wordt gevonden, wordt eerst geanalyseerd waar de fout precies door veroorzaakt wordt. Het kan immers ook zo zijn dat het algoritme niet goed is.

Als voorbeeld noemt Vaandrager het door Apple ontwikkelde Zeroconf-protocol. Puur door dat te modelleren, dus door te beschrijven wat er precies gebeurt, wist zijn vakgroep daar vijf fouten in te vinden. Dat zijn er behoorlijk wat voor een internetstandaard die uitvoerig is beschreven. Zijn studenten en promovendi hebben ook naar Bluetooth gekeken, daar een model voor gemaakt en dat doorgerekend. Bluetooth zit echter redelijk doortimmerd in elkaar, er werden dan ook geen fouten in gevonden. In Firewire daarentegen, wat tegenwoordig op iedere computer is geïnstalleerd, hebben ze wel weer een echte fout gevonden. Die kwam er op neer in dat als iemand een kabeltje in een Firewire-netwerk even lostrok en dan weer vastmaakte, het hele systeem hierdoor volledig van slag kon raken. En dat terwijl Firewire juist ontworpen was als een plug-and-play netwerk waarin je op ieder moment apparaten los en vast kunt koppelen.

De gevonden fouten worden gecommuniceerd naar de beheerders van de desbetreffende protocollen. In het geval van Firewire werd het protocol aangepast. Vaandrager verwacht bovendien dat de fouten in Zeroconf er in een nieuwe versie uitgehaald zullen zijn.

Menselijk falen

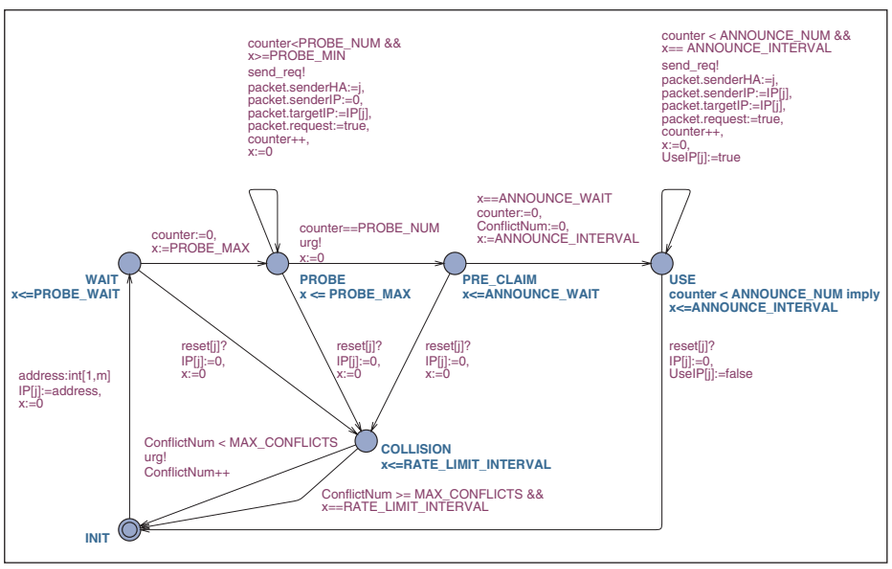
Computers maken geen fouten, mensen die software programmeren maken fouten. Het is bijvoorbeeld niet juist om het neerstorten van de Ariane 5-raket in 1996 toe te schrijven aan een computerfout. De programmeurs hadden software van de Ariane 4-raket hergebruikt voor de nieuwe Ariane, maar de parameters daarvan waren anders. Die kon namelijk schuiner vliegen. Het is natuurlijk wel een computergerelateerde fout en dat is een subtiel verschil, maar uiteindelijk gaat het toch om system engineering. De programmeurs hadden systematischer moeten bijhouden wat de aannames zijn waaronder een bepaald stukje software werkt.

Wanneer je tijdens het gehele ontwikkelproces in het achterhoofd houdt dat de mens feilbaar is en het proces daarop afstemt, maak je de kans op fouten steeds kleiner. Door het ontwerp door meerdere mensen te laten nakijken, modellen te maken en te analyseren kunnen wij als 'onbetrouwbare' mensen toch een betrouwbaar systeem bouwen.

LINKS

www.st.uni-sb.de/zeller: leerstoel voor softwaretechniek (prof. Andres Zeller), Universiteit van Saarland

www.cs.ru.nl/~fvaan: Homepage van Prof. Frits Vaandrager, met daarop verwijzingen naar zijn onderzoeken.



MODEL Dit is een stukje van het model dat de vakgroep van Prof. Frits Vaandrager voor het Zeroconf-protocol gebruikte.