De Ingebouwde Informatica

Dr. F.W. Vaandrager

Rede uitgesproken bij de aanvaarding
van het ambt van hoogleraar in de
Informatica voor Technische Toepassingen
aan de Faculteit der Wiskunde en Informatica
van de Katholieke Universiteit Nijmegen
op vrijdag 6 december 1996

Wat een verschrikking!—dacht ik bij mezelf: zouden de mensen die automobielen werkelijk zo fraai vinden als de tweespansrijtuigen van vroeger dat waren?
Marcel Proust

Mijnheer de Rector Magnificus,
Zeer Gewaardeerde Toehoorders,

Veel van mijn voorouders waren werkzaam als landbouwer in plaatsen als Barendrecht en Charlois. Zo passeerde iemand die omstreeks 1920 de Charloisse Lagendijk van de Dordtsestraatweg af insloeg drie boerderijen en een ruime woning zonder schuren, alle bewoond door Vaandragers [1]. In die tijd werd het graan nog gemaaid met de sikkel en het hooi met een zeis. Alle vervoer ging per wagen. De brik bracht de Vaandragers, die zeer kerks waren, op zondag naar de hervormde kerk van Charlois, per verenwagen ging men melken en de "kromme dissel", de boerenwagen met zware paarden voor vervoer van hooi en graan in de oogsttijd, was het meest imponerende vervoermiddel.

Het is onvoorstelbaar hoe snel en hoe ingrijpend het leven sindsdien veranderd is. De Charloisse boerderijen lagen onder de rook van het zich steeds uitbreidende Rotterdam-Zuid. Binnen enkele jaren werden de eigenerfde boerengronden onteigend en de Vaandragers werden pachters van Rotterdam.

Veel tekenen van de op handen zijnde grote veranderingen moeten toen al zichtbaar zijn geweest. Toch zou ik de gezichten van mijn voorouders wel eens hebben willen zien wanneer iemand ze verteld had over de frequente reizen van hun achterkleinkinderen in fly-by-wire vliegtuigen, hoge snelheidstreinen en met airbags toegeruste personenwagens, over hoe Kurt Cobains gedigitaliseerde stemgeluid zal klinken uit hun stereotorens, over het internet waarover hun nakroost zal surfen en emails versturen, over de chipkaarten waarmee ze intelligente magnetrons zullen afrekenen, en — last-but-not-least — over één van hun achterkleinkinderen die notabene in dienst van een katholieke universiteit betrokken zal zijn bij de verdere ontwikkeling van al deze nieuwlichterij. De paarden zijn op hol zijn geslagen en de brik van mijn voorouders raast met onverantwoorde vaart over de informatiesnelweg in cyberspace. Kan dit wel goed aflopen?

Wat buitenstaanders zich mogelijk onvoldoende realiseren is dat ook de informatici zelf al deze ontwikkelingen nooit hebben voorzien en er nog steeds dagelijks door worden verrast. Zo waren omstreeks 1945 de bouwers van de allereerste computers zoals de MARK I, de ENIAC en de Nederlandse ARRA, weliswaar overtuigd van het grote belang van deze machines voor bijvoorbeeld het berekenen van kogelbanen, maar toch meenden zij dat er voor ieder land slechts één of hooguit een paar van deze apparaten nodig zouden zijn. Geen van deze pioniers heeft kunnen vermoeden dat het aantal microprocessors dat één persoon tegenwoordig op één dag tegenkomt ergens omstreeks de 50 ligt. Bill Gates, misschien wel de invloedrijkste computergoeroe van deze tijd, geloofde ruim een jaar terug nog niet in het belang van internet. Toch maken honderdduizenden Nederlanders daar momenteel dagelijks gebruik van en als de huidige hype doorzet dan is in het jaar 2000 toegang tot internet even normaal als het bezit van een telefoon vandaag. Binnen de informatica maakt iedereen regelmatig grote inschattingsfouten en moeten we allemaal het roer soms helemaal omgooien. Het op de juiste manier omgaan met de snelle veranderingen is een van de belangrijkste uitdagingen waar onderzoekers binnen de informatica zich mee geconfronteerd zien. Juist omdat iedereen tegen ons zegt dat wij werken aan "het vak van de toekomst" belanden wij mogelijk nog sneller in een situatie waarin de onderzoeksvragen waaraan wij werken niet meer relevant zijn en de vakken die wij doceren verouderd.

Denk je dat dit interessant zou zijn? D.w.z. wetenschappelijk en (zucht ...) commercieel?
Vraag van een collega

Het Rozenberg dilemma

Toen ik tijdens mijn studie in Leiden college liep bij professor Rozenberg, loofde die regelmatig een chocoladereep uit voor de eerste student die er in zou slagen om een gegeven probleem x binnen y seconden op te lossen. Hierbij waren x en y door hem altijd zo op elkaar afgestemd dat voorzover ik weet niemand ooit een beloning in de wacht heeft gesleept. Als onderzoeker in de informatica weet men zich regelmatig geplaatst in een soortgelijke situatie, die ik daarom zal aanduiden als het *Rozenberg dilemma*: indien men een maatschappelijk/industrieel relevant probleem probeert op te lossen, dan bestaat er een goede kans dat dit probleem niet meer actueel is op het moment dat men een oplossing heeft bedacht.

Een eerste uitweg uit het Rozenberg dilemma is om uitsluitend onderzoek te verrichten met een dermate theoretisch karakter dat de praktijk, hoe ingrijpend die zich ook wijzigt, daar nooit vat op krijgt. Toepassingen liggen hierbij niet in het verschiet maar, zo betoogt men, hoe vaak is het niet gebeurd dat theoretisch onderzoek waarvan men eerst dacht dat het nergens goed voor was uiteindelijk toch hoogst relevant bleek? Nadat er een tiental jaren in Nederland maar ook elders in Europa op grote schaal is geïnvesteerd in theoretisch informatica-onderzoek, is het momenteel echter buitengewoon moeilijk om hiervoor financiering te vinden. Geldschieters zijn niet meer tevreden met abstracte verhalen over het belang van fundamenteel onderzoek maar eisen concrete industriële en maatschappelijke relevantie op korte termijn. En terwijl er vroeger waarschijnlijk teveel geld is besteed aan theoretisch onderzoek gebeurt dat nu, vrees ik, te weinig.

Toch weten sommigen zich ook binnen het nieuwe regime goed te handhaven met een strategie die men zou kunnen aanduiden als "ijsschots springen". Dit komt er op neer dat men vrolijk hetzelfde theoretische onderzoek blijft doen maar dat wel steeds in termen van de meest recente "buzz words" omschrijft. Zo kan men jarenlang blijven werken aan zeg hemi-semi-demi groepen maar die in onderzoeksvoorstellen achtereenvolgens aanduiden als parallelle machine groepen, internet groepen, reverse engineering groepen, embedded systems groepen, jaar 2000 groepen, enz. Het kenmerkende van deze aanpak zit hem erin dat voordat collega's en financiers doorkrijgen dat de resultaten van het onderzoek niet echt toepasbaar zijn, men alweer is overgesprongen naar een nieuwe terminologische ijsschots.

Veel Nederlandse IT bedrijven lossen het dilemma van Rozenberg op door niet of relatief heel weinig aan onderzoek te doen. Immers, tegen de tijd dat men zelf een oplossing voor een bepaald probleem heeft bedacht, is er in het buitenland al een betere oplossing te koop voor minder geld. Derhalve beperkt binnen de R&D afdelingen van veel grote IT bedrijven in Nederland de "research" zich (de uitzonderingen daargelaten!) tot het goed in de gaten houden van internationale ontwikkelingen en vindt er vooral "development" plaats.

Tijdens een discussie over derde geldstroom financiering van het onderzoek op het Amsterdamse CWI merkte mijn promotor Jan Bergstra ooit op dat wij natuurlijk ook gewoon ijs kunnen gaan verkopen op de Kruislaan. In het huidige tijdsgewricht, waarin de omvang van de binnengebrachte financiële middelen steeds vaker wordt gezien als een maat voor wetenschappelijke kwaliteit en de IT industrie staat te springen om menskracht voor het klaren van automatiseringsklussen, is het voor een academische onderzoeker soms wel erg verleidelijk om wetenschappelijke diepgang in te ruilen voor financieel gewin.

In de hierboven beschreven strategieën wordt het Rozenberg dilemma eigenlijk niet zozeer opgelost als wel ontweken. Ik denk dat de kloof die momenteel in Nederland bestaat tussen enerzijds het veelal zeer theoretische onderzoek aan universiteiten en anderzijds de vrijwel uitsluitend zeer praktische vragen van de IT industrie, voor een deel verklaard kan worden vanuit deze ontwijkende strategieën.[1] Uiteindelijk is het Rozenberg dilemma niet zo onoplosbaar als het op het eerste gezicht lijkt. In tegenstelling tot de Leidse studenten kan men als universitair onderzoeker kiezen uit een vrijwel onbegrensde verzameling van problemen. Zolang men daarvoor maar de moeite wil doen, vindt men altijd wel relevante problemen x waarvoor men redelijkerwijs kan verwachten dat ze oplosbaar zullen zijn binnen de gegeven tijd y.

Tropisch oerwoud

De enorme snelheid van de ontwikkelingen heeft ook veel positieve gevolgen voor het onderzoek op het gebied van de informatica. Aangezien er voortdurend nieuwe onderzoeksvragen boven komen drijven, liggen er relatief vaak belangrijke en fundamentele resultaten min of meer voor het oprapen. Bij informatica-onderzoek voelt men zich soms als een ontdekkingsreiziger die als eerste een onbekend stuk tropisch oerwoud ontdekt vol adembenemende flora en fauna. Dit is wezenlijk anders bij bijvoorbeeld de getaltheorie waarvan we weten dat ze al sinds eeuwen door andere reizigers doorkruist wordt, of onderdelen van de geneeskunde en biotechnologie waar de ontdekkingsreizigers soms met jumbojets tegelijk aangevoerd worden.

Ik heb reeds gesproken over de geweldige invloed van de resultaten van de informatica op ons dagelijks leven. De theoretische informatica heeft daarnaast ook veel resultaten opgeleverd die ons fundamenteel nieuwe inzichten verschaffen over het wezen van berekening. Voorbeelden hiervan zijn de cryptografische systemen zoals RSA, de noties van een interactief en probabilistisch controleerbaar bewijs, de onmogelijkheidsresultaten uit de theorie van gedistribueerde algoritmen, en de theorie van programmaverificatie.

Een grappig punt waarop de informatica zich onderscheidt van veel andere vakgebieden is dat, wanneer iemand een slim programma of efficiënte algoritme bedenkt, dit 'selfish meme' [2] dikwijls tot in lengte van dagen teruggevonden zal kunnen worden op vele honderden miljoenen plaatsen (computergeheugens)

[1] Het kan daarom geen toeval zijn dat juist Grzegorz Rozenberg een van de intitiatiefnemers is van de door de onderzoekschool IPA georganiseerde cyclus "Feedbacks between Academia and Industry" waarin gepoogd wordt deze kloof te dichten.

over de gehele wereld. Zo worden ideeën van informatici met de regelmaat van de klok verspreid op een schaal die vroeger voorbehouden was aan de allergrootste wetenschappers en kunstenaars.

Ook Nederlandse onderzoekers hebben belangrijke bijdragen geleverd aan de ontwikkeling van de informatica. Dijkstra's memes zijn vermoedelijk op zo ongeveer iedere computer ter wereld terug te vinden. Wanneer ik mij beperk tot de terreinen waar ik zelf iets van af weet, zoals semantiek, programmaverificatie en gedistribueerde algoritmen, ga ik niet te ver wanneer ik stel dat Nederland goed meespeelt in de champions league.

Er is inderdaad een wereld van verschil tussen het leven van een informaticus nu en dat van een Charloisse landbouwer anno 1920. Sommigen zullen, net als Marcel Proust, het leven uit die voorbije dagen romantiseren en hun afschuw uitspreken over de hectiek en oppervlakkigheid van het tegenwoordige bestaan. Mij bevalt het "cosmopolitische" leven van informaticus echter uitstekend en met af en toe een fikse wandeling in de natuur — waartoe de omgeving van Nijmegen zoals bekend talloze mogelijkheden biedt — valt het goed vol te houden.

Viermaal het roer om

Verstandig omgaan met het snelle veranderingstempo binnen de informatica vergt dat men extra goed nadenkt bij het uitkiezen van onderzoeksvragen en altijd bereid is om indien nodig het roer om te gooien. Ik wil nu iets vertellen over mijn eigen onderzoek en daarbij vier onderwerpen aanstippen waar ik de afgelopen jaren heel anders over ben gaan denken. Om te beginnen schets ik de context van mijn onderzoek.

Ingebouwde systemen

De eerste digitale computers communiceerden nauwelijks met hun omgeving. Ze gedroegen zich als autistische kinderen, volledig in beslag genomen door hun eigen gedachten en niet reagerend op stimuli uit de omgeving. Wanneer een programma eenmaal was ingevoerd rekende de computer net zo lang totdat dit voltooid was, en alleen door hardhandig ingrijpen van de programmeur kon de berekening onderbroken worden. Al snel realiseerden de bouwers van deze computers zich echter de mogelijkheden die zouden ontstaan voor het besturen van fysische processen indien men de rekencapaciteiten van deze apparaten zou combineren met het vermogen actief te communiceren met hun omgeving. Het ontwerp van de volgende generatie computers werd aangepast om dit mogelijk te maken en zo werd al in 1953 de WHIRLWIND computer van MIT gebruikt voor real-time procesbesturing in een systeem voor het volgen van vliegtuigen.

Door de spectaculaire daling in prijs en grootte van computerhardware, heeft het gebruik van computertechnologie voor dit soort toepassingen een ongekende vlucht genomen. Men spreekt tegenwoordig vaak over *ingebouwde systemen* (van het Engelse *embedded systems*) omdat computertechnologie is ingebouwd (of ingebed) in producten waarvan het de functionaliteit geheel of gedeeltelijk bepaalt; de computer en het product zijn onlosmakelijk met elkaar verbonden. De klasse van ingebouwde systemen is zeer breed en strekt zich uit van de eenvoudige microregelaars die men aantreft in moderne scheerapparaten tot de complexe multiprocessor gedistribueerde computersystemen aan boord van vliegtuigen. Zowel de omvang als de complexiteit van ingebouwde systemen is de laatste jaren dramatisch gegroeid. Zo zit er momenteel bijvoorbeeld omstreeks 2 Mbyte aan programmatuur in de duurdere televiesietoestellen en 0.5 Mbyte aan programmatuur in een doorsnee mobiele telefoon. Men schat dat de informatieverwerkende onderdelen van toekomstige motorvoertuigen verantwoordelijk zullen zijn voor 20% (!) van de totale kosten.

Kenmerkend voor ingebouwde computersystemen is het feit dat ze voortdurend signalen ontvangen uit de buitenwereld en (vaak binnen zeer korte tijd) op deze signalen reageren. Dit impliceert dat bij het ontwerp van dergelijke systemen ook het gedrag van hun omgeving mee in beschouwing genomen dient te worden. Beschouw als voorbeeld de situatie waarin een computer een chemisch proces bestuurt. Regelmatig ontvangt het besturingsprogramma gegevens van de sensoren, zoals de temperatuur en druk op diverse plaatsen. Op basis van deze gegevens kan het programma beslissen om een verwarmingselement aan te zetten, een pomp uit te schakelen, enz. Zodra er een gevaarlijke situatie dreigt te ontstaan, en bijvoorbeeld de druk in een tank te hoog wordt, dient het programma actie te ondernemen om het gevaar af te wenden. Indien men eigenschappen wil bewijzen als "De tank zal nooit exploderen", dan zal het hiertoe gebruikte model zowel informatie moeten bevatten over het besturingsprogramma als over de wijze waarop de chemische reacties zich voltrekken, de invloed van deze reacties op de temperatuur en druk van de tank, enz.

Hybride systemen

Traditioneel houden informatici er een discrete kijk op de wereld op na, in die zin dat zowel de computer als zijn omgeving worden gemodelleerd als systemen met discrete (losstaande) gebeurtenissen. Deze aanpak was gerechtvaardigd en heeft bewezen zeer succesvol te zijn in gebieden als bedrijfs- en informatiesystemen. De laatste jaren zien we echter meer en meer toepassingen, in het bijzonder op het gebied van de ingebouwde systemen, waar het modelleren van de omgeving als een systeem met discrete gebeurtenissen de werkelijkheid wel erg vervormt, onnatuurlijk is en kan leiden tot onbetrouwbare conclusies [6]. Voor dit soort toepassingen zijn de op continue wiskunde gebaseerde modellen die zijn ontwikkeld door wiskundigen en systeemtheoretici veelal compacter, nauwkeuriger en vormen een betere basis voor berekeningen dan de discrete benaderingen van de informatici. Omdat zoals gezegd het ontwerpen van en redeneren over ingebouwde computersystemen alleen mogelijk is wanneer men over een model beschikt van zowel het gedrag van de computer als dat van zijn omgeving, levert dit volop praktische motivatie voor het ontwikkelen van een omvattende theorie van *hybride systemen*, d.w.z. systemen die bestaan uit een niet-triviaal samenstel van discrete en continue componenten. Een dergelijke theorie moet een synthese leveren van concepten, wiskundige theorieën en gereedschappen, die op dit moment verspreid zijn over een aantal disciplines: in eerste instantie Informatica en Systeem- en Regeltheorie, maar bijvoorbeeld ook Elektrotechniek, Werktuigbouwkunde en Natuurkunde.

Ook al worden hybride applicaties gebouwd sinds het begin van de jaren vijftig, toch krijgt de wiskundige studie van hybride systemen pas sinds enkele jaren de aandacht die het verdient. Tot ver in de jaren zeventig ging de interesse van theoretische informatici vooral uit naar het ontwikkelen van formele methoden voor de analyse van programmatuur voor "autistische" batch computers. De tachtiger jaren gaven een revolutie te zien in ons inzicht op het gebied van de *reactieve systemen*, systemen die in interactie staan met hun omgeving. Maar men ging er hierbij nog wel steeds van uit dat zowel het reactieve systeem als zijn omgeving gemodelleerd kunnen worden in termen van discrete gebeurtenissen. Toen men de notie van een reactief systeem eenmaal goed begreep realiseerde men zich echter vrij snel dat het niet zo moeilijk zou zijn om ook kwantitatieve tijdsaspecten aan deze modellen toe te voegen, en een aantal van de belangrijke basisprincipes waren reeds in 1991 geformuleerd. Rond die tijd vonden de eerste twee belangrijke workshops plaats op het gebied van hybride systemen in Ithaca (VS) en Lyngby (Denemarken), bijeenkomsten die aanleiding hebben gegeven tot een explosie van onderzoek op dit nieuwe gebied. Het is fascinerend om te zien hoe onderzoekers vanuit heel verschillende disciplines (Theoretische

Informatica, Systeemtheorie, Robotica, Elektrotechniek, Chemische Technologie, Vliegtuigbouw, Werktuigbouwkunde, enz.) veelal onafhankelijk van elkaar zijn uitgekomen op het model van *hybride automaten*, en dat grote delen van de door informatici ontwikkelde theorie van reactieve systemen gegeneraliseerd kan worden naar deze hybride automaten. Maar ook al hebben we hiermee een aantal technieken in handen om correctheid van hybride systemen aan te tonen, toch valt er nog ongelooflijk veel werk te doen: zodra de verificatieproblemen het niveau van speelgoedvoorbeelden overstijgen lopen alle tools nog vast en verder is er nog nauwelijks theorie die houvast biedt bij het ontwerpen van hybride regelaars. Ik meen dat het heel belangrijk is om ons bij het verder ontwikkelen van de theorie der hybride systemen te laten leiden door voorbeelden uit de praktijk. Zelf ben ik momenteel betrokken bij een voorstel voor een Europees project waarin gekeken zal worden naar de besturing van robots die gras maaien, stofzuigen en hun weg kunnen vinden op golfterreinen.

Heel lang hebben onderzoekers op het gebied van de theoretische informatica (waaronder ikzelf) gemeend dat het modelleren van kwantitatieve en continue aspecten van de omgeving om te beginnen nergens voor nodig is en bovendien aanleiding zal geven tot een hopeloos complexe theorie. Pas sinds kort realiseren we ons dat het aantal toepassingen van hybride formalismen vrijwel onbeperkt is en dat het expliciet modelleren van kwantitatieve en continue aspecten de analyse dikwijls juist eenvoudiger maakt.

Cybertaal

In het laatste jaarboek van de Vereniging van Informatica-ingenieurs Eindhoven, legt Wim Hendriksen, manager software development bij ASM Lithography, uit waarom er bij zijn bedrijf slechts een paar informatici werkzaam zijn [4]:

> *De reden is dat informatici gewend zijn om te leven in hun eigen virtual reality. Het opzetten van een fraaie architectuur is aanmerkelijk eenvoudiger als je je systeemgrenzen zelf kunt bepalen. Wij hebben te maken met mechanische, elektronische en optische beperkingen, naast onze eigen beperkingen. We moeten communiceren met mensen uit andere vakgebieden. Dus onze neiging naar nog formelere specificaties en nog abstractere beschrijvingswijzen wordt simpelweg afgedaan als wereldvreemd gedrag. En terecht. Informatici moeten leren communiceren in de taal van hun klant, en niet in hun eigen cybertaal.*

ASM Lithography ontwikkelt, produceert en verkoopt geavanceerde lithografische apparatuur voor de halfgeleiderindustrie en is een van de meest succesvolle hightech bedrijven in Nederland. Het hybride karakter van deze apparatuur is evident en de teams die hieraan werken zijn dan ook in hoge mate multidisciplinair. Informatici zijn goed in het schrijven van programma's maar hebben klaarblijkelijk een communicatieprobleem dat zo groot is dat ASM Lithography ervoor kiest om het schrijven van software over te laten aan mensen die waarschijnlijk niet zo goed kunnen programmeren, maar in ieder geval wel begrijpen *wat* er geprogrammeerd moet worden.

Ik vrees dat het door Hendriksen gesignaleerde probleem niet alleen speelt bij ASM Lithography, maar bij veel andere IT bedrijven. Misschien zou de opleiding informatica daarom, meer dan op dit moment gebeurt, aandacht moeten besteden aan de communicatieve vaardigheden van studenten. Ook zou men studenten in de technisch gerichte informatica nog nadrukkelijker moeten wijzen op het belang van vakken als analyse, optica, mechanica en elektromagnetisme. Net zoals het voor een diplomaat belangrijk is om goed Duits, Engels en Frans te spreken, zo zou iemand die technisch gerichte informatica heeft gestudeerd in staat moeten zijn om bijvoorbeeld mechanische en elektronische tekeningen te lezen.

Ik heb altijd sympathie gevoeld voor "nerds", voor die contactgestoorde types met hun onverzorgde uiterlijk die tot diep in de nacht achter de computer zitten en elkaar schaterlachend emails sturen over bits en bytes waar de rest van de wereld niets van begrijpt, laat staan er de humor van kan inzien. De briljante ideeën van nerds zijn essentieel voor de vooruitgang van de informatica, maar toch zal het iedereen duidelijk zijn dat bedrijven als ASM Lithography meer behoefte hebben aan een ander soort medewerkers. Een rampzalig misverstand dat bij veel buitenstaanders leeft ten aanzien van de studie informatica is dat deze vooral bedoeld zou zijn voor nerds. Wanneer wij erin slagen om dit misverstand de wereld uit te helpen en wanneer wij aankomende studenten duidelijk weten te maken hoe belangrijk communicatieve en sociale vaardigheden zijn voor het werk van een informaticus, dan denk ik dat de studentenaantallen wel eens zullen kunnen gaan stijgen, en dat in het bijzonder meer vrouwen informatica als studierichting zullen kiezen.

Communicatieve aspecten zijn ook lang ondergewaardeerd bij de ontwikkeling van formele methoden voor de specificatie en analyse van software. Een nieuwe theorie voor hybride systemen is volstrekt kansloos wanneer ze werktuigbouwkundigen, elektrotechnici, enz., dwingt om een geheel nieuwe taal te gebruiken. Veeleer zal zo'n theorie er naar moeten streven om, als een soort wiskundige lijm, bestaande talen en notaties zoveel mogelijk te integreren. Ook de talen die gebruikt zullen worden voor het specificeren van de software zelf zullen leesbaar moeten zijn voor mensen uit andere disciplines, zodat ook specificaties van software bediscussieerd kunnen worden door de leden van multidisciplinaire ontwerpteams. Veel formalismen voor reactieve systemen, zoals de procesalgebra waaraan ik zelf veel onderzoek heb gedaan, zijn geoptimaliseerd om wiskundige analyse mogelijk te maken: de bondigheid van expressies is belangrijker dan hun leesbaarheid. Bij andere formalismen, zoals Statecharts, SDL en gekleurde Petrinetten, is leesbaarheid wel een belangrijk ontwerpcriterium geweest, en het is duidelijk dat de goed doordachte grafische notaties uit deze formalismen een belangrijke verklaring vormt voor hun succes.

Tools are not the most important thing about formal methods — they are the only *important thing.*
John Rushby

Computerondersteuning

Een ander aspect dat lang onderschat is bij de ontwikkeling van formele methoden voor de specificatie en analyse van computersystemen is het belang van computerondersteuning hierbij. Terwijl binnen andere disciplines het nut van computerondersteuning al snel werd onderkend, zijn notabene de informatici zelf wel heel lang doorgegaan met het gebruik van pen en papier voor het leveren van correctheidsbewijzen. Ter verdediging van mij en mijn vakgenoten kan wel aangevoerd worden dat het bouwen van computertools in deze context beslist geen sinecure is.

Ruwweg kan men onderscheid maken tussen twee benaderingen bij het gebruik van tools. Bij de eerste benadering, dikwijls aangeduid met de term *model checking*, laat men de computer met brute kracht alle mogelijke toestanden doorlopen om op die manier te bepalen of alle toestanden voldoen aan de gewenste correctheidseigenschappen. Het probleem bij deze aanpak is dat bij serieuze toepassingen het aantal toestanden vrijwel altijd een astronomische omvang bereikt. Bij de tweede benadering, aangeduid met de termen *proof checking* en *theorem proving*, laat men de computer een door mensen geleverd bewijs checken en de nog ontbrekende details invullen. Deze aanpak heeft het bezwaar dat men de computer vaak in een dermate groot detail moet uitleggen wat nu precies de gemaakte redeneerstappen zijn, dat men er als gebruiker soms moedeloos van wordt en ten einde raad toch maar weer naar pen en papier grijpt.

De afgelopen vijftien jaar is er door een grote groep onderzoekers heel hard gewerkt om de problemen van beide benaderingen te ondervangen. Deze inspanningen beginnen nu hun vruchten af te werpen. Binnen de model checking aanpak heeft men een hele serie technieken bedacht (BDDs, partiële ordening technieken, uitbuiten van symmetrie, region graphs, hashing- en compressietechnieken, enz.) waardoor men niet langer alle toestanden één voor één hoeft af te lopen maar grote verzamelingen van toestanden tegelijkertijd kan afhandelen. Bij proof checking en theorem proving is grote vooruitgang gerealiseerd door het implementeren van efficiënte beslissingsalgoritmen voor bijvoorbeeld rekenkunde, propositielogica en unificatie. De geboekte resultaten zijn dermate indrukwekkend dat *computer-aided verification* de laatste paar jaar een hottopic is geworden binnen de Amerikaanse hardware-industrie. Alleen door intensief gebruik te maken van dit soort technieken denkt men een kans te hebben om in de toekomst dure fouten (zoals recentelijk in de Pentium processor) te voorkomen.

Wanneer informatici eenmaal hebben besloten dat het wenselijk is om hun favoriete formalisme van computerondersteuning te voorzien, dan besluiten zij vervolgens in veel gevallen om alle hiervoor benodigde software zelf te schrijven. Informatici zijn eigenwijs en denken altijd dat ze betere programma's kunnen schrijven dan hun collega's. Toch is er de laatste jaren een aantal tools op de markt gekomen (voorbeelden zijn PVS, SPIN, Cospan en SMV) waaraan door zoveel slimme mensen is bijgedragen en die als gevolg daarvan dermate snel en effectief zijn, dat ikzelf tot de conclusie ben gekomen dat het verstandig is om zoveel mogelijk gebruik te maken van deze tools en er eventueel alleen wat interfaces en extra intelligentie aan toe te voegen. Ik zie het als een teken van rijpheid van een vakgebied wanneer het wiel niet steeds opnieuw wordt uitgevonden, maar er bepaalde gereedschappen en technieken zijn die iedereen gebruikt.

The fundamental principle of science, the definition almost, is this: the sole test of the validity of any idea is experiment.
Richard Feynman

Experimentele evaluatie

In een recent artikel doen Lukowicz, Heinz, Prechelt en Tichy [5] verslag van een onderzoek dat de hypothese lijkt te bevestigen dat informatici verhoudingsgewijs weinig artikelen publiceren met experimenteel onderbouwde resultaten:

> *Of the papers in the random sample that would require experimental validation, 40% have none at all. In journals related to software engineering this is over 50%. In comparison, the fraction of papers lacking quantitative evaluation in Optical Engineering and Neural Computation is only 15% and 12%, respectively.*
>
> *Conversely, the fraction of papers that devote one fifth or more of their space to experimental validation is almost 70% for Optical Engineering and Neural Computation, while it is a mere 30% for the CS random sample and 20% for software engineering.*

De auteurs suggereren dat het verhoudingsgewijs kleine aantal gevalideerde resultaten een serieuze zwakte vormt van het informatica-onderzoek, een zwakte die gecorrigeerd zou moeten worden wil het vakgebied op termijn gezond blijven.

Claims ten aanzien van de industriële relevantie van formele methoden voor specificatie en verificatie zijn moeilijk experimenteel te onderbouwen:

- Iedere methode zal goed werken voor sommige problemen en minder voor andere.
- Het succes van een methode hangt op een cruciale manier af van de persoon die haar gebruikt.
- Het doen van de benodigde hoeveelheid case studies vergt enorme investeringen in termen van geld en menskracht.

Wetenschappers die hun claims experimenteel willen onderbouwen zien zich derhalve geplaatst voor een enorme opgave.[2] Het gecombineerde effect van het Rozenberg dilemma en het bekende prisoners dilemma (wetenschappers die hun claims niet onderbouwen krijgen hun werk wel gepubliceerd) verklaart waarom experimenteel onderzoek binnen de informatica tot nu toe zo impopulair is.

Ik zou op deze plaats krachtig willen pleiten voor een opwaardering van experimenteel onderzoek binnen de informatica. Experimenteel onderzoek mag

[2] Overigens is deze opgave niet a priori onoplosbaar. Onderzoekers binnen de sociale wetenschappen zien zich geplaatst voor een soortgelijke problematiek en hebben toch onderzoeksmethodieken ontwikkeld waarmee dit soort claims experimenteel onderbouwd kunnen worden.

dan vaak niet zo spectaculair ogen, maar het is wel belangrijk. Het ontbreken van experimentele onderbouwing dient daarom in negatieve zin mee te tellen bij de beoordeling van artikelen. Verstandig onderzoeksbeleid binnen de informatica zou er niet direct op gericht moeten zijn om onderzoekers van theoretisch naar toepassingsgericht te drijven, maar simpelweg vaker van wetenschappers moeten eisen dat zij hun claims experimenteel onderbouwen.

Dankwoord

Het CWI, de UvA, het MIT en de Ecole des Mines wil ik bedanken voor de aangename en vruchtbare tijd die ik op ieder van deze instituten heb doorgebracht.

Van alle collega's met wie ik gedurende de jaren met zoveel plezier heb samengewerkt wil ik volstaan er twee te noemen. Jan Bergstra wil ik bedanken voor zijn rol als promotor. Van hem heb ik geleerd *hoe* je onderzoek moet doen. Nancy Lynch wil ik bedanken voor een zeer inspirerend jaar in Boston en de daaropvolgende samenwerking. Tijdens de vele discussies met haar is me duidelijk geworden *wat* voor onderzoek ik de komende tijd wil doen.

Mijn dank gaat uit naar het Stichtingsbestuur, naar het College van Bestuur en naar de Faculteit der Wiskunde en Informatica van de Katholieke Universiteit Nijmegen, voor mijn benoeming als hoogleraar.

Mijn opa is ongetwijfeld mijn eerste leermeester geweest op het gebied van de techniek. Dit maakt het voor mij extra bijzonder dat ik de toga die ik vandaag draag van hem cadeau heb gekregen. Mijn ouders wil ik bedanken voor de manier waarop ze me altijd gestimuleerd hebben, ook toen ik een voor hen volstrekt vreemde richting opging. Lieve Nicolaas, zou het toeval zijn dat papa staat te oreren op jouw naamdag? Of zullen we het zien als een bedankje voor de vrolijke wijze waarop jij dagelijks tegen mij oreert vanuit je buikdrager, box en tummy-tub? Lieve Emérence, veel collega's danken bij een gelegenheid als deze hun partner voor een grote tolerantie voor overwerk. Laat ik het omgekeerde doen en je ervoor bedanken dat je me telkens weer wijst op het bestaan van mooie en belangrijke zaken die niets met informatica van doen hebben.

Mijn nieuwe collega's en de studenten wil ik tot slot enkele woorden voorhouden uit een recente voordracht van Amir Pnueli, de winnaar van de laatste Turing Award. Wanneer wij het gebruik van onze nieuwe, formele methoden ingang willen doen laten vinden in de industrie dan zullen wij ons vooralsnog zeer bescheiden moeten opstellen. Verder zullen we hard moeten werken aan het vergroten van de toepasbaarheid van deze methoden, want zoveel tijd is er niet.

Ik heb gezegd.

Referenties

[1] W. den Boer. Vaandragers tussen 1920 en 1930. In *Genealogie van het geslacht Vaandrager*. De Residentie, Den Haag, 1973.

[2] R. Dawkins. *The Selfish Gene*. Oxford University Press, 1976.

[3] D.K. Hammer en F.W. Vaandrager. Ingebouwde systemen nog verre van feilloos. *Automatiserings Gids* 17, april 1996.

[4] W. Hendriksen. Informatici ongeschikt. In *VIE-jaarboek 1996*, pp 92–97, Vereniging voor Informatica-ingenieurs Eindhoven, 1996.

[5] P. Lukowicz, E.A. Heinz, L. Prechelt en W.F. Tichy. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software* 28(1):9–18, januari 1995.

[6] O. Maler, Z. Manna en A. Pnueli. From timed to hybrid systems. In *Proceedings REX Workshop on Real-Time: Theory in Practice*, Mook, Nederland, Juni 1991, LNCS 600, pp 447–484, Springer-Verlag, 1992.

[7] F.W. Vaandrager. Hybrid systems. In *Images of SMC Research 1996*, pp 305–316. Stichting Mathematisch Centrum, Amsterdam, 1996.