Prof. dr. Marko van Eekelen

# Leven Lang Computeren, Leven Lang Foeteren?

## Er valt nog veel te leren!

Open Universiteit
www.ou.nl

Leven Lang Computeren, Leven Lang Foeteren?

Er valt nog veel te leren!

Rede

in verkorte vorm uitgesproken

bij de openbare aanvaarding van het

ambt van **hoogleraar softwaretechnologie**

bij de Open Universiteit Nederland

op vrijdag 5 maart 2010

door *prof.dr. M.C.J.D. van Eekelen*

ISBN: 978-90-5291-108-3



Drukwerk: Datawyse, Maastricht

Foto Omslag: Dick van Aalst, Radboud Universiteit Nijmegen

Ontwerp Omslag: Vivian Rompelberg, Open Universiteit Nederland

## *Opening*

Geachte rector, dames en heren, collega's, vakgenoten, vrienden en familie, voor ik mijn oratie uitspreek, wil ik allereerst u allen danken voor uw aanwezigheid hier in Heerlen. Het is geen kort tripje. Daarom ben ik extra blij dat u er allemaal bent.

Schrikt u niet van de camera's. De Open Universiteit staat voor afstandsonderwijs waarbij het leren onafhankelijk is van tijd en plaats. Daar horen goede online, multimedia voorzieningen bij. De Open Universiteit is, met de TU Delft, er als eerste van Nederland bij om deel te nemen aan iTunes U: een online voorziening voor open onderwijsmateriaal met op dit moment al meer dan 250.000 presentaties waar ook gerenommeerde universiteiten als Stanford, Harvard, MIT en Oxford aan deelnemen. Als u daar behoefte aan heeft kunt u via iTunes U deze rede downloaden en hem op uw PC, iPod of iPhone nog eens bekijken en beluisteren. De website van de OU wijst u ongetwijfeld de weg[1].

## *De Open Universiteit*

Velen van u zijn wellicht niet zo bekend met wat de Open Universiteit doet. Daar zal ik snel wat aan doen.

De Open Universiteit Nederland is opgericht in 1984, alweer bijna 26 jaar geleden. In die periode heeft de Open Universiteit meer dan een kwart miljoen mensen als student mogen verwelkomen. Samen hebben deze studenten bijna anderhalf miljoen modulen gekocht en er zijn bijna 600.000 cursuscertificaten uitgereikt.

De Open Universiteit heeft 7 faculteiten, Informatica, Natuurwetenschappen, Managementwetenschappen, Psychologie, Rechtswetenschappen, Cultuurwetenschappen en Onderwijswetenschappen.

De OU heeft 15 studiecentra in Nederland en er zijn ook 6 studiecentra in Vlaanderen. Op studiecentra kunnen studenten niet alleen elkaar ontmoeten maar ze kunnen er ook terecht voor het inzien van cursusmateriaal, studiebegeleiding, studieadvies en het afleggen van tentamens. Medewerkers hebben hun werkkamer in Heerlen, bij een studiecentrum of bij een andere universiteit.

De OU is de jongste universiteit van ons land. Als enige universiteit in Nederland verzorgen wij open academisch afstandsonderwijs. In studies die de onderwijskwaliteit vergelijken komt de Open Universiteit steevast op de tweede

plaats, na de Universiteit van Wageningen, maar vóór alle andere universiteiten.

Naast het geven van onderwijs heeft de Open Universiteit sinds 2009 ook het doen van onderzoek als wettelijk taak. Er is een onderzoekinstituut, Centre for Learning Sciences and Technologies (CELSTEC), dat naast de faculteiten staat en natuurlijk is er ook veel facultair onderzoek.

Overigens kent u vermoedelijk wel een aantal OU studenten, zij het van een afstand. De OU psychologie studente Marianne Vos heeft vele wielrentitels op haar naam staan en bij de recente Olympische winterspelen hebt u op de schaats de OU studenten Jan Blokhuijsen, Stefan Groothuis, Annita van Doorn, en Laurine van Riessen in actie kunnen zien en in de bobslee de OU studenten Sybren Jansma en Arnold van Calker.

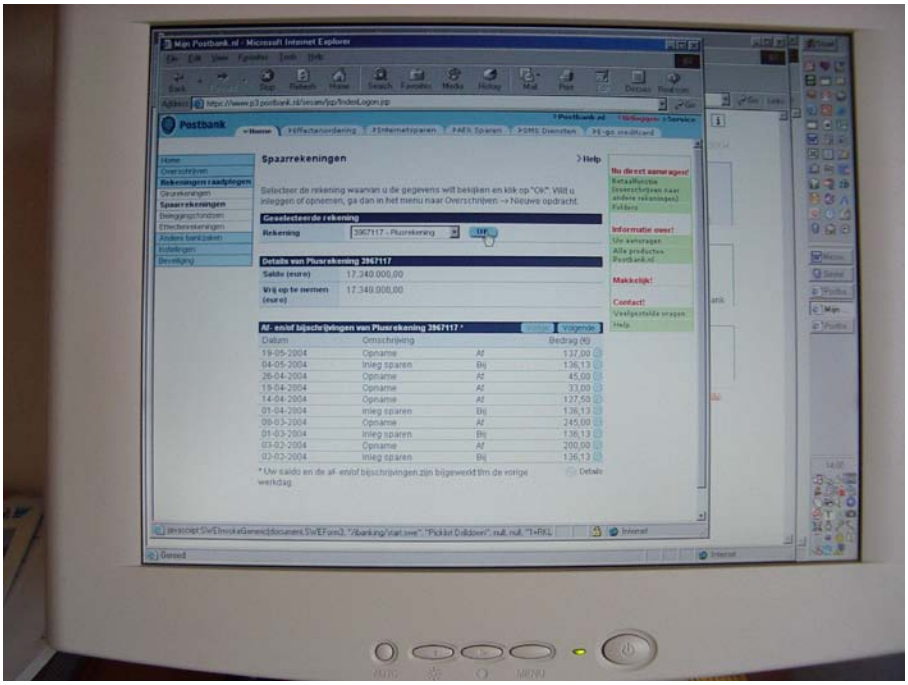
### *Leven Lang Computeren, Leven Lang Foeteren?*

Mijn oratie heeft als titel 'Leven Lang Computeren, Leven Lang Foeteren?' en als ondertitel 'Er valt nog veel te leren'. U zult zich wel afgevraagd hebben wat ik daarmee bedoel. 'Is het nu echt allemaal zo erg?' vraagt u zich misschien af maar ook 'Is Informatica niet meer dan computeren? Is dat nou wetenschap?' en toen u las in de samenvatting, dat ik ga vertellen wat ik wil gaan doen om de situatie te verbeteren dacht u misschien. 'Denkt hij dat hij in zijn eentje de wereld kan veranderen?'. U heeft volkomen gelijk natuurlijk, met uw kanttekeningen. Ik heb het wat scherp aangezet om het duidelijker te maken. Ik hoop dat het na deze lezing nog wat duidelijker zal worden wat ik doe, waarom ik het doe en natuurlijk wat ik in de toekomst wil gaan doen. Al doende hoop ik u ook deelgenoot maken van mijn visie op Informaticaonderwijs en op Informaticaonderzoek. In die visie staat het belang van Software Analyse centraal.

Maar laat ik allereerst toelichten wat ik met 'Leven Lang Computeren, Leven Lang Foeteren?' wil aanduiden. Er wordt inderdaad door heel wat mensen gemopperd op de computer. Nu heeft u geregeld met softwarefouten te maken die niet zo'n probleem vormen. Geregeld komt het bijvoorbeeld bij Office gebruikers voor dat plots een regel op het scherm er maar half of twee keer staat terwijl dat niet zo hoort: even heen en weer scrollen en het probleem is opgelost.

Een andere variant hiervan is de internetrekening waar plotsklaps miljoenen op lijken te staan. In 2004 is mij dit overigens zelf overkomen (zie Figuur 1). Ik heb natuurlijk hard gelachen. Maar iemand met weinig computerervaring (soms 'digibeet' of beter 'adigitaal' genoemd) zou kunnen denken dat dat geld er echt

is en het proberen uit te gaan geven, hetgeen uiteraard alleen maar kan eindigen met een grote schuld en nog grotere frustratie over het gebruik van computers.




*Figuur 1*: Wat zou u doen met 17 miljoen?

Deze frustratie is zo'n dagelijks gewoon gevoel dat het voor cabaretiers interessant is om dit uit te spinnen. Zo gaat Hans Sibbel (artiestennaam Lebbis) in zijn show "Wanneer begint het schieten" tekeer over het ongemerkt wijzigen van de instellingen bij een update van een routeplanner. Als u het gezien heeft, krijgt u wellicht de indruk dat dit vooral gaat over onvrede over de manier waarop het apparaat moet worden gebruikt. Het is vanbuiten uit, zonder naar de code of naar het ontwerp te kijken, moeilijk vast te stellen of het om een ontwerpfout of om een programmeerfout gaat. Hoe dan ook, mag je van een volwassen software product verwachten dat de update software de gebruikersinstellingen overneemt.

Dat een cabaretier ergens veel succes mee heeft, is vaak een teken dat er iets aan de hand is. Dat is in dit geval ook zo. We zijn met zijn allen in groeiende mate afhankelijk geworden van computers. Dit heeft zich vrijwel helemaal in de loop van de laatste 30 jaar afgespeeld. Dit geldt zeker in maatschappelijke zin, (denk bijvoorbeeld aan het betaalverkeer) maar misschien meer nog in de persoonlijke sfeer (denk aan mobieltjes, YouTube, Google, en marktplaats). De

wereld is de afgelopen 30 jaar sterk gedigitaliseerd en dit proces is nog lang niet afgelopen.

Deze groeiende afhankelijkheid van computers maakt het ook problematischer als computers zich niet gedragen zoals je zou verwachten. In de persoonlijke sfeer leidt dit tot groeiende frustratie en tot groeiende agressie, ja zelfs tegen de computer zelf. Onderzoek[2] geeft aan dat ongeveer 2 op de 3 mensen wel eens schelden tegen hun computer, 1 op de 3 slaat wel eens met de muis op tafel of gooit er mee en 1 op de 6 slaat tegen het scherm of schopt tegen de computerkast.

De oorzaken van de frustratie en agressie tegen computers liggen in een groeiende afhankelijkheid van computers in onze maatschappij, niet alleen op reis van 'Tom-Tom's' maar in vrijwel elk facet van het dagelijks leven: televisiezenders, elektriciteitslevering, geld (niet meer dan bits in een database). Ook vrijwel elk apparaat heeft een chip met software erin: de wasmachine, je mobieltje, de televisie, de muziekinstallatie, de magnetron, soms zelfs de koelkast en de oven.

Mijn oven thuis slaat af als ik eerst de temperatuurknop draai en daarna het klokje aanzet. Pas als ik de knop weer eerst uitzet en daarna weer aan, dan doet de oven het weer. Mijn collega aan de Open Universiteit, Evert van de Vrie, had een vergelijkbare ervaring met zijn auto. Het olielampje deed het niet. Toen het in de garage gemaakt was, wilde hij het daar direct controleren maar het werkte nog steeds niet. De klep van de auto stond nog open. Eerst moest de klep dicht dan pas werd de computer gereset (opnieuw gestart). Het is als een Windows systeem wat eerst alles moet afsluiten en dan opnieuw opstarten voor wijzigingen effectief zijn. Dit effect treedt nu dus in het dagelijks leven ook op. Straks wordt het nog zo dat we inderdaad, zoals een bekende grap zegt, problemen oplossen volgens de Windows methode: autoproblemen bijvoorbeeld oplossen door eerst allemaal uit te stappen, de motor uit te doen en dan de motor weer aanzetten om vervolgens allemaal weer in te stappen en te hopen dat het probleem is verholpen. Zo gaat dat met Windows toch ook?

Overigens, heb ik de indruk dat de Windows methode van probleem oplossen ook al in de politiek is doorgedrongen. Alleen is het daar de vraag of dezelfde ministers die uit het kabinet zijn gestapt, er ook weer in zullen kunnen stappen en of het bijdraagt aan het oplossen van problemen, weet ik ook zo net nog niet.

Aangezien er zoveel apparaten zijn met software erin, zijn we eigenlijk allemaal de hele dag aan het computeren. Misschien bent u het zich niet altijd be-

wust maar vandaag de dag geldt voor vrijwel iedereen ‘Een dag niet gecomputerd is een dag niet geleefd’.

Informatica is de wetenschap van de informatieverwerking met behulp van computer, de wetenschap van het computeren dus. Vanuit de wetenschap is een enorme bijdrage geleverd aan de ontwikkeling van computersystemen variërend van programmeertalen en programmeertechnieken tot informatiemodellering. De nadruk hierbij ligt traditioneel meer op ontwikkeling dan op analyse.

Zelf heb ik ook uitgebreid[3] bijgedragen[4] aan theorie[5,6,7], ontwikkelmethoden[8,9,10,11] en tools[12,13] voor *functionele*, i.e. op het wiskundige functiebegrip gebaseerde, programmeertalen[14]. Het ontwikkelen met dergelijke programmeertalen biedt belangrijke voordelen voor het redeneren[15,16,17,18] over programma’s en voor het parallel uitvoeren[19,20,21] van programma’s. Als Chairman van het Steering Committee van de Internationale Symposium reeks ‘Trends in Functional Programming’ draag ik hier niet alleen inhoudelijk maar ook organisatorisch aan bij.

Maar na vele jaren van nadruk op ontwikkeling valt er met een computer nog steeds moeilijk te werken zonder te ‘’foeteren’’. Aan vrijwel elke eis die je aan een normaal product zou stellen, wordt niet voldaan: standaardisatie, gegarandeerde eigenschappen: vergeet het maar! Het is daarom nodig dat de komende jaren veel meer aandacht besteed gaat worden aan analyse van software[22].

### *Waarom software analyse*

Die groeiende afhankelijkheid van computers maakt enerzijds zoals gezegd agressie en frustratie los wanneer computers niet doen wat we van ze verwachten, anderzijds kan het ook tot grote maatschappelijke schade leiden wanneer er iets misgaat met software waar we van afhankelijk zijn. Problemen liggen veelal niet in de hardware maar in de software. Bugs in auto’s (airbags, remmen, gas), software voor maanreizen, fouten in chirurgische apparatuur, falende communicatiesystemen in kritieke situaties, belastingformulieren die kwijt zijn, bedrijven die failliet gaan door softwarefouten: software is alomtegenwoordig!

Onze maatschappij is van software afhankelijk geworden maar ook het dagelijkse persoonlijke leven kan niet meer zonder. Dit gaat meestal goed maar soms niet en in sommige situaties willen we er alles aan doen om dat te voorkomen. Wat denkt u van waterkeringen? De kosten als ze niet dicht gaan zijn natuurlijk enorm, nog vervelender is de schade als ze onverhoopt wel dicht gaan terwijl het niet nodig is. Niemand wil bruggen die opengaan als ze dicht moeten blijven en als kerncentrales door software gestuurd gaan worden dan

zal dat met de aller-allergrootste zorg geregeld moeten worden. Dat geldt ook voor toepassingen waarbij de veiligheid en/of het bedrijfsmatig functioneren op cruciale wijze afhankelijk is geworden van de software. Dit soort (onderdelen van) toepassingen worden 'safety critical' of 'industry critical' genoemd, in goed Nederlands: veiligheidskritisch of bedrijfskritisch. Een softwarefout in een dergelijk kritisch onderdeel kan een bedrijf in een kritieke toestand brengen. De schade kan enorm zijn. Het leven van veel mensen kan ervan afhangen.

U begrijpt dat het nodig is om bij safety critical en industry critical toepassingen extra aandacht aan de correctheid te besteden. Er is een keur van ontwikkel- en testtechnieken die daarbij gebruikt worden. Er zijn standaarden, zoals de DO178B standaard voor de luchtvaart die vereisen dat elke stap in de ontwikkeling bekeken wordt en dat er een veelheid aan testen gedaan wordt. Zo'n standaard stelt eisen aan het proces. Er moet overal een procedure voor zijn, er moet overal naar gekeken zijn. Alles moet afgevinkt zijn. In die zin heeft zo'n standaard veel gemeen met ISO9000 standaarden die u de garantie van een goede procedure geven maar of het product goed is dat garandeert het niet.

Mijn vrouw, Lianne Dirven, heeft aan een dergelijke ISO9000 certificering meegewerkt. Zij was verbaasd over het gebrek aan kwaliteitseisen voor het product. Ze zei 'je kunt een bedrijf gecertificeerd krijgen, dat betonnen zwembandjes maakt als je alle procedures maar volledig hebt beschreven zodat controleerbaar is of je je aan je procedures houdt.' De procedures zijn controleerbaar maar dat geeft geen garantie dat het product dan ook van goede kwaliteit is. Het is duidelijk niet voldoende om alleen maar naar het proces te kijken. Een zorgvuldige analyse van het product is daarnaast noodzakelijk.

Ook voor software producten wordt analyse steeds belangrijker. MITRE[23] en SANS[24] vinden dat ook. MITRE en SANS zijn twee grote internationale organisaties die jaarlijks lijsten produceren van software risico's en veel gemaakte software fouten die aanleiding geven tot cybercrime. Zij adviseren sinds dit jaar de makers van software contractueel vast te leggen op de kwaliteit van hun software. Dit soort initiatieven gaan ongetwijfeld een keer aanslaan en het is evident dat het belang van software analyse dan zal groeien, niet alleen vanwege het voorkomen van fouten maar ook om juridische conflicten te beslechten.

### *Soorten software analyse*

In de software wereld zijn er traditioneel verrassend weinig middelen om software als product te analyseren. Er zijn strikte procedures. Er wordt veel getest maar het eindresultaat analyseren is niet gangbaar in de softwarebedrijven. Maar er wordt wel heel veel onderzoek naar software analyse gedaan in de aca-

demische wereld en de eerste resultaten daarvan worden al mondjesmaat op bescheiden schaal toegepast. Er moet echter nog veel gedaan worden.

In de 80-er jaren van de vorige eeuw werd voor het analyseren van software vrijwel uitsluitend pen en papier gebruikt. Voor een programma van een paar regels kon je op papier een bewijs leveren dat het correct was maar in veel gevallen zaten er in het bewijs meer fouten dan in de software. In de 90-er jaren ontstonden in programmeeromgevingen allerlei tools om de programmeur bij te staan. Voor het analyseren van software betrof dit vooral typeringssystemen die volautomatisch controleren of een routine wel aangeroepen wordt met een object van de juiste soort. Dat vormde een belangrijke vooruitgang en een belangrijke bron van fouten werd hiermee weggenomen. In de 0-der jaren (de kinderjaren) van deze eeuw zijn er aanzienlijk meer hulpmiddelen voor software analyse ontstaan. Dit betreft zowel automatische tools als tools die intensieve interactie vergen; zowel tools gebaseerd op formele methoden en diepe wiskundige theorie (bv. MCRL2, Uppaal, SPIN, PVS, COQ, ESC/Java, Java Pathfinder, BLAST, Spec#, Frama-C) als tools gebaseerd op het verzamelen van informatie en het herkennen van patronen hierin (Coverity, Fortify). Ook tools voor het genereren en automatisch uitvoeren van software tests zijn nu in ruime mate beschikbaar (veel taalspecifieke tools maar ook algemener toepasbare frameworks zoals QuickCheck, GAST en TorX).

Zonder enige twijfel kan gesteld worden dat de technieken voor het analyseren van software de laatste 10 jaar aanzienlijk zijn verbeterd en dat ze de komende 10 jaar, tijdens de 10-er jaren van deze eeuw, nog meer zullen verbeteren. Ik verwacht dan ook dat in de 20-er jaren van deze eeuw de Informatica eindelijk ook zal uitgroeien tot een volwassen ‘engineering science’ met gevestigde op wetenschappelijke kennis gebaseerde technieken voor ontwikkeling en analyse van software producten.

### *LaQuSo*

Om het onderzoek naar het analyseren van software producten te stimuleren en in de praktijk van het bedrijfsleven te laten zien wat de mogelijkheden zijn van de tools van dit moment is het Laboratory for Quality Software opgericht. Dit was in 2004 een initiatief van de Technische Universiteit Eindhoven waar in 2005 de Radboud Universiteit Nijmegen zich bij heeft aangesloten. Vanaf het begin ben ik verantwoordelijk geweest voor de Nijmeegse tak van LaQuSo, sinds 2009 samen met Pieter Koopman.

In de afgelopen jaren hebben we tientallen projecten gedaan waarbij in opdracht van een bedrijf software bestudeerd is zowel op security gebied als op algemene correctheidseigenschappen. We hebben een eigen certificatiemodel (LSPCM)[25] ontwikkeld en in een aantal gevallen hebben we na een assessment studie ook een certificaat kunnen uitreiken.

### *Formele methoden*


Om de toepasbaarheid van formele, wiskundige methoden in de praktijk te vergroten worden bijvoorbeeld, in het door de Europese Unie gesubsidieerde onderzoekproject *CHARTER*[26] tools ontwikkeld om vanuit requirements gebruik makend van transformaties Real-Time Java code te genereren die een hoge mate van betrouwbaarheid heeft. Dit is een samenwerkingsproject met Engelse, Ierse, Zweedse en Duitse partners en vanuit Nederland de Radboud Universiteit Nijmegen, de Technische Universiteit Twente, het Nationaal Lucht en Ruimtevaartlaboratorium en het softwarebedrijf Luminis. In het project zijn bedrijven vertegenwoordigd uit de gezondheidszorg, de bewakings-, de luchtvaart- en de automobielindustrie.

Vanuit Nijmegen coördineer ik de CHARTER verificatieactiviteiten. Die betreffen het verder ontwikkelen van tools gebaseerd op formele methoden waarmee de betrouwbaarheid van de software aangetoond kan worden. Met Olha Shkaravska, Rody Kersten, Wojciech Mostowski en Erik Poll werken we zowel aan tools voor functionele correctheid (komt er het juiste resultaat bij de juiste invoer) als niet-functionele correctheid (blijft het programma binnen de beschikbare hoeveelheid geheugen).

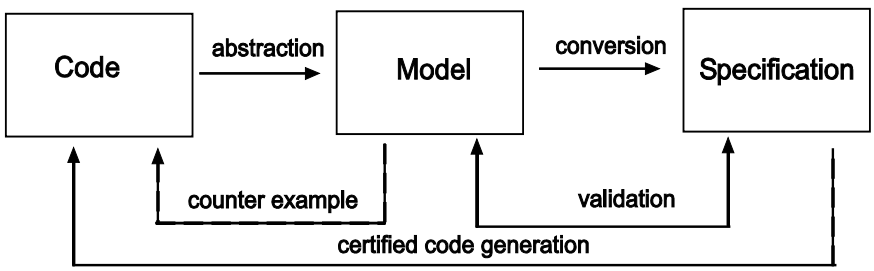
Een alternatieve techniek om betrouwbaardere software te produceren betreft het ontwerpen van een vaste structuur waar verschillende onderdelen in passen. Uitgaand van die structuur kunnen dan eigenschappen worden aangetoond die voor de gehele software gelden, mits de onderdelen aan bepaalde eigenschappen voldoen. Dit valt te vergelijken met een dragende constructie van een gebouw die doorgerekend kan worden en waarvoor dan aangetoond kan worden dat wanneer de onderdelen aan bepaalde voorwaarden voldoen, de constructie als geheel bijvoorbeeld een stevige storm of zelfs een orkaan kan doorstaan. Hiermee ontstaan een soort Software Product Lijnen waarvan je kunt garanderen dat er betrouwbare producten uitkomen. Onderzoek in deze richting wordt in de context van netwerken op chips[27] uitgevoerd door promovendus Freek Verbeek onder begeleiding van Julien Schmaltz van de Open Universiteit Nederland waarbij Frits Vaandrager en ik de beoogde promotoren zijn.

In dit decennium zouden dergelijke tools en dergelijk structuren wel eens tot de standaarduitrusting van de academisch gevormde programmeur kunnen gaan behoren. Hiermee zou het ontwikkelen van software aanzienlijk betrouwbaardere code op kunnen leveren.

Er blijft echter altijd nog heel veel bestaande software waarvan het nodig is de betrouwbaarheid te verbeteren. Daartoe heb ik een onderzoekprogramma gestart waarbij technieken ontwikkeld worden om van een groot programma de kern te isoleren, van die kern een formeel model te maken, dat formeel model te analyseren en indien nodig het te corrigeren om er vervolgens de betrouwbaarheid van aan te tonen waarna er vanuit dat model weer bewezen betrouwbare code gegenereerd kan worden die de originele code kan vervangen.

Deze techniek lijkt wel wat op wat een medisch chirurg doet die het hart van een menselijk lichaam isoleert eruit haalt en vervangt door een kunsthart met de gewenste betrouwbaarheid. Om die reden noem ik dit wel eens het *Software Surgery*[28] onderzoek (zie Figuur 2). Hieraan werk ik samen met Sjaak Smetsers, Leonard Lensink, Luc Rutten en Ken Madlener. Er is echter nog een lange weg te gaan voor alle onderdelen hiervan zodanig zijn ontwikkeld dat de hele keten in de praktijk toepasbaar is.

Niettemin is het nu al mogelijk deelstudies te doen die passen in die grote lijn.




*Figuur 2*: The Software Surgery approach.

### *Formal Methods in Industrial Critical Systems*

Over dergelijke deelstudies voor toepassingen die een centrale rol vervullen in het garanderen van de veiligheid of het functioneren van een bedrijf, zogeheten veiligheidskritieke en bedrijfskritieke toepassingen, heb ik de afgelopen 3 jaar een aantal artikelen gepubliceerd over het gebruik van formele methoden voor analyse van software in de praktijk. Dit betrof zowel administratieve toepassingen (Aia[29]) als industriële toepassingen (Cybernetix[30]) als software onderdelen

die gebruikt worden om andere software mee te maken (Trolltech[31]). Ook hebben we gewerkt aan de correctheid van een voor betrouwbaarheid ontworpen operating systeem (Fiasco[32]) een aan de correctheid van numerieke algoritmen (ModRed[33]). Een studie over de Maeslantkering is recent afgerond en de publicatie ervan is in voorbereiding. Ik zal op die laatste studie wat nader ingaan om u een idee te geven van hoe dat gaat.

### *Een case study: de Maeslantkering*

De Maeslantkering (van bovenaf gezien in Figuur 3) is sinds 1997 de laatste grote schakel van de Deltawerken in de Nieuwe Waterweg bij Rotterdam. Elke arm is ongeveer zo lang als de Eiffeltoren hoog is. Hij is op Discovery Channel te zien in de serie Extreme Engineering als de grootste robot van de wereld.

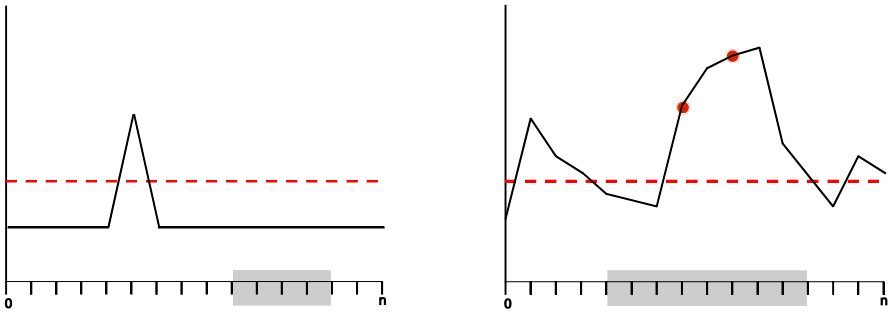
*Figuur 3*: De Maeslantkering.

Hij beschermt circa 1 miljoen bewoners in Zuid-Holland. De kering is gebouwd onder de eis van een maximaal verwachte overstromingskans van eens in de tienduizend jaar wat neerkomt op een faalkans van 1 op 100 voor zowel het sluiten als het openen van de kering.

Rijkswaterstaat en NRG (Nucleair Research en Consultancy groep) vroegen LaQuSo een onderzoek te doen naar hoe formele methoden kunnen bijdragen aan het verlagen van geschatte faalkansen van software onderdelen van de Maeslantkering. Het project werd uitgevoerd door mij, Ken Madlener (die het meeste werk er aan heeft besteed) en Sjaak Smetsers.

Hoe gaat een dergelijke studie te werk? Allereerst is de vraag welk onderdeel van de software bestudeerd zal worden en welke eigenschap bestudeerd zal worden. In dit geval was dat het onderdeel van de software dat bepaalt of de kering dicht gaat of niet. Dit betrof ongeveer 150 regels formele specificatie in Z en 400 regels programmatekst in C: de routine ‘BepaalPeilOverschrijding’. Die routine bepaalt op grond van verwachte waterhoogten of het peil, naar verwachting, overschreden zal gaan worden of niet.

Laten we nu even vooral naar het sluiten kijken. Het doel is sluiten als er binnenkort water hoger dan de norm verwacht wordt. Bij een kleine, korte overschrijding, zoals in het linkerdeel van Figuur 4, moet er niet gesloten worden.




*Figuur 4*: links niet sluiten, rechts wel sluiten.

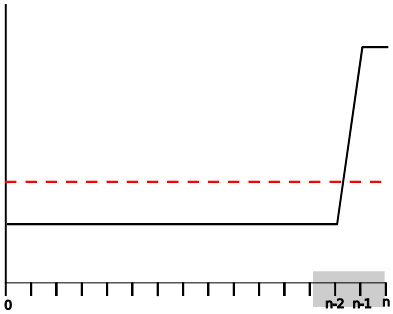
Een voorbeeld van wanneer er wel gesloten moet worden is in Figuur 4 rechts aangegeven. Twee punten, met een verwacht tijdsverschil van totaal 20 minuten (2 eenheden) die allebei boven het peil zitten en waarbij de tweede niet lager zit dan de eerste. Dit noemen we een 'niet-dalende overschrijding'.

In formele taal was dit gespecificeerd als *sluiten* =
$\exists$ i : N {i $\in$ 1.. length_interval -2 $\wedge$ Verwachting (i+2) $\geq$ Verwachting (i) $>$ norm}

Allereerst is de C-code vertaald naar een model in de modeltaal Promela, een formele taal geschikt voor model checking, waarbij door het SPIN tool eigenschappen gecontroleerd kunnen worden voor alle toestanden die het systeem doorloopt. Dat moeten er dan wel eindig veel zijn natuurlijk, anders blijf je letterlijk bezig. Het voordeel hiervan is dat je letterlijk alle mogelijkheden afloopt waardoor je vaak fouten vindt die een mens niet ziet. Ook vergroot het het vertrouwen als je hiermee geen fouten vindt. Er zijn een aantal verschillende model checking tools die deze techniek toegankelijk maken. Model checking wordt dan ook steeds vaker toegepast in de praktijk. De uitvinders van model checken, Clarke, Emerson en Sifakis hebben om die reden in 2007 de Turing award gekregen (een soort Nobelprijs voor de Informatica).

Dit leverde op dat in ons model van de code de laatste 2 waarden van de voorspelling niet worden meegenomen. Ook in de code zelf bleek dit het geval te zijn. Als, zoals in Figuur 5, er aan het eind de verwachtingsperiode een peiloverschrijding verwacht wordt, zal de routine dus (nog) niet tot een sluiting besluiten. Gelukkig bleek in de praktijk de periode ruim genoeg genomen zodat

dit niet tot problemen zal kunnen leiden. Het besluit wordt alleen iets later genomen dan eigenlijk de bedoeling van de specificatie was.




*Figuur 5*: kering sluit (nog) niet.

Vervolgens hebben we het model vertaald naar een bewijs assistent (proof assistant). Daarmee maak je een formeel wiskundig model, kun je eigenschappen van dat model bewijzen en het bewijs door de bewijsassistent volautomatisch laten controleren. Aannemend dat de controle van bewijzen correct geïmplementeerd is, weet je dan 100% zeker dat je bewijs goed is en dat de eigenschap geldt voor je model. Aangezien het model dicht bij de code zit, geeft dit natuurlijk ook meer vertrouwen in de code.

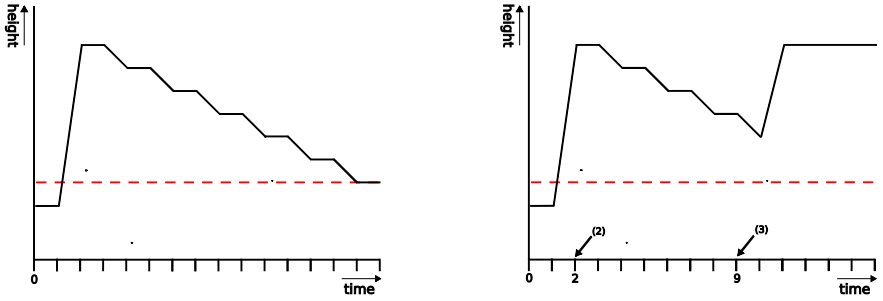
Deze techniek laat zich niet alleen nog moeilijk toepassen (het vergt veel tijd van een onderzoeker van hoog niveau) maar vooral moeilijk schalen (het lukt vaak alleen voor relatief kleine programma's). Er is echter al een complete C-compiler correct bewezen met deze techniek en er wordt door heel veel mensen heel hard aan gewerkt om de techniek te verbeteren. Het is dus zeer beloftevol.

In de toekomst wil ik onderzoek doen naar het automatisch vertalen van de code naar het model waarbij we de vertaling zelf bewijzen zodat gegarandeerd wordt dat de voor het model bewezen eigenschappen ook voor de code gelden. Dat zal niet in alle gevallen mogelijk blijken te zijn (omdat het model bijvoorbeeld te complex wordt). Daarom wil ik ook onderzoek doen naar het analyseren van de voorwaarden waaronder een dergelijke, bewezen vertaling mogelijk is. Uiteraard moet ook die analyse weer correct bewezen worden.

Terug naar de kering: het bewijs is geleverd. Het vertrouwen is vergroot. Het model behorend bij de code voldoet aan de specificatie. Maar is die specificatie wel goed? Dekt de specificatie wel op de juiste manier alle situaties waarvan men vindt dat de kering moet sluiten? Dit heet het valideren van de specificatie.

Om de specificatie (niet-dalende overschrijding) te valideren hebben we alternatieve definities opgesteld van wanneer er van een peiloverschrijding sprake zou kunnen zijn Vervolgens hebben we situaties laten genereren waarbij er op grond van de alternatieve definities verschillende besluiten genomen zouden kunnen worden.

Dit leidde tot de scenario's in Figuur 6 die we aan de experts van Rijkswaterstaat hebben voorgelegd. Bij de eerste sluit de kering niet, bij de tweede sluit de kering te laat.




*Figuur 6*: links: kering sluit niet; rechts: kering sluit te laat.

De conclusie van Rijkswaterstaat was dat dit gevallen waren waarvan de specificatie inderdaad niet goed de intentie van het programma dekte: de kering zou in deze gevallen (eerder) moeten sluiten. Echter, is het ook uiterst onwaarschijnlijk dat een dergelijke hoge golf binnen 10 minuten ontstaat en bovendien zou dan de hoeveelheid water beperkt zijn en geen schade aan kunnen richten.

Gelukkig maar, u hoeft dus niet bang te zijn dat u natte voeten (of erger) krijgt. De LaQuSo studie heeft al met al enerzijds geleid tot een groter vertrouwen in de software van de Maeslantkering en anderzijds heeft het geleid tot de conclusie dat formele methoden een belangrijke rol kunnen spelen in het verkleinen van de faalkans.

Overigens is het bepalen van daadwerkelijke getallen voor faalkansen voor software een bijzonder lastig probleem omdat het in feite gaat om ontwerpfouten (waar weinig statistisch onderzoek naar is gedaan) en niet om slijtagefouten (waar veel statistisch onderzoek naar is gedaan).

### *Size analyse*

Het voorgaande onderzoek betrof het analyseren van functionele eigenschappen met behulp van formele methoden. Ook niet-functionele eigenschappen van

softwareproducten kunnen met formele methoden geanalyseerd worden. Een voorbeeld hiervan is het analyseren van de hoeveelheid geheugen die een programma gebruikt. In het kader van mijn NWO-project AHA (Amortised Heap usage Analysis)[34,35] (met Alejandro Tamalet, Ron van Kesteren en Olha Shkaravska) wordt op dit terrein goede vooruitgang geboekt.

Het bepalen van de grootte van data structuren is essentieel voor geheugen analyse[36]. In het AHA project onderzoeken we polynomiale afhankelijkheden van de grootte van de uitvoer ten opzichte van de invoer. In de praktijk komen dergelijke polynomiale afhankelijkheden het vaakst voor. Logaritmische afhankelijkheden zitten vaak in specifieke, geoptimaliseerde bibliotheekfuncties. Exponentiële afhankelijkheden zijn uiterst onwenselijk.

We hebben niet alleen een methode (een type systeem) gevonden om polynomiale afhankelijkheden te controleren maar ook om ze af te leiden. We vereisen daartoe een natuurlijke beperking die controle in eindige tijd altijd mogelijk maakt. We hebben bewezen dat deze restrictie nodig is omdat zonder die restrictie het controleren van het type equivalent is met het $10^e$ probleem van Hilbert: het oplossen van Diophantische polynomiale vergelijkingen over gehele getallen. Het $10^e$ probleem van Hilbert is een van de grote problemen van de wiskunde, in 1900 door Hilbert opgesteld. Het is in 1970 door Matiyasevich aangetoond equivalent te zijn met het Halting problem. Het Halting probleem behelst het schrijven van een algoritme dat voor alle programma's van een programmeertaal in eindige tijd beslist of het programma stopt of niet. Het is door Alan Turing[37] in 1936 aangetoond dat zo'n terminerend algoritme niet bestaat voor algemene programmeertalen. Het Halting problem is dus 'onbeslisbaar'. Met de equivalentie met het Halting problem is dus aangetoond dat het onmogelijk is een algoritme te schrijven dat in eindige tijd ofwel een oplossing levert ofwel aangeeft dat er geen oplossing is.

Dankzij de resultaten van ons onderzoek kunnen we voor een grote klasse van functies de size afhankelijkheden van de output ten opzichte van de input niet alleen controleren maar ook afleiden.

Een leuk aspect van ons werk is dat we de precieze polynomiale afhankelijkheid kunnen afleiden door middel van een aantal tests[38]. Een lijn (een eerste graads polynoom) wordt bepaald door twee punten. Aannemend dat de polynoom van graad 1 is, zijn dus twee tests voldoende om de polynoom te bepalen. Door nu te beginnen met graad 0, een polynoom te bepalen, die te controleren en als die niet voldoet de graad te verhogen en op dezelfde manier verder te gaan, zullen we uiteindelijk de juiste polynoom vinden. Hiertoe moeten we wel aannemen dat het programma inderdaad een polynomiale afhankelijkheid heeft.

Het is echter niet altijd zo eenvoudig. Polynomen kunnen elke willekeurige hogere graad hebben zodat meer punten nodig zijn. We kunnen meerdere invoer argumenten hebben zodat onze polynomen meerdere variabelen hebben. Het resultaat kan samengesteld zijn zodat we meerdere polynomen tegelijk moeten bepalen. De argumenten zijn soms zelf samengesteld uit meerdere onderdelen zodat niet elke test informatie oplevert over alle onderdelen en er dus meer tests nodig zijn. Maar dat hebben we allemaal volledig opgelost.[39] Ook werken we aan oplossingen voor het probleem dat een algoritme onder bepaalde voorwaarden zich gedraagt volgens de ene polynoom en onder andere voorwaarden volgens een andere[40].

Dit is vrij fundamenteel onderzoek. Het onderzoek is uitgevoerd voor een formele, stricte, functionele programmeertaal. In het vorig jaar gestarte EU CHARTER project proberen we de resultaten om te zetten in praktijkresultaten, voor imperatieve talen in het algemeen en voor Real-Time Java in het bijzonder.

Met ons werk zitten we ergens tussen het heel praktische onderzoek, waarbij men met zeer grote restricties op de programmeertaal geheugencellen van daadwerkelijke programma's erg goed kan schatten[41],[42], en het heel theoretische werk waarbij men beschrijft welke complexiteitsklassen door bepaalde constructies beschreven worden[43]. Vanuit die positie hebben we een nieuwe reeks internationale workshops gestart: FOPARA (Foundational and Practical Aspects of Resource Analysis)[44]. Dit bleek aan te slaan. Niet alleen is de tweede FOPARA ook al gepland maar ook was de workshop een van de succesvolste workshops van de grote Formal Methods Europe conferentie. Naar aanleiding hiervan heeft Professor Ricardo Peña van de Universidad Complutense te Madrid besloten zijn sabbatical bij de Radboud Universiteit door te brengen om gezamenlijk onderzoek te doen op dit gebied. Ook zal een van zijn medewerkers voor een half jaar als gastonderzoeker aan de Radboud Universiteit verbonden zijn.

Dit soort resource analyses (waarbij resources zowel geheugen als tijd kan zijn) zijn met name belangrijk voor embedded systemen. Als je precies weet hoeveel geheugen een programma gebruikt, hoef je minder marge aan te houden en kan je systeem kleiner, en goedkoper zijn. Ook kun je daarmee garanderen dat je programma niet door een Denial of Service aanval vast kan lopen doordat het te weinig geheugen heeft. Dit is zowel vanuit correctheid als vanuit security oogpunt belangrijk. Natuurlijk spelen nog veel meer aspecten een rol bij security onderzoek.

### *Security en Privacy bij Slimme Meters*

In de context van LaQuSo heb ik met Engelbert Hubbers diverse security assessment projecten geleid betreffende slimme energiemeters. Bij dergelijke slimme meters hoeft de meteropnemer niet meer bij u thuis aan te kloppen om de stand op te nemen maar de meter kan de stand zelf doorgeven via een digitale netwerkverbinding met de centrale van de netbeheerder. Ook kan de netbeheerder slimme meters gebruiken voor het aan- en uitschakelen van de energietoevoer. Verder bieden slimme meters mogelijkheden om de energie-infrastructuur te optimaliseren door een beter maar vooral actueler zicht op de energiebehoefte in de diverse onderdelen van het netwerk. Ten slotte biedt de slimme meter de mogelijkheid om de tarifering flexibeler te maken. Dit laatste kan variëren van meerdere verschillende tarieven en tarieftijden tot een prepaid model waarbij tot alleen tot een bepaald bedrag energie afgenomen kan worden. Het is een manusje van alles, die slimme meter.

Maar het is ook een schoolvoorbeeld van technology push waarbij de technologie al beschikbaar is en de behoefte nog gecreëerd moet worden. Aangezien bij het ontwerp van de technologie nauwelijks is nagedacht over security en privacy aspecten[45] is het niet verwonderlijk dat er het een en het ander valt aan te merken wat security en privacy betreft[46]. Voor een consument is het niet altijd een-twee-drie duidelijk dat de persoonlijke energiegegevens goed beschermd zijn. Kan een journalist of een potentiële inbreker zomaar bij uw gegevens en zien of u thuis bent of afleiden wat u aan het doen bent of wellicht zelfs van afstand de energietoevoer uitschakelen? Natuurlijk moeten voor deze problemen goede oplossingen komen en daar wordt ook hard aan gewerkt.

Maar het grootste probleem is eigenlijk dat de sector niet echt gewend is om ICT-oplossingen te verzinnen en te implementeren. De Openbaar Vervoersector had hetzelfde probleem toen ze van de papieren strippenkaarten en de papieren treinkaartjes over wilden gaan op één OV-chip.

Als je nieuw bent in de ICT-wereld, maak je qua security al gauw een van de twee grote, klassieke fouten:

***'Collect before you protect'.***
Alle gegevens van iedereen verzamelen om die dan vervolgens te gaan beveiligen. Klinkt goed maar het is vergelijkbaar met het verzamelen van al het vuurwerk van Nederland op 1 plaats teneinde de veiligheid te verbeteren. Die vuurwerkverzameling vormt zélf een veiligheidsrisico. Als je dan ook nog allerlei mensen met allerlei verschillende bevoegdheden wilt toelaten dan is het risico wel heel erg groot dat het een keer misgaat.

***'Security through obscurity'***
Als we maar niemand vertellen hoe onze beveiliging werkt, dan is het heel veilig. Vergeet het maar. Het werkt in de praktijk juist andersom. Wie erop vertrouwt dat toch niemand weet hoe de beveiliging werkt, maakt de beveiliging juist minder goed. Zodra er iets over bekend wordt (en daar kun je op wachten) wordt de beveiliging al snel gekraakt. Beter is het om het beveiligingsalgoritme openbaar te laten zijn en de sterkte van de beveiliging alleen af te laten hangen van de sterkte van de sleutel (in de IT-wereld vaak het aantal bits wat ervoor gebruikt is). Wie vertrouwt u eerder uw kostbaarheden toe: iemand die even achter een gordijntje verdwijnt en zegt dat alles uiterst veilig is opgeborgen maar hij vertelt u niet hoe of iemand die u de kluis laat zien en vertelt dat je een 12-cijferige code nodig hebt die alleen hijzelf weet?

Op diverse manieren (LaQuSo projecten, ronde tafel overleg, consultatiegroepen) adviseer ik met anderen van de afdeling Digital Security de netbeheerders hoe ze hun slimme meters beter kunnen beveiligen en ook beter kunnen ontwerpen. Hier valt nog veel te doen.

We moeten toe naar een moderne security architectuur gebaseerd op public key cryptografie met een 'trusted computing base' in de meter en voor elke doelstelling van de meter een apart op maat gemaakt security protocol waarbij de minimaal benodigde informatie wordt gecommuniceerd zodat zowel privacy als security aspecten voldoende gewaarborgd kunnen worden.

Mijn STW onderzoekproject Secure Metering (met Professor Bart Jacobs en Flavio Garcia) heeft als doel de problematiek van slimme meters en rekeningrijden te generaliseren door cryptografische protocollen voor secure metering te ontwerpen. Dit project wordt ondersteund door de Rijksdienst voor het Wegverkeer (RDW) en Alliander, de netbeheerder (voorheen NUON).

## *Duurzaamheid*

Er zijn 2 dingen waar ik last krijg van plaatsvervangende schaamte: waar ik me schaam over mijn vakgebied, de informatica.

Het eerste is het als ik een mail of een bestand binnenkrijg waarin plots na elke regel een extra lege regel staat. De oorzaak hiervan is dat de interne code voor de overgang op een nieuwe regel na al die jaren nog steeds niet gestandaardiseerd is (er zijn twee verschillende codes voor: Carriage Return en Line Feed; Commodore en Mac gebruikten vroeger de Carriage Return, Linux en Unix gebruiken Line Feed, Windows gebruikt niet 1 teken maar twee tekens namelijk een Carriage Return gevolgd door een Line Feed). In Unicode (een nieuwe standaardcodering die bedoeld is om universeel toepasbaar te zijn) zijn er maar

liefst 7 verschillende codes voor de overgang naar een nieuwe regel). Standaardisatie heeft dus in dit geval niet tot minder varianten maar juist tot meer varianten geleid! Dit staat in schril contrast tot bijvoorbeeld de transportsector waar de standaardisatie op containers geleid heeft tot een enorme kwaliteits- en productiviteitsverbetering. Denk aan containerboten, containervrachtwagens en containerterminals. Die zijn allemaal mogelijk geworden door standaardisatie. In de ICT is standaardisatie veelal meer lippendienst dan realiteit.

Misschien is het dan ook niet zo verwonderlijk maar wel bijzonder ergerlijk dat er nog steeds software is die niet goed met de verschillende varianten omgaat en daardoor te veel lege regels produceert. Na 50 jaar of meer computertechnologie kunnen we zelfs de overgang op een nieuwe regel nog niet altijd goed krijgen. Triest. Als dat alles is, zie ik u denken, maakt hij zich daar druk om?
U heeft gelijk, het is maar een van de vele voorbeelden waar marktconcurrentie echte standaardisatie met bijbehorende schaalvoordelen in de weg staat. Het is verspilling van energie om al die varianten uit elkaar te houden en om de een in de ander om te zetten als bestanden van de ene naar de andere computer gaan. Het is ook verspilling van energie om je daar druk over te maken als daar weer eens iets niet helemaal goed is gegaan.

Over verspilling van energie gesproken: daar is informatica goed in! De duurzaamheid is ver te zoeken. Daar moeten we ons pas echt voor schamen. Het heeft natuurlijk ook te maken met de snelle ontwikkelingen van het vak. Wat heb je nu bijvoorbeeld nog aan een computer zonder internet? Maar toch,…. (als u mij toestaat het even in het Engels te zeggen) 'IT-technology owes an apology to ecology.' Natuurlijk zijn er ook veel besparingen mogelijk dankzij allerlei IT-ontwikkelingen maar dat is maar een kant van de medaille. De andere kant is dat bijvoorbeeld dat elke 3-4 jaar een nieuwe computer heel gewoon is, niet omdat het apparaat versleten is maar omdat steeds nieuwere software steeds hogere eisen stelt aan de apparatuur.

Op een of andere manier is dit ook in het gedrag, 'de cultuur', doorgedrongen. We drukken bijvoorbeeld iets gemakkelijker opnieuw even af dan dat we het bewaren waardoor de hoeveelheid verbruikt papier alleen maar stijgt en de belofte van het papierloze kantoor ver weg blijft. Hopelijk gaan de e-readers (even groot als een velletje papier, goed contrast, fijn leesbaar, handzaam, niet te duur, digitaal en goed leesbaar) hier verandering in brengen. Ook hebben we spaarlampen om minder Watt te verbruiken. Maar hoeveel Watt de computer verbruikt, weten we vaak niet eens. Thuis doen we het licht netjes uit als we een kamer verlaten terwijl de router aan blijft staan en als het even tegen zit, staat de computer ook de hele dag te snorren.

Ik wil iets aan die schaamte gaan doen door middel van onderzoek om de duurzaamheid te verbeteren. Als lid van de raad van advies van het Innovatieplatform Sustainable ICT zal ik hierover adviseren maar ik wil ook zelf een bijdrage leveren. Duurzaamheid en leren gaan goed samen. Als docent ben je per slot van rekening bezig met het opleiden voor de toekomst. Een bijdrage aan de duurzaamheid van die toekomst ligt natuurlijkerwijs in het verlengde daarvan. Ik ben met Anda Counotte-Potman van de Open Universiteit bezig met plannen voor een duurzaamheidanalyse van kennisinstellingen. Ook maak ik plannen voor het verbeteren van de compilertechnologie zodat niet geoptimaliseerd wordt op tijd of op geheugen maar op energie. (met Bernard van Gastel, Rody Kersten en Sjaak Smetsers van de RU en Trident/NXP). Op het komende ICT-Delta congres zal ik in de Sustainable ICT sessie hier meer over vertellen.

### *Software Usage Analysis*

Software analyse dient zich mijns inziens niet te beperken tot het bestuderen van de innerlijke werking van het product. Ook het gebruik van het product dient geanalyseerd te worden. Hiertoe wil ik aan de Open Universiteit onderzoek opstarten op het gebied van het monitoren van een software applicatie. Met monitoren bedoel ik het gadeslaan van buitenaf of van binnenuit van het gebruik van de software. Vervolgens kan de relevante informatie geanalyseerd worden. De informatie die uit een dergelijke analyse komt is van zichzelf interessant. Maar de analyse kan ook gebruikt worden om enerzijds het gebruik van het softwareproduct te verbeteren (ditzelfde doel had de 'Office Assistant' van Office applicaties. Deze 'paperclip' had een beperkte analyse en was daardoor meer irritant dan behulpzaam. In de 2007 en latere versies komt hij dan ook niet meer voor. Hier is overduidelijk ruimte voor onderzoek naar verbetering.), anderzijds kan de analyse gebruikt worden om de software te verbeteren (wordt het wel zo gebruikt zoals is bedoeld, hoe komt dat, kan product verbeterd worden?).

Dit is in principe algemeen toepasbaar onderzoek. In eerste instantie zal gekeken worden naar het ontwikkelen van een educational monitoring methode waartoe een toolset voor het verzamelen en analyseren van relevante data van educationele systemen ontwikkeld zal worden op analoge wijze als bij de e-learning feedback aanpak van Professor Jeuring.

### *Wikiwijs - Wikiwijzer*

Samen met Professor Paul Kirschner van het Centre for Learning Sciences and Technologies van de Open Universiteit en Harold Vranken, Lloyd Rutledge en Frans Mofers van de Faculteit Informatica van de Open Universiteit werk ik aan het definiëren van een onderzoekproject ('Wikiwijzer') naar het verbeteren

de effectiviteit van grootschalige repositories voor onderwijsmateriaal. Het onderzoek is algemeen maar het zal zich met name richten op Wikiwijs.

Met Wikiwijs[47] wil de Minister van Onderwijs onderwijsinstellingen stimuleren om de mogelijkheden van ICT beter te benutten. Het is een platform voor docenten om open, digitaal leermateriaal te kunnen vinden, gebruiken en te bewerken. De komende jaren moet Wikiwijs zich gaan ontwikkelen tot een plek met leermateriaal voor iedere docent in het hele onderwijsstelsel van basisonderwijs tot Universitair onderwijs. Stichting Kennisnet en de Open Universiteit hebben van het Ministerie van Onderwijs Cultuur en Wetenschap de regierol gekregen in het realiseren en uitbouwen van Wikiwijs.

In het *Wikiwijzer* onderzoekprojectvoorstel komen verschillende onderzoekcomponenten aan bod:

- een trust model, zoals in de context van security gebruikelijk is, teneinde een model van educational trust te ontwikkelen dat gebruikt kan worden om het daadwerkelijk vertrouwen in het onderwijsmateriaal van Wikiwijs op een goede manier weer te geven.
- technieken van het semantic web en van semantic wiki's om een op semantiek gebaseerd onderzoeksfaciliteit te ontwikkelen zodat binnen Wikiwijs 'zoeken' vaker eindigt met 'vinden'.
- de zojuist genoemde educational monitortechnieken om het gebruik van Wikiwijs te analyseren en voorstellen tot verbetering te doen
- en tot slot tools voor het visualiseren van interactie in sociale netwerken om de netwerken binnen Wikiwijs zichtbaar te maken.

Hiermee hopen we de effectiviteit van het Wikiwijs project te optimaliseren zodat iedereen er wijzer van kan worden.

### *Kenniseconomie: allemaal jaarlijks op cursus*

Niet alleen zijn er technieken te ontwikkelen en nieuwe terreinen te ontginnen maar ook zijn er vele ervaringsfeiten die tot de folklore[48] van de informatica behoren die ontkracht kunnen worden of tot harde wetenschappelijke resultaten omgezet kunnen worden. Dankzij de resultaten van huidig en toekomstig onderzoek zal de toekomst ons meer mogelijkheden brengen om software als een product te analyseren hetgeen ons meer vertrouwen zal geven in de software waar we van afhankelijk zijn. Daar is nog veel onderzoek voor nodig.

Om de resultaten van nieuw onderzoek toe te kunnen passen in de praktijk is het natuurlijk nodig dat de praktiserend informaticus zich de resultaten van dat

onderzoek eigen maakt. Dit brengt me bij het motto van de open universiteit 'leven lang leren'.

De Open Universiteit maakt het mogelijk om naast een baan een aantal vakken of een complete studie te volgen. Hiermee vormt de Open Universiteit een belangrijke schakel in de kenniseconomie.

Nu de politieke partijen zich opmaken voor nieuwe verkiezingen, lijkt het me goed dat ze voor hun verkiezingsprogramma overwegen om het begrip kenniseconomie nog meer inhoud te geven door iedereen te verplichten jaarlijks een minimum aantal uren aan bijscholing te doen. In veel beroepen is een dergelijke jaarlijkse scholing waarin nieuwe ontwikkelingen belicht worden, al verplicht om het beroep uit te mogen oefenen. Waarom zouden we dit niet voor elk beroep verplicht stellen: allemaal elk jaar verplicht een paar dagen of een week op cursus, van minister tot tegelzetter, van programmeur tot manager, van docent tot hoogleraar. Dat is pas kenniseconomie!

### *Een nieuwe opleiding: Master Software Engineering*

Er zijn natuurlijk meer mogelijkheden om bij te dragen aan de kenniseconomie. De faculteit Informatica van de Open Universiteit doet dat al met een scala aan losse *cursussen*, met *focusopleidingen* (bestaande uit een aantal inleidende cursussen op een bepaald gebied bv 'Informatica en maatschappij' of 'omputers en communicatienetwerken'), met *certified professional programs* (cursussen met extra begeleiding die soms in-house bij bedrijven gegeven worden, bijvoorbeeld 'Certified System Developer' of 'Certified Software Architect') en natuurlijk een volledige Bachelor en Master Informatica en ook nog samen met de faculteit Managementwetenschappen de Master 'Business Process Management and Information Technology'. In de Free Technology Academy werkt de OU samen met andere Europese universiteiten om studenten vanuit de hele wereld volledig on-line, via Open Educational Resources, technische cursussen over software ontwikkeling en netwerk technologie te laten volgen gebruik makend van open standaarden en Free Software.

Uit contacten met bedrijven blijkt dat er onverminderd behoefte is aan hoger opgeleiden op ICT gebied[49]. Uit contacten met studenten van de OU blijkt dat er behoefte is aan een kortere meer gespecialiseerde academische Master naast de algemenere academische Informatica Masteropleiding die een omvang heeft van 120 ec (vergelijkbaar met 2 jaar voltijdstudie).

Om die reden werkt de faculteit Informatica aan het opzetten van een Master Software Engineering met een omvang vergelijkbaar met 1 jaar voltijdstudie.

Het is mij echt een groot genoegen dat ik hieraan leiding mag geven. Er wordt door veel mensen hard aan gewerkt, Marleen Sint wil ik hier met name noemen omdat zij als domeincoördinator Software technologie hier zowel inhoudelijk als organisatorisch het meest bij betrokken is. De opzet is klaar. Het proces van aanvragen en accrediteren is in gang gezet. Gedeeltelijk kunnen bestaande vakken worden ingezet, gedeeltelijk zullen nieuwe vakken ontwikkeld moeten worden.

Voor de nadere invulling van diverse onderdelen van het curriculum zal samengewerkt worden met 3 andere universiteiten: met de Universiteit Utrecht (voortbouwend op de huidige samenwerking), met de Universiteit van Amsterdam (op het gebied van Software Evolutie) en met de Radboud Universiteit Nijmegen (op de gebieden Verificatie, Validatie en Security).

Hiermee wordt een unieke opleiding gecreëerd die Informatica professionals de gelegenheid geeft om, naast hun baan, op een open en flexibele wijze hun Software Engineering kennis en vaardigheden op academisch niveau te brengen. Ik ben ervan overtuigd dat het een hele mooie, academische opleiding zal worden.

### *Informatica op de middelbare school*

Aan de basis van onze kenniseconomie ligt natuurlijk het middelbaar onderwijs. In het middelbaar onderwijs is het vak Informatica voor HAVO en VWO sinds de invoering van de herziene tweede fase in 2007 uitgebreid waarbij het enerzijds een volwaardig profielkeuzevak voor het profiel Natuur en Techniek is geworden terwijl het anderzijds voor de ander profielen een keuzevak in het vrije deel is gebleven.

Ik voel me betrokken bij het middelbare schoolvak Informatica. Ik ben lid van een Vaknetwerk Informatica waar docenten van voortgezet en hoger onderwijs samen overleggen over mogelijke inhoudelijke vernieuwingen. Ik leid vanuit de Radboud Universiteit het CodeYard project[50] waarin scholieren de gelegenheid krijgen om samen te werken in Open Source projecten[51]. Nu al een vijftal jaren maken jaarlijks een vijftigtal scholieren hier gebruik van. Elk jaar ben ik weer onder de indruk van de snelheid waarmee ze zich de technieken eigen maken en van de creativiteit waarmee ze hun projecten uitvoeren. De resultaten worden beoordeeld door een vakjury die een prijs voor het beste project mag geven. Door de crisis valt het niet mee hier elk jaar weer sponsoring voor te verwerven. Een goede sponsor met een meerjarig commitment zou zeer welkom zijn.

Het middelbare schoolvak Informatica beoogt voor alle profielen een beeld te geven van het vak Informatica en de voor iedereen benodigde basiskennis bij te brengen. Gezien de mogelijkheid om het voor alle profielen te gebruiken is het niet ingevuld als een puur technisch vak maar als een multidisciplinair vak. Dit sluit ook aan bij de gegroeide praktijk waar leraren Informatica veelal zelf geen puur technische achtergrond hebben en door bij- en omscholing de Informatica bevoegdheid verworven hebben.

De consequentie hiervan is dat het vak in veel gevallen minder aantrekkingskracht heeft voor de in techniek geïnteresseerde leerling. Er zijn hoogleraren[52] die, enigszins radicaal, beweren dat Informatica op de middelbare scholen desnoods maar moet worden afgeschaft. Vanuit mijn betrokkenheid heb ik zowel via de vakbladen[53] als op het congres van I&I (de vereniging voor Informaticadocenten en Informaticaondersteuners in het voortgezet onderwijs) aan de discussie bijgedragen. Ik ben het niet eens met het radicale voorstel om Informatica dan maar af te schaffen. Integendeel, het is een belangrijk vak en een mooi vak dat het verdient om een prominente rol in het middelbaar onderwijs te spelen.

Maar het is wel nodig dat er verbeteringen komen. Ik wil hier een aantal elementen voor een mogelijk verbetervoorstel aandragen:

- *Verplichte jaarlijkse bijscholing voor Informatica docenten*.
  Dit sluit aan bij mijn eerdere voorstel over de kenniseconomie. Voor informatietechnologie gerelateerde sectoren (welke sector is dit eigenlijk niet) is dit bij uitstek nodig.
- *Geef scholieren de gelegenheid als alternatief onderdeel een volledig informaticavak bij de universiteit te volgen*.
  De kwaliteit van de inhoud en de tentaminering is dan geborgd door de universiteiten. Diverse universiteiten bieden deze mogelijkheid. De OU doet dit via het Academic Experience programma, een programma dat leerlingen de gelegenheid geeft via afstandsonderwijs een universitair vak te volgen. Academic Experience wordt door 25 scholen, verspreid over heel Nederland, aangeboden aan hun leerlingen en 2 instituten voor hoogbegaafdheid adviseren haar cliënten het studiemateriaal van de Open Universiteit. Sinds de start in 2006 hebben 388 leerlingen hiervan gebruik gemaakt.
- *Stel voor een deel van de stof een centraal examen in en laat de onderwerpen hiervan rouleren*.
  Dit houdt het vak up-to-date en geeft de docenten extra motivatie om te investeren in het bestuderen van een nieuw onderwerp. Door dit onderwerp 3 tot 5 jaar in het centraal examen te laten staan is de investering steeds de moeite waard. Ook bevordert het de investering in het ontwikkelen van cursusmateriaal voor de gekozen onderwerpen. Dit geeft ook kansen om het

multidisciplinaire karakter van het vak goed in te vullen zodat het vak past in alle profielen.
Daarnaast opent dit een natuurlijke mogelijkheid om meer te differentiëren naar inhoud en niveau voor HAVO en VWO.

- *Vergroot de mogelijkheden voor het verkrijgen van de Informatica lesbevoegdheid*.
  Bijvoorbeeld door meer universiteiten meer educatieve minoren te laten aanbieden die tot een (extra) lesbevoegdheid Informatica kunnen leiden.
- *Stel een Informatica overlegorgaan in waar vertegenwoordigers van middelbare schooldocenten, didactici, hbo's, universiteiten en bedrijven samen voorstellen maken voor vernieuwing en stimulering van het Informatica onderwijs op de middelbare scholen*.
  De Informatica beroepsgroep is relatief slecht georganiseerd en vaak zeer verdeeld. Dat moet veranderen.

Iedereen kan voorstellen doen voor wat nodig is voor verbetering. Maar wat met meest nodig is, is dat we het samen eens worden over wat nodig is.

### *Heen en weer tussen RU en OU*

Als laatste gedeelte van deze lezing wil ik u uitleggen hoe bevoorrecht ik me voel dat ik werkzaam kan zijn zowel aan de Radboud Universiteit Nijmegen als aan de Open Universiteit Nederland.

Bij de RU maak ik deel uit van het instituut voor Informatica en Informatiekunde. Dit instituut heeft in de recente universitaire onderzoekbeoordeling de hoogste beoordeling gekregen van alle Informatica instituten van Nederland. De afdeling Digital Security waar ik deel vanuit maak, heeft voor elk beoordeeld aspect de hoogst mogelijke beoordeling gekregen.

Bij de OU maak ik deel uit van de faculteit Informatica. Bij de recente beoordeling in de keuzegids hoger onderwijs heeft de Bachelor opleiding Informatica de hoogste score gekregen van alle universiteiten van Nederland.

Ik reis dus heen en weer tussen het beste onderzoek van Nederland en het beste onderwijs van Nederland. Wat wil je nog meer?

Het ligt voor de hand: ik wil eraan bijdragen dit zo blijft maar ook wil ik bijdragen aan de verbetering van het Informaticaonderwijs bij de RU en aan het verbeteren van het Informaticaonderzoek bij de OU.

Bij de RU speelt, naar aanleiding van een gewijzigde financieringsstructuur, de discussie over het verbeteren van de onderwijsrendementen. Overheersend hierbij zijn de geluiden dat de studie voor de studenten minder vrijblijvend moet worden. Een bindend studieadvies wordt overwogen. Vanuit mijn ervaring bij de OU breng ik in dat het ook voor de docenten minder vrijblijvend moet zijn en dat er, door de student meer centraal te stellen, verbeteringen in de opzet, de organisatie, de begeleiding en de uitvoering van het onderwijs moeten komen.

Bij de OU speelt, naar aanleiding van de recente wetswijziging waarbij onderzoek nadrukkelijk tot de taak van de OU is gaan behoren, de discussie hoe het onderzoek uitgebouwd kan worden. Vanuit mijn ervaring bij de RU breng ik in dat de onderzoekcultuur verbeterd moet worden, dat naast onderzoek op onderwijsgebied ook een belangrijk deel van het onderzoek moet aansluiten bij de bedrijfspraktijk zodat studenten vanuit hun beroepspraktijk hieraan kunnen bijdragen en dat de ambitie gesteld moet worden om het onderzoek tot een zodanig niveau en omvang te brengen dat deelname aan een landelijke onderzoek assessment mogelijk wordt.

### *Afsluiting*

Al met al is ‘’je leven lang computeren’’ een leven vol ploeteren en foeteren maar de eerste tekenen van verbetering tekenen zich aan. Dankzij wetenschappelijk onderzoek wordt het langzamerhand steeds meer mogelijk om niet alleen het software proces te analyseren maar ook het software product. Hopelijk zullen we hierdoor in de toekomst dankzij zorgvuldige analyse gebruikmakend van betrouwbare tools met recht meer kunnen gaan vertrouwen op software.
Maar daarvoor is er nog veel te onderzoeken en nog veel te leren.

### *Dankwoord*

Tot slot, wil ik het College van Bestuur van de Open Universiteit Nederland en de decaan van de faculteit Informatica danken voor het in mij gestelde vertrouwen. Ook dank ik het bestuur van het Nijmeegse onderzoekinstituut Instituut voor Informatica en Informatiekunde (iCIS) en de Faculteit Natuurwetenschappen, Wiskunde en Informatica voor het mogelijk maken van de constructie waarbij ik aan beide universiteiten werkzaam kan zijn. Het eerste jaar is mij in ieder geval zeer goed bevallen. Ik hoop dat dit gevoel wederzijds is.

Verder wil ik de hoogleraren bedanken bij wie ik de afgelopen jaren in Nijmegen heb gewerkt. Ik heb van hen veel geleerd. Ik dank professor Koster voor het bijbrengen van passie voor het programmeren en de didactiek van het programmeren, professor Boute voor de passie voor het functioneel programmeren

waar ik mijn wiskundige achtergrond goed in kwijt kan, professor Barendregt voor de passie voor het theoretisch fundament, professor Plasmeijer voor de passie voor tool-gericht onderzoek en professor Jacobs voor de passie voor maatschappelijke impact vanuit academische expertise zowel op het gebied van software kwaliteit als op het gebied van privacy en security.

Natuurlijk dank ik al mijn collega's, zowel informatici, onderzoek(st)ers en docenten als managers en ondersteun(st)ers (hier wil ik graag met name Chrisja en Maria bedanken zonder wie de zaal leeg was gebleven omdat ik dan nu nog steeds met de adressen voor de uitnodigingen in de weer was geweest). Ik dank al die mensen zowel aan de Radboud Universiteit Nijmegen als aan de Open Universiteit Nederland en bij de bedrijven waar ik contact mee heb, voor de prettige samenwerking, de goede gesprekken en de samen geschreven artikelen. Samen staan we sterk!

Ook wil ik de studenten van zowel de Open Universiteit als de Radboud Universiteit bedanken. Toekomstige ontwikkelingen in de IT zullen voor een groot deel door afgestudeerde studenten worden ingezet. Het is mij een genoegen om daar via jullie een beetje aan bij te dragen.

Ook dank ik al mijn vrienden en bekenden hier aanwezig maar in het bijzonder Bernard, Thijn en Kees en de mensen van de dansgroep de Pierewaaiers: jullie vormen al pakweg 30 jaar een vast punt in mijn leven. Laten we nog minstens 30 jaar zo doorgaan.

Uiteraard dank ik ook mijn familie. Een bijzondere band geeft een bijzondere ondersteuning: de prettige ongedwongenheid bij de Dirvens en de diepe vertrouwdheid bij de van Eekelens. Helaas zijn ze er niet meer allemaal bij: mijn vader, mijn schoonvader, mijn moeder, mijn oudste broer, alle vier zijn ze overleden, elk op hun eigen moment. Ik vond hen geweldig en zij zouden dit vast ook geweldig gevonden hebben. Ik denk nu ook aan hen.

Nog dichterbij is mijn eigen gezin. Mijn kinderen, Laura en Paul. Jullie weten het eigenlijk wel maar soms mag het ook hardop gezegd worden. Jullie zijn het licht in mijn leven. Ik ben supertrots op jullie allebei. Ik hoop dat jullie vandaag ook een beetje trots op mij zijn.

Last but by no means least: Lieve Lianne, jij bent mijn baken, mijn spiegel, en mijn inspiratie maar bovenal ben je mijn geliefde: je bent mijn eeuwigdurende vlam! Ik hoop dat we nog vele jaren van het samenzijn met elkaar mogen genieten.

Afsluitend nog een laatste persoonlijke gedachte over 53. Het is vandaag 5 maart, 5 3, samengevoegd 53 en, zoals mijn zoon Paul heeft bedacht, is 53 het enige priemgetal dat een concatenatie (een samenvoeging) is van een priemgetaltweeling (5 en 3). Een heel bijzonder getal dus. Maar er is meer. Mijn vader overleed toen hij 53 jaar en 2 maanden oud was. Ik ben nu zelf 53 jaar, en 3 maanden oud. Vandaag is voor mij 53 veranderd van een droevige mijlpaal in een vreugdevolle mijlpaal.

"Ik heb gezegd".

NOTEN

[1] Voor info over de open universiteit op iTunes U: http://www.ou.nl/iTunesU.

[2] Marleen Brinks. Aggression gegen Computer. Eine wissenschaftliche Untersuching eines alltäglichen Phänomens. FernUniversität Hagen. ISBN: 3-89821-550-4. Ibidem-Verlag. Stuttgart 2005.

[3] Rinus Plasmeijer, M.J. and Marko van Eekelen. *Functional Programming and Parallel Graph Rewriting,* Addison Wesley, Reading MA, ISBN 0-201-41663-8. 571 pages 1993.

[4] Marko van Eekelen ( Ed.) Trends in Functional ProgrammingVolume **6**. Selected papers from the Sixth Symposium on *Trends in Functional Programming* (TFP 2005). Intellect Books. ISBN 9781841501765. 2007.

[5] Maarten de Mol, Marko van Eekelen, Rinus Plasmeijer. A Single-Step Term-Graph Reduction System for Proof Assistants. Andy Schürr, Manfred Nagl, Albert Zündorf (eds). Applications of Graph Transformations with Industrial Relevance, Third International Symposium, AGTIVE 2007. Proceedings of Revised Selected and Invited Papers. Kassel, Germany, 2007, Lecture Notes in Computer Science, Springer Verlag. Vol. **5088**, pp 184-200. 2008.

[6] Marko van Eekelen, Sjaak Smetsers, Rinus Plasmeijer, Graph Rewriting Semantics for Functional Programming Languages, In Proc. of CSL '96, Fifth Annual conference of the *European Association for Computer Science Logic* (EACSL), Utrecht, Dirk van Dalen Ed., Lecture Notes in Computer Science, Springer Verlag. Vol. **1258**, pp. 106-128. 1997.

[7] Sjaak Smetsers, Erik Barendsen, Marko van Eekelen and Rinus Plasmeijer. Guaranteeing safe destructive updates through a type system with uniqueness information for graphs. In Proc. of *Graph Transformations in Computers Science.* Dagstuhl Castle, Germany, Schneider and Ehrig Eds., Lecture Notes in Computer Science, Springer Verlag. Vol. **776**, pp. 358-379. 1994.

[8] Artem Alimarine, Sjaak Smetsers, Arjen van Weelden, Marko van Eekelen, Rinus Plasmeijer: There and back again: arrows for invertible programming. In Daan Leijen (Ed.): Proceedings of the ACM *SIGPLAN Workshop on Haskell*, Tallinn, Estonia. pp. 86-97, 2005.

[9] Walter de Hoon, Luc Rutten and Marko van Eekelen: Implementing a Functional Spreadsheet in Clean. *Journal of Functional Programming* **5**(**3**): 383-414. 1995.

[10] Marko van Eekelen, en Rinus Plasmeijer, Constructing Medium Sized Efficient Functional Programs in CLEAN, In Proc. of First International Spring School on Advanced Functional Programming Techniques, Båstad, Sweden, Jeuring and Meijer Eds., Lecture Notes in Computer Science, Springer Verlag. Vol. **925**, pp. 183-228, 1995.

[11] Peter Achten, Marko van Eekelen, Rinus Plasmeijer, Arjen van Weelden. GEC: a toolkit for Generic Rapid Prototyping of Type Safe Interactive Applications. In Vene, Varmo; Uustalu, Tarmo (Eds.). *Advanced Functional Programming 5th International School*, (AFP), Tartu, Estonia, August 14-21, Revised Lectures. Lecture Notes in Computer Science, Springer Verlag. Vol. **3622**, pp 210-245. 2004.

[12] Marko van Eekelen, Maarten de Mol: Proof Tool Support for Explicit Strictness. In Andrew Butterfield, Clemens Grelck, Frank Huch (Eds.): *Implementation and Application of Functional Languages*, 17th International Workshop, IFL 2005, Dublin, Ireland, Revised Selected Papers. Lecture Notes in Computer Science, Springer Verlag. Vol. **4015**. 2006.

[13] Ron van Kesteren, Marko van Eekelen, Maarten de Mol. Proof Support for General Type Classes, Trends in Functional Programming Volume 5, Selected papers from the Fifth Symposium on *Trends in Functional Programming* (TFP 2004), editor Hans Wolfgang Loidl. Intellect Publishers, pp 1-16. This paper received by unanimous vote the **Best Student Paper Award** of TFP04.

[14] Rinus Plasmeijer and Marko van Eekelen. Keep it Clean: A unique approach to functional programming, ACM *Sigplan Notices*, Volume **34** (6), pp 23-31, ACM Press, June 1999.

[15] Pieter Koopman, Marko van Eekelen and Rinus Plasmeijer, Operational machine specification in a functional programming language, In *Software-Practice and Experience* 25:5, pp. 463-499. 2005.

[16] Maarten de Mol, Marko van Eekelen, Rinus Plasmeijer. Theorem Proving for Functional Programmers - SPARKLE: A Functional Theorem Prover. In: Arts, Th., Mohnen M., eds. Proceedings of the 13th International Workshop on the *Implementation of Functional Languages*, Selected Papers, Älvsjö, Sweden, September 24-26, Lecture Notes in Computer Science, Springer Verlag. Vol. **2312**, pages 55-71. 2001.

[17] Malcolm Dowse, Andrew Butterfield, Marko van Eekelen, Reasoning about Deterministic Concurrent Functional I/O. Selected Papers of the 16th International Workshop Proceedings of *Implementation and Application of Functional Languages*, IFL'04, Lübeck, Germany, Lecture Notes in Computer Science, Springer Verlag. Vol. **3474**, pp 177-195. 2004.

[18] Maarten de Mol, Marko van Eekelen, Rinus Plasmeijer. Proving Properties of Lazy Functional Programs with SPARKLE. In Zoltan Horvath ed: 2nd *Central-European Functional Programming School*. CEFP 2007. Cluj-Napoca, June 23-30, 2007, Lecture Notes in Computer Science, Springer Verlag. Vol. **5161**. pp 41-86. 2008.

[19] Rinus Plasmeijer, Marko van Eekelen, Jan Vytopil and Theo Schouten, *Parallel Processing*, videotape on the advantages of parallel processing highlighting several transputer applications, Chriet Titulaer Produkties bv. 1990.

[20] Pieter Edelman, 'Wiskundige notatie temt multicore processor', Interview met Kevin Hammond en Marko van Eekelen. In *Bits & Chips*, 18 december, 11e jaargang, nr. **19/20**, pp 40-42. 2009.

[21] Henk Barendregt, Marko van Eekelen, Pieter Hartel, Bob Hertzberger, Rinus Plasmeijer and Wim Vree. The Dutch Parallel Reduction Machine Project, In *Future Generations Computer Systems 3*, pp. 261-270. Proceedings of the International Conference on Frontiers of Computing. Amsterdam, December 1987.

[22] Marko van Eekelen, Jan Tretmans, and Tim Willemse. Vastleggen van Kwaliteit is Essentieel. In *Automatisering gids*, **39**(11):13, 2005.

[23] http://www.mitre.org

[24] http://www.sans.org

[25] Petra Heck. Martijn Klabbers, Marko van Eekelen. A Software Product Certification Model. In *Software Quality Journal*. Springer Verlag. Volume **18**, nr 1, March 2010, pp 37-55.

[26] https://www.artemis-ju.eu/charter en http://charterproject.ning.com/

[27] Tom van den Broek and Julien Schmaltz. Towards a formally verified network-on-chip. In Proceedings of 9th International Conference on *Formal Methods in Computer-Aided Design,* FMCAD 2009, 15-18 November 2009, Austin, Texas, USA}, IEEE, 2009.

[28] Sjaak Smetsers and Marko van Eekelen. *LaQuSo: Using Formal Methods for Analysis, Verification and Improvement of Safety Critical Software.* Quarterly Magazine of the European Research Consortium for Informatics and Mathematics. Special Theme Safety-Critical Software (ed. Pedro Merino and Erwin Schoitsch). *ERCIM NEWS. Volume **75***, October 2008. pp 38-39.

[29] Marko van Eekelen, Stefan ten Hoedt, René Schreurs, Yaroslav S. Usenko. Analysis of a Session-Layer Protocol in mCRL2. Verification of a Real-Life Industrial Implementation. In P. Merino and S. Leue (eds). *Proceedings 12th International Workshop on Formal Methods for Industrial Critical Systems* (FMICS 2007). Lecture Notes in Computer Science, Springer Verlag. Vol. **4916**. pp. 182-199. 2008.

[30] Leonard Lensink, Sjaak Smetsers, Marko van Eekelen. Machine Checked Formal Proof of a Scheduling Protocol for Smartcard Personalization. In P. Merino and S. Leue (eds). *Proceedings 12th International Workshop on Formal Methods for Industrial Critical Systems* (FMICS 2007). Lecture Notes in Computer Science, Springer Verlag. Vol. **4916**. pp 115-132.

[31] Bernard van Gastel, Leonard Lensink, Sjaak Smetsers, and Marko van Eekelen. Reentrant Readers-Writers: A Case Study Combining Model Checking and Theorem Proving. In D. Cofer and A. Fantechi (eds). Revised selected papers of the 13th International Workshop on *Formal Methods for Industrial Critical Systems* (FMICS 2008). L'Aquila, Italy. Lecture Notes in Computer Science. Springer Verlag, Vol. **5596**. pp. 85-103. 2009. This paper received the FMICS2008 Best Paper Award recognizing it as the '**Best Software Science Paper**'.

[32] Erik Schierboom, Alejandro Tamalet, Hendrik Tews, Marko van Eekelen, Sjaak Smetsers. Preemption Abstraction - A Lightweight Approach to Modelling Concurrency. In M. Alpuente and B. Cook (eds). Proceedings 14th International Workshop on *Formal Methods for Industrial Critical Systems* (FMICS 2009). Lecture Notes in Computer Science. Springer Verlag, Vol. **5825**. Springer. pp 149-164. 2009.

[33] Luc Rutten and Marko van Eekelen. Efficient and Formally Proven Reduction of Large Integers by Small Moduli. In ACM *Transactions on Mathematical Software*. ACM Press. 19 pages. to appear 2010.

[34] Marko van Eekelen, Olha Shkaravska, Ron van Kesteren, Bart Jacobs, Erik Poll and Sjaak Smetsers. AHA: Amortized Heap Space Usage Analysis. Project Paper. In Marco Morazán and Henrik Nillsson (eds). *Trends in Functional Programming* volume **8**, Selected Papers of the Eighth Symposium on Trends in Functional Programming, TFP 2007, New York City, USA. Intellect Publishers 2008. pp. 36-53. 2008

[35] http://www.aha.cs.ru.nl

[36] Alejandro Tamalet, Olha Shkaravska, and Marko van Eekelen. Size Analysis of Algebraic Data Types. in Peter Achten, Pieter Koopman, and Marco T. Morazán. *Trends in Functional Programming* volume **9**, Selected Papers of the Ninth Symposium on Trends in Functional Programming, TFP 2008, Radboud University Nijmegen, 2008, Intellect Publishers. pp. 33-49. 2009.

[37] Dezelfde Alan Turing stond aan de leiding van het Engelse team dat de Enigma code kraakte tijdens de tweede wereldoorlog en naar dezelfde Alan Turing is de Turing Award (de meest prestigieuze Informatica prijs) vernoemd.

[38] Ron van Kesteren, Olha Shkaravska, Marko van Eekelen, Inferring static non-monotonically sized types through testing. In Rachid Echahed (ed.) Revised Selected Papers of the 16th international Workshop on Functional and (Constraint) Logic Programming (WFLP) 2007. *Electronic Notes in Theoretical Computer Science*. *(ENTCS)*. Volume 216C. pp. 45-63. Elsevier, 2008.

[39] Olha Shkaravska, Marko van Eekelen, and Ron van Kesteren. Polynomial Size Analysis of First-Order Shapely Functions. *Logical Methods in Computer Science,* volume **5**, issue 2, paper 10, 35 pages, Creative Commons License, 2009.

[40] Olha Shkaravska, Marko van Eekelen, Alejandro Tamalet. Collected Size Semantics for Functional Programs over Lists. In Sven-Bodo Scholz. Selected Papers of the 20th Symposium on *Implementation and Application of Functional Programming*, IFL 2008, University of Hertfordshire, UK, Lecture Notes in Computer Science. Springer Verlag, to appear 2010.

[41] Steffen Jost, Kevin Hammond, Hans-Wolfgang Loidl, Martin Hofmann: Static determination of quantitative resource usage for higher-order programs. Proceedings of the 37th *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, (POPL), Madrid, Spain, pp. 223-236. 2010.

[42] Sumit Gulwani. SPEED: Symbolic Complexity Bound Analysis. In Ahmed Bouajjani and Oded Maler (eds) 21st International Conference *on Computer Aided Verification* (CAV), Grenoble, France, Lecture Notes in Computer Science, volume **5643**, pp 51-62. Springer Verlag, 2009.

[43] Patrick Baillot, Jean-Yves Marion, Simona Ronchi Della Rocca: Guest editorial: Special issue on implicit computational complexity. *ACM Trans. Comput. Log.* **10** (4). 2009.

[44] Marko van Eekelen and Olha Shkaravska (eds). Proceedings of the First International Workshop on *Foundational and Practical Aspects of Resource Analysis* (FOPARA). Lecture Notes in Computer Science, Springer Verlag, to appear 2010.

[45] Marko van Eekelen, Engelbert Hubbers. Slimme Meters: Energie wordt ICT? *Energie+*, Jaargang 28, nr 4, pp 23-24. 2008.

[46] Marko van Eekelen, Engelbert Hubbers. *Security, Privacy and AMR systems: Small Issues or Sources of Future Problems?* Conference Proceedings of Metering Europe 2008. 22-24 September 2008. Amsterdam RAI. page 41. 2008.

[47] http://www.wikiwijs.nl

[48] Marko van Eekelen and Maarten de Mol. Proving Lazy Folklore with Mixed Lazy/Strict Semantics. In: (eds) Erik Barendsen, Venanzio Capretta, Herman Geuvers and Milad Niqui. *Reflections on Type Theory, λ-calculus, and the Mind.* Essays dedicated to Henk Barendregt on the Occasion of his 60th Birthday. Radboud University Nijmegen. pp. 87-101. 2007

[49] Kim de Vries. OU-studie Informatica wordt steeds gewilder. Interview met Marko van Eekelen. *Computable*. 20 april 2009.

[50] http://www.codeyard.net

[51] Adriaan de Groot, Sebastian Kügler, Donna Metzlar, Jasper Stein, Marko van Eekelen. Exposing High-School Students to Open Source Development with CodeYard. Working Joint IFIP Conference: WG3.1 Secondary Education, WG3.5 Primary Education. *"Informatics, Mathematics, and ICT: a 'golden triangle'"*. College of Computer and Information Science Northeastern University Boston, Massachusetts, USA, 27th – 29th June 2007. ISBN 978-0-615-14623-2.

[52] Chris Nap. "Schaf Informatica op middelbare school af". Interview met Prof. Jan Friso Groote, opleidingsdirecteur Informatica TU Eindhoven. *Automatisering Gids*, 5 september 2008.

[53] Chris Nap, 'van Eekelen: Informatici, daal af uit ivoren toren*'*. *Automatisering Gids*. 21 November. pp 24-25. 2008.