I/O VIVAT

JAARGANG 26
NUMMER 4

Micro vs. Monolitische kernels

'Hét debat' tussen de grondleggers van Linux en Minix

Leven op het web

Hebben wij desktopapplicaties nog wel nodig?

VerCors project

Programma's met meerdere executie-threads, maar zonder fouten

Bad programming habbits

Wie heeft ze niet?

Kingpin

Het verhaal van een alleenheersende hacker

En verder...

Werking/beveiliging OV-chipkaart
Software hergebruik bij agile ontwikkelomgevingen
X86 versus ARM
Van de voorzitter
CλaSH
Katern: Immorailty

Inter-Actief

Colofon

Jaargang 26, nummer 4,
juni 2011
ISSN: 1389-0468

I/O Vivat is het populair-wetenschappelijke tijdschrift van I.C.T.S.V. Inter-Actief, de studievereniging voor Technische Informatica, Bedrijfsinformatietechnologie en Telematica van de Universiteit Twente. I/O Vivat verschijnt vier maal per jaar en heeft een oplage van 1800 exemplaren.

Hoofdredactie:
Rick van Galen

Redactie:
Bas Stottelaar, David Huistra, Herman Slatman, Michel Brinkhuis, Stijn van Winsen

Vormgeving:
Niels Witte

Gastschrijvers:
Christiaan Baaij, Berend van den Brink, Marieke Huisman, Matthijs Kooijman, Rom Langerak, Jacco Roest, Wouter Spoelstra, Symposium Commissie

Voor vragen, suggesties en tips is I/O Vivat bereikbaar via e-mail op vivat@inter-actief.net, telefonisch op 053-489 3756 of per post:
Studievereniging Inter-Actief
Postbus 217
7500AE Enschede

De studievereniging wil de adverterende bedrijven bedanken voor de samenwerking.

Drukwerk:
Drukkerij van den Bosch & Fikkert

I/O VIVAT

Redactioneel

Microsoft liet vandaag nieuws vrij over de aankondiging van een nieuwe versie van Windows. Daarbij zette het uiteen wat haar visie was over “technology transitions” door de jaren heen: van logge desktopcomputers bedoeld voor tekstverwerking zijn we overgestapt naar in-house client/serversystemen en verder naar internet en cloud-applicaties. Vandaag de dag zouden we in overstap zijn naar de “unified ecosystems”-fase, waarin je op je smartphone, tablet en laptop dezelfde mogelijkheden en toegang hebt.

Kenmerkend voor deze overgang is het grotere gebruik van zuinige maar krachtige ARM-processors. Onderdeel van Microsofts aankondiging is een versie van Windows die draait op ARM-processoren – iets dat het steeds grotere belang van de ARM-chips illustreert. Gaat ARM de x86-wereld veroveren? Lees er alles over in het artikel op pagina 26.

Een ander kenmerk van deze overgang is het loslaten van applicaties die voor een specifiek platform zijn geschreven. Niet meer afhankelijk van een editor voor Mac OS X, die ene bibliotheek onder Linux of deze ontwerptool onder Windows: de applicaties die je benadert zijn geschreven in HTML en Javascript en draaien op een webserver die je benadert. Zijn webapplicaties al zo volwassen dat je er helemaal afhankelijk van kunt zijn? Op pagina 30 ondervind je mijn ervaringen hiermee.

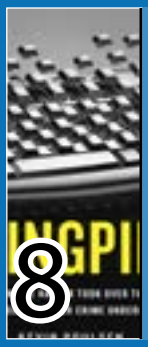
Als men kijkt naar ontwikkeling van deze applicaties zien we ook een verandering in ontwikkelingsstijl: in plaats van grote projecten die met een watervalmodel worden uitgevoerd, worden er snel prototypes opgeleverd die daarna worden uitgebreid. Met deze modellen is het vaak belangrijk zo snel mogelijk code op te leveren, wat nog wel eens fouten en ergernissen kan opleveren. Een klein overzicht volgt op 10.

Dit is slechts een kleine greep uit de artikelen in deze extra dikke Vivat-editie. We hebben onder andere ook artikelen over de OV-chipkaart, parallelle verificatie, hardwarebeschrijvingstalen en een eerste blik op het symposium Immorality. Een geschikte Vivat om mee te nemen op vakantie dus!

Tot volgend jaar,

Rick van Galen
Hoofdredacteur

Artikelen



Kingpin

Herman Slatman

CARDING, CYBERSECURITY, CREDITCARDS, MAX BUTLER, HACKER

8

Bad programming habbits

Stijn van Winsen

PROGRAMMEREN, JAVA, PHP, SLECHTE GEWOONTES

10

Micro vs. Monolitische kernels

Michel Brinkhuis

MICROKERNEL,MONOLITISCHE-KERNEL, LINUX, MINIX

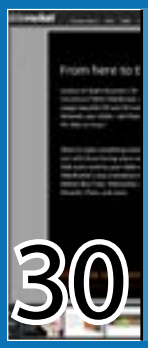
14

X86 versus ARM

Bas Stottelaar

X86, ARM, INTEL, SOC

26



Leven op het web

Rick van Galen

EEN FLINKE, LIJST MET, KEY-WORDS, GESCHEIDEN DOOR, KOMMA

30

Columns & ENIAC

Onderwijsvernieuwing en Scandinavië

Rom Langerak

17

Nieuws

BeursgangLinkedIn

6

Meerderheid kamer voor voorstel netneutraliteit

6

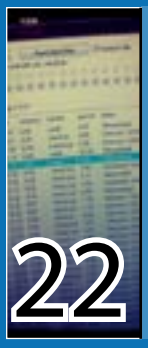
“WindowsPhone 7 in 2013 groter dan Android”

7

18 VerCors project

Marieke Huisman

PROGRAMMAVERIFICATIE, SPECIFICATIE, CONCURRENCY, LOGICA, SEPARATION LOGIC



22 Werking/beveiliging OV-chipkaart

David Huistra

TRANS LINK SYSTEMS, MYFARE CLASSIC, HACKEN,INDELING, FRAUDEURS

25 Topicus: Softwarehergebruik.

Wouter Spoelstra

SOFTWARE, ONTWIKKELOMGEVING, HERGEBRUIK, TOPICUS

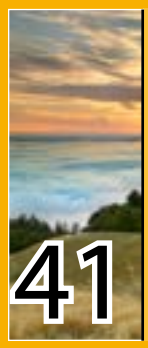
35 Van de voorzitter

Jacco Roest

38 CλaSH

Christiaan Baaij & Matthijs Kooijman

CλASH, HASKELL, CAES, HARDWAREBESCHRIJVING



41 ENIAC: Van de Voorzitter

Berend van den Brink

Symposium Immorality

45 Immorality

Symposium Commissie

IMMORALITY, SOCIAL ENGINEERING, HACKING, PRIVACY

46 Interview: David Nieborg

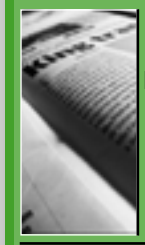
GAMEMAKERS,GAMES,FEITELIJKE ASPECTEN, GEWELD, MEDIA

En verder...

36 Op bezoek bij Quinity

42 Op bezoek bij Shell

47 Volgende keer in I/O Vivat



Nieuws

I/O Vivat redactie

LINKEDIN, BEURS, NETNEUTRALITEIT, KPN, WHATSAPP, WINDOWS PHONE 7, ANDROID

Nieuws

Beursgang LinkedIn

De concurrentie tussen de social media netwerken is groot op het moment. Facebook is in de Verenigde Staten en in vele andere landen oppermachtig, en ook in Nederland is men aan een flinke opmars bezig. Dit laatste gaat met name ten koste van Nederlands eigen sociale netwerk Hyves. LinkedIn, het meer zakelijk georiënteerde sociale netwerk, kondigde onlangs een beursgang aan. Dit met het doel concurrent Facebook voor te blijven. De beursgang komt niet geheel als een verrassing, omdat men in januari dit jaar al een zogenaamde IPO (initial public offering) indiende bij de Amerikaanse beursautoriteit SEC.

Initieel maakte LinkedIn bekend ongeveer 7,9 miljoen aandelen uit te geven, tegen een waarde van 45 dollar. Op 19 mei dit jaar ging het bedrijf daadwerkelijk naar de beurs. De aandelen maakten in de eerste twee uur na de beursgang een stijging door van zo'n 160%. Gedurende de eerste dag daalde de koers ook weer wat, maar alsnog eindigde het aandeel aan het einde van de dag met een verdubbelde waarde.

Al met al heeft het Amerikaanse bedrijf, op de NYSE-beurs nu bekend als LNKD, een ruime 350 miljoen dollar opgehaald met de beursgang. De profielensite telt inmiddels meer dan 100 miljoen gebruikers wereldwijd. Dat is een indrukwekkend getal, vooral gezien het feit dat de site in 2003 live ging en in de eerste maand 4500 leden wist binnen te halen. Op dit moment verwelkomt LinkedIn ongeveer een miljoen leden per maand.

Analisten schatten de waarde van LinkedIn op zon 3 miljard dollar, meldt Tweakers.net.

Bron: http://www.tijd.be/nieuws/geld_-_beleggen_markten/IPO-gekte_op_beurzen.9060152-3452.art?ckc=1

Meerderheid kamer voor voorstel netneutraliteit

Een meerderheid van de Tweede Kamer heeft het voorstel van GroenLinks gesteund om de plannen voor heffingen op mobiele diensten als voip en chatten te blokkeren. In de Telecomwet moet opgenomen worden dat netneutraliteit gegarandeerd moet worden.

KPN gaf onlangs aan van plan te zijn klanten extra te laten betalen voor diensten als de chat-applicatie WhatsApp. Dit zou gedaan worden met behulp van deep packet inspection. Na protesten trok het bedrijf uiteindelijk de plannen in, maar maakte mobiel internet voor haar klanten wel duurder.

Volgens de Tweede Kamer mogen aanbieders echter bepaalde diensten die via mobiel internet lopen tariferen. Bruno Braakhuis, GroenLinks-lid, stelt voor om in de Telecommunicatiewet op te nemen dat telecomaanbieders dit niet op basis van het soort gebruik van dataverkeer mogen differentiëren, en daarmee netneutraliteit te garanderen. Aanbieders mogen hierdoor geen diensten blokkeren of voorrang geven op het netwerk.

Volgens Braakhuis moeten telecombedrijven zelf met nieuwe verdienmodellen komen om de dalende inkomsten op bel- en smsverkeer het hoofd te bieden, maar niet door controle van het dataverkeer. Minister Verhagen heeft eerder gezegd dat hij vindt dat internetdiensten beschikbaar moeten blijven en dat hij zich kan voorstellen dat de consumenten daarvoor meer moeten betalen.

“Windows Phone 7 in 2013 groter dan Android”

Het Amerikaanse onderzoeksbedrijf Pyramid Research verwacht dat in 2013 Windows Phone 7 groter zal zijn dan elk ander mobiel besturingssysteem. Pyramid verwacht een flinke groei van WP7 doordat het Finse bedrijf Nokia haar nieuwe generatie toestellen gaat uitrusten met het mobiele besturingssysteem.

Het is niet alleen WP7 dat zal groeien. Ook Google Android zal blijven groeien, terwijl BlackBerry OS van RIM en Apple iOS juist marktaandeel gaan verliezen.

Nokia heeft onder de consument nog steeds een sterke naam omdat het in het verleden heeft laten zien kwalitatieve en goedkope telefoons te kunnen produceren. Daarnaast heeft het nauwe banden met telecomoperators wereldwijd, die uiteindelijk besluiten wat ze gaan verkopen.

Begin dit jaar heeft IDC Nokia aangewezen als de voornaamste leverancier van mobiele telefoons in het vierde kwartaal van 2010. Ondanks dat de populariteit van Symbian sterk aan het afnemen is, wist men het marktaandeel in 2010 iets te vergroten door 453 miljoen telefoons te verkopen, tegenover 432 miljoen in het jaar daarvoor.

Windows Phone 7 wordt niet door iedere partij als het meest populaire platform geprezen. Het zal daarom menig persoon verbazen dat Pyramid Research dit voorspelt. Toch zijn er een aantal partijen die daar heel anders over denken en dit ook kunnen onderbouwen. Microsoft heeft in ieder geval laten zien dat Windows Phone 7 een flinke verbetering is ten opzichte van Windows Mobile 6.

Bron: http://computerworld.nl/article/12884/waarom-windows-phone-7-het-gaat-maken.html

Kingpin

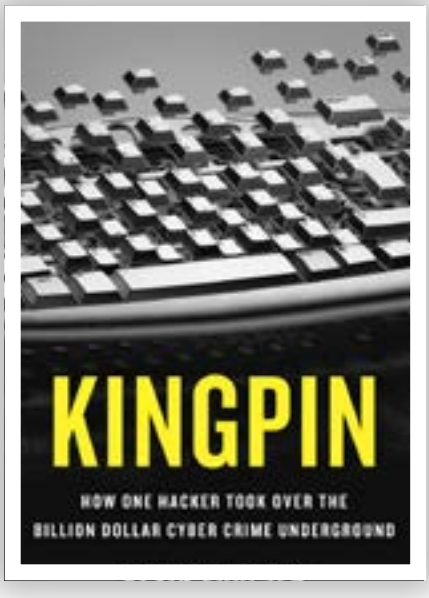
Herman Slatman
Redacteur I/O Vivat

CARDING, CYBERSECURITY, CREDITCARDS, MAX BUTLER, HACKER

Het verhaal van een alleenheersende hacker

Kevin Poulsen weet als geen ander, als ex-hacker en ex-gedetineerde, hoe het is om te leven als hacker die in aanraking is gekomen met justitie. Sinds zijn gevangenisstraf heeft hij zijn leven echter gebeterd en tegenwoordig is hij hoofdredacteur voor Wired News. Hij heeft zich daar onderscheiden als één van de beste onderzoeksverslaggevers op het gebied van cybercrime en privacy. Met het schrijven van Kingpin brengt hij voor het eerst zijn expertise in boekvorm naar buiten. In Kingpin schrijft Poulsen over het waargebeurde verhaal van Max 'Vision' Butler, een white hat hacker die zijn witte hoed inruilde voor een zwart exemplaar. Hij weet als geen ander de lezer te boeien als hij ingaat op de meer technische facetten van het hacken en de technieken die Max tijdens zijn notoire carrière gebruikte.

Het waargebeurde verhaal van Max 'Vision' Butler



Figuur 1: Kingpin cover

De jonge Butler

De jongen die geboren werd als Max Butler was eigenlijk een vrij normale tiener. Hij groeide op met computers in de computerzaak van zijn vader en had zich het programmeren in BASIC eigen gemaakt op zijn achtste. Maar toen zijn ouders scheidden brak er iets, en nadien schommelde zijn karakter tussen rustig en extreem geschift. Hij kwam al vroeg in zijn leven voor het eerst in aanraking met de sterke arm der wet, toen hij in zijn middelbare school inbrak om rotzooi te trappen. Zijn eerste misdaad was een kleintje, maar de impulsiviteit ervan zat inmiddels diep geworteld in zijn karakter en die impulsiviteit zou hem zijn leven lang blijven achtervolgen.

Nadat hij voor enkele andere vergrijpen een aantal jaren in de gevangenis had doorgebracht, trok hij in bij enkele oud klasgenoten en brak er voor hem een nieuw leven aan. Daar hoorde ook een nieuwe naam bij. Vanaf dat moment noemde hij zich Max Vision, naar een digitaal alter ego van zichzelf. Bij zijn nieuwe leven hoorde ook nieuw werk; hij werd geïntroduceerd bij de cybercrime-afdeling van de FBI waar hij gedreven aan het werk ging om de samenleving te beschermen tegen beroepscriminelen die het gemunt hadden op computersystemen.

Computercriminaliteit was in de tijd vóór Butler nog niet echt aan de orde. De eerste mensen die zich daadwerkelijk 'hacker' noemden waren studenten en onderzoekers aan het MIT in Amerika. Toen in de vroege jaren 80 van de vorige eeuw steeds meer mensen toegang kregen tot een eigen computer, met apparaten als de TRS-80 en de Commodore 64, veranderde dit. De eerste scriptkiddies begonnen het ARPANET, de voorloper van het huidige internet, af te tasten en gingen daarbij regelmatig over de schreef. In de dagbladen werden ze hackers genoemd, en daarmee kreeg het woord voor het eerst een kwade betekenis. Midden jaren '90 nam het thuisgebruik van computers en de toegang tot internet enorm toe. Recreatieve hackers waren er in overvloed, totdat er enkele grote namen in de hackerswereld, zoals Kevin Mitnick, werden gearresteerd. Opeens was het voor recreatieve hackers niet slim meer om te blijven hacken; het risico om gepakt te worden werd namelijk steeds groter en het plezier dat men in het hacken had woog niet op tegen dit vergrote risico. Sommige hackers gingen steeds dieper ondergronds en werden vaak echte criminelen. Het grote geld was nu te halen bij het beveiligen van computersystemen en dat was wat veel andere hackers gingen doen: ze ruilden hun criminele verleden in voor een rechtvaardige toekomst.

Shadowcrew

Max vond van zichzelf dat hij een white-hat-hacker was, tot op het moment dat de FBI voor een nieuw onderzoek van hem vroeg om een goede vriend en collega te verraden. Hij moest een gesprek met hem aanknopen waarin zijn vriend een bekentenis zou afleggenen dit opnemen om het later door te spelen aan de FBI. Max tekent het contract voor de zaak, maar houdt er een rotgevoel aan over. De volgende dag besluit hij zijn vriend in een café te ontmoeten, waar hij hem een briefje geeft. Max had besloten niet mee te werken aan het onderzoek en zijn vriend te waarschuwen. Zodra zijn begeleiders bij de FBI hier van horen, ontslaan ze hem als informant en Max werd niet lang daarna opnieuw ingerekend.

In 2001 werd het voornamelijk Russische forum CarderPlanet geopend. Op CarderPlanet konden criminelen handelen in onder andere creditcard gegevens. Al gauw werd het succes van deze site duidelijk en het duurde niet lang of iemand zette een forum op dat meer georiënteerd was op Europa en Amerika: Shadowcrew. Op Shadowcrew werden naast duizenden creditcard dumps ook identiteitsbewijzen verhandeld en het was al snel net zo'n succes als CarderPlanet. Ook Max had na zijn zoveelste vrijlating het forum ontdekt. Samen met een nieuwe partner dook hij in de illegale praktijk van het carding: het misbruiken van creditcards en andere betaalkaarten. Max hackte betaalsystemen van restaurants en andere bedrijven, haalde de database met creditcardgegevens leeg en leverde de dumps af aan zijn partner. Deze verhandelde de dumps op Shadowcrew. Het handeltje in dumps liep lekker, totdat bleek dat de oprichter van het forum door de US Secret Service was opgepakt en gedwongen werd mee te werken aan Operation Firewall. Tijdens Operation Firewall werd het forum veranderd in een honeypot om de andere leden te kunnen arresteren. Max had het naderende onheil al bespeurd en hield zich koest tijdens het oprollen van het forum.

Iceman et. al.

Na de ondergang van Shadowcrew bleven de duizenden leden gedesillusioneerd en ongeorganiseerd achter. Hun handeltje leek naar de maan, maar al gauw ontstonden er enkele nieuwe startups waar men hun criminele activiteiten kon voortzetten. Max had zijn nieuwe plan ook al klaarliggen: hij zou zelf een forum beginnen waar hij ongestoord zijn werk zou kunnen doen. Midden 2005 werd Cardersmarket.com geboren met aan het hoofd Iceman, Max' nieuwste troef. Iceman was zijn nieuwe alter ego dat het forum zou uitbaten. Onder andere schuilnamen kon hij de dumps van creditcards verkopen. Op deze manier werden zijn illegale praktijken niet gelinkt aan het feit dat hij de oprichter was van een forum, dat in principe legaal is. Dit zorgde ervoor dat Max tijdens de jaren na de oprichting van zijn forum ongestoord kon doorgaan met het verkopen van dumps. Tijdens zijn gehele notoire loopbaan stal Max de gegevens van meer dan een miljoen creditcards, resulterend in een verlies van vele miljoenen dollars bij verschillende banken.

Conclusie

Kingpin, Poulsens eerste boek, is een zeer informatief en tegelijk vermakelijk boek. Poulsen schrijft vlot en het boek leest goed weg. Het beschrijft niet alleen het verhaal van Max Butler, de white hat hacker die zijn vaardigheden besluit toe te passen voor minder legale zaken, maar gaat daarnaast ook dieper in op technische details die bij verschillende hacktechnieken om de hoek komen kijken. Ook komt er uitvoerig achtergrondinformatie aan bod omtrent de ontwikkelingen in computerveiligheid die speelden tijdens het leven van Max Butler. Kortom, een interessant en vermakelijk boek voor mensen die geïnteresseerd zijn in cybersecurity.

Bronnen

Wired
http://www.wired.com/threatlevel/2011/02/kingpin-excerpt/

Kingpin,
http://www.kingpin.cc

Kingpin (2011)
New York: Crown Publishing Group
Poulsen, K.L.

Bad programming habbits


Stijn van Winsen
Redacteur I/O Vivat

PROGRAMMEREN, JAVA, PHP, SLECHTE GEWOONTES

Wie heeft ze niet?

Wie heeft het nog niet meegemaakt? Een grote klasse vol met ongedocumenteerde code van je practicumpartner, en jij nu mag gaan uitzoeken hoe het werkt. Veel mensen hebben het: slechte programmeergewoontes. Wat zijn ze, en wat kan je ertegen doen? Het kan geen kwaad het een keer door te lezen en te zien dat jij dit natuurlijk niet doet.

Je plant niet voordat je gaat coden

Gewoon beginnen met coden werkt maar voor weinig mensen. Voor je ook maar begint met coden moet je een goed plan van aanpak hebben. Het helpt je niet alleen om op spoor te blijven, maar zorgt er ook voor dat je later niet in de war raakt van je eigen verspreide code. Laat staan iemand anders die je code moet bekijken. Een plan van te voren kan goed helpen om gestructureerd te programmeren. Een voorbeeld is het van te voren in comments opschrijven wat je gaat programmeren. Je weet van te voren al hoe het programma er uit komt te zien. De precieze implementatie is dan nog niet belangrijk, maar je hebt in ieder geval een idee over hoe je programma gaat werken. Comments zijn makkelijk te veranderen, dus een wijziging in je aanpak resulteert alleen maar in een wijziging van je comment. Verwacht ook dat je veel comments zal veranderen, verwijderen of toevoegen. Zie het als een boodschappenlijstje; het houdt je op de goede weg als je boodschappen gaat doen, maar het is geen definitieve lijst van producten die je uiteindelijk mee naar huis zal nemen.

Een plan kan goed helpen gestructureerd te programmeren

```
public void c(int v){
    if (v>3)
    if (v<8)
        System.out.println("3<v<8");
    else
        System.out.println("v>9");
    else if(v<3)
        System.out.println("v<3");
        System.out.println("Zo willen we het hebben");
    if (v==3)
        System.out.println("hier hoor je niet te omen");
}
```

Figuur 1: Voorbeeld van onduidelijke code door het niet gebruiken van accolades.

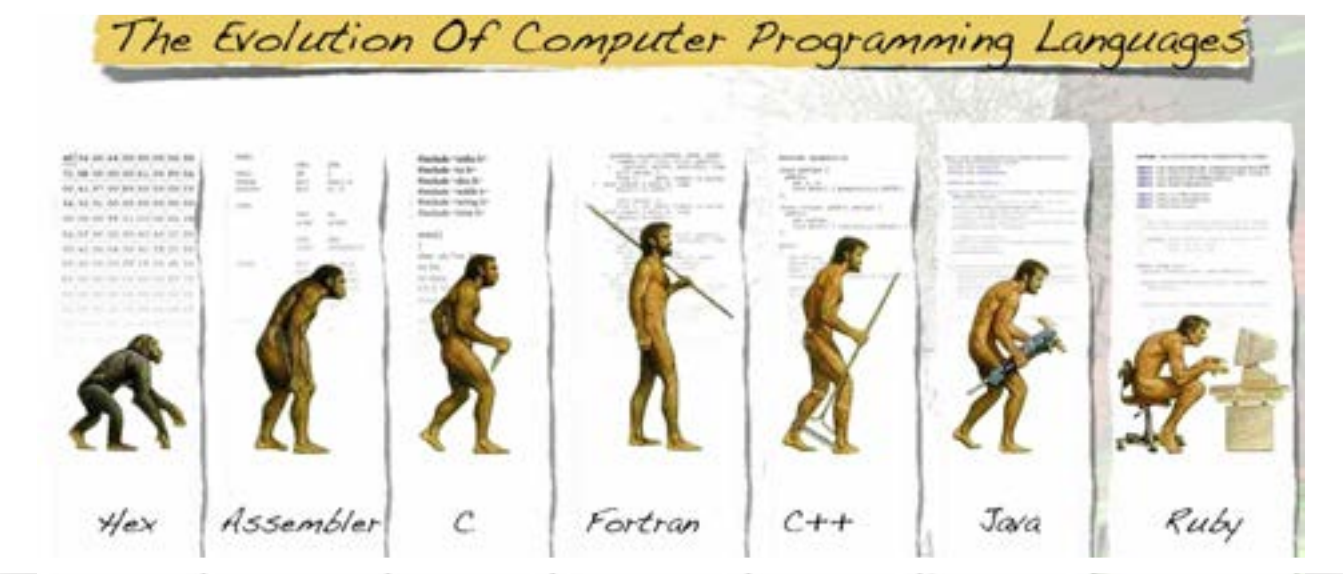
Eenvoud boven duidelijkheid

Veel mensen verkiezen compactere code boven duidelijkheid. Goede voorbeelden hiervan zijn het gebruik van onduidelijke variabelen als 'foo', 'AL' voor een ArrayList of gewoon het alfabet afgaan. Het is soms lastig om de shift knopt ingedrukt te houden voor een underscore of Camelcase, maar het kan zoveel duidelijker gemaakt worden. Een stijl die misschien kan helpen bij het benoemen van variabelen is een naam te laten beginnen met een letter die staat voor het type variabele. Bijvoorbeeld nFoo voor een integer en bFoo voor een boolean.

Ook het gebruik van accolades bij een if-else kan verwarrend zijn. Zie figuur 1. Natuurlijk snapt iedereen na even kijken dat "Zo willen we het hebben" altijd geprint wordt. Maar moeten we dat zelf uitvogelen, of is het makkelijker om gewoon wat accolades te gebruiken? Natuurlijk is zo min mogelijk regels gebruiken goed om je code beknopt te houden, maar dit mag niet ten koste gaan van de duidelijkheid van je programma.

Een standaard is voor mietjes

Iedereen wil anders zijn, een programmeerstijl hebben die jou van anderen onderscheidt. Programmeren is als een taal, grammatica en interpunctie bestaan met een reden; zo kan iedereen elkaar begrijpen als je dingen opschrijft. Kies dus een standaard en houdt je hieraan. Het volgen van een standaard zorgt ervoor dat mensen begrijpen wat je bedoelt, en niet dat je een saai persoon bent zonder eigen stijl. Daarnaast zorg je ervoor dat je over een paar jaar



je oude code ook nog kunt teruglezen en niet verdwaalt in je oude programmeerstijl. Er zijn genoeg standaarden, bijvoorbeeld de PEAR[1] standaard.

Patterns

Structuur is belangrijk als je een programma schrijft. Patterns als het MVC kunnen helpen ervoor te zorgen dat je weet waar componenten van je programma moeten komen. Maar niet alleen standaard patronen zijn handig, ook de manier waarop je dingen aanpakt kan via vaste patronen. Als je dit vaak genoeg doet, zorg je ervoor dat je zulk soort beslissingen automatisch gaat maken. Je programma is een stuk gestructureerder en je kan makkelijker later dingen terugvinden en veranderen als het nodig is. Ook helpt het om binnen een klasse standaard posities aan te houden voor methoden en variabelen. Een standaardvolgorde is bijvoorbeeld: variabelen, constructor en methoden.

Krijg je door dat je bepaalde stukken coden vaker aan het typen ben, maak er dan een aparte methode van als dit kan. Mocht je toch iets fout hebben gedaan, dan kan je gewoon één methode wijzigen in plaats van al deze stukken code op verschillende plekken. Bespaart je veel zoektijd.

Het nieuwe programmeren

Het is altijd leuk een nieuw trucje te gebruiken waar je laatst over gelezen hebt. Maar probeer geen complexe oplossingen te maken waar een simpele ook al werkt. Het kan er ook voor zorgen dat iemand anders je code niet meer kan lezen, omdat het te moeilijk is om te begrijpen.

Wat je echter nooit moet doen, is stoppen met leren. Als je stopt met het leren van nieuwe manieren van programmeren blijf je hangen in die tijd. De technologie gaat steeds weer vooruit en het is belangrijk dit te blijven volgen. De vooruitgang is er niet alleen maar om iets mooier te maken, maar ook om dingen efficiënter en makkelijker te maken. Ga dus geen complexe oplossingen gebruiken van anderen, maar kijk zeker wat er mogelijk is om toe te passen in je programma, waar dit het effectiever of duidelijker kan maken.

Houd het leuk

Het vinden van uitdagingen in een project is wat het programmeren zo leuk maakt. Bij elk project dat je doet, moet er iets zijn dat je uitdaagt. Zo houdt je het programmeren leuk en leerzaam. Vraag jezelf voor je begint aan een project eens af of er nieuwe technologieën bruikbaar zijn of dat je ondertussen nieuwe ideeën hebt om toe te passen. Met een andere bril naar een project kijken kan geen kwaad. Houd het programmeren leuk voor jezelf, en zorg voor genoeg uitdaging en afwisseling.

Eén voor allen, allen voor één

De meeste projecten kan je gewoon niet in je eentje doen. Probeer niet alles voor jezelf te houden maar spreek met je teamgenoten af wie wat doet. In een team moet je gewoon een plan hebben van hoe alles eruit komt te zien, anders wordt het één grote warboel. Al eerder gegeven tips zorgen er ook nog eens voor dat je snapt wat de ander schrijft. Ook kunnen je teamgenoten je voorzien van waardevol commentaar en andersom. Overleg dus goed over wie wat doet. Een goed project kan nou eenmaal niet van één persoon afkomen.

Comments

Één van de ergste slechte gewoontes is het niet documenteren van code. Het is zo simpel om te doen, maar toch zijn er veel mensen die het niet doen. Het maakt het voor iemand anders heel lastig om jouw code door te lezen, laat staan het ergens te verbeteren. Zelfs bij simpele stukjes code kan het enorm helpen te documenteren. Al kost het maar 3 seconden om een regeltje code te snappen, in een project met 100 regels code is dat al 5 minuten. En dat is als de regels simpel te begrijpen zijn! Vooral slechte comments zijn een ramp. Programmeurs die midden in een zin van Nederlands naar Engels overgaan of 10 regels aan comments nodig hebben om een methode nog niet goed uit te leg-

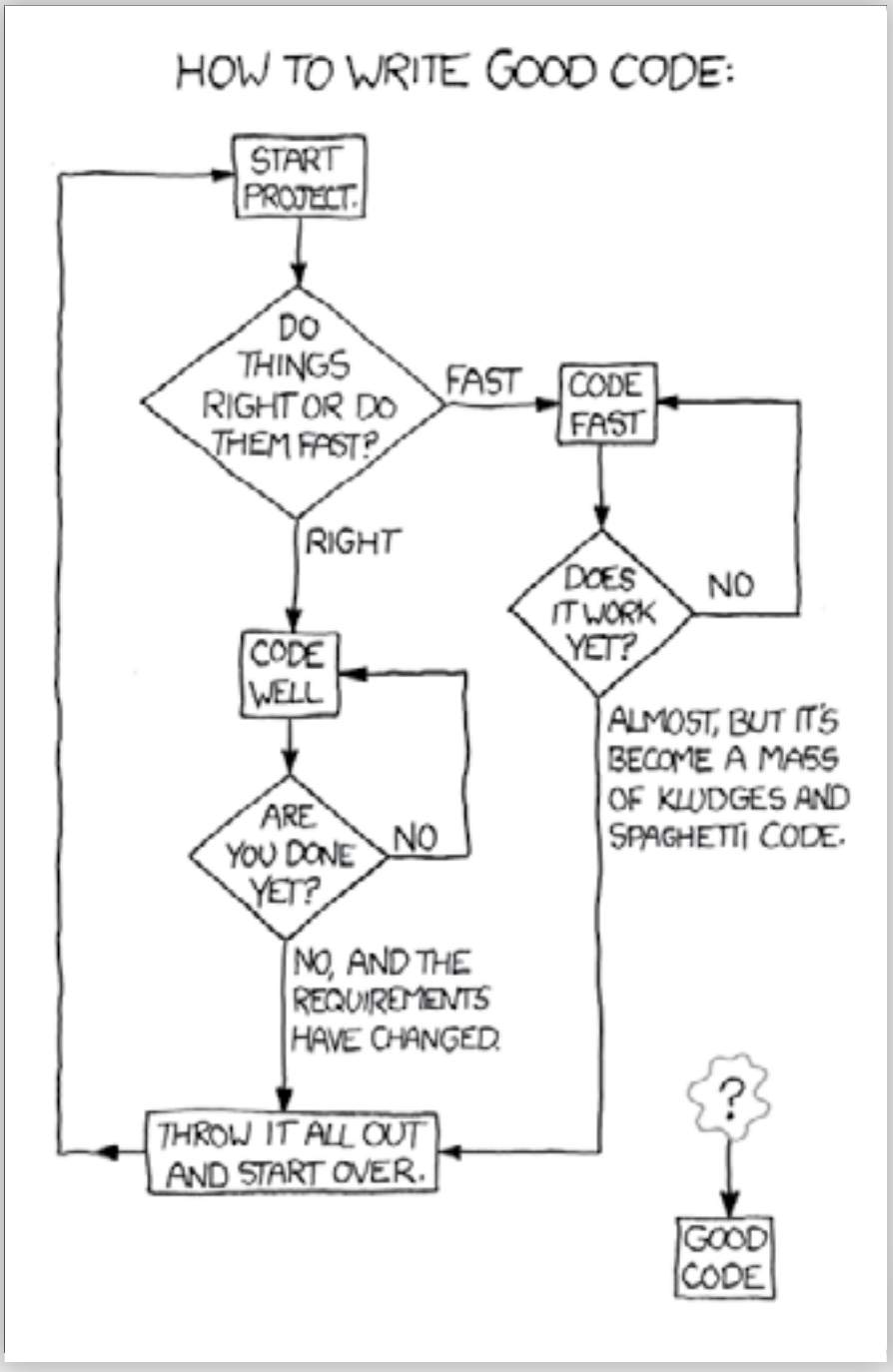
Zelfs bij simpele stukjes code kan het enorm helpen te documenteren

gen, maken het alleen maar erger. Natuurlijk moet je ook niet alles becommentariëren. Een handige vuistregel is, als je een paar seconde moet nadenken over je hoe je een probleem moet oplossen, is het handig een kleine uitleg toe te voegen voor iemand anders. Dus doe jezelf en anderen een plezier en lever commentaar bij je code.

Hoe moet het dan

Een belangrijk punt is dat je goed de requirements van het programma kent. Anders verspil je veel tijd aan een stuk code dat helemaal niet nodig is. Maak daarnaast al snel een eenvoudig systeem. Zo kan je met dat systeem kijken of het naar wens is van de doelgroep en alvast problemen uit je programma halen voordat je het programma af hebt. Dit komt het uiteindelijke programma alleen maar ten goede en je hebt meteen feedback op je werk.

Hoe goed je ook bent in programmeren, geen enkel programma kan perfect zijn. Hier moet je dus ook niet naar streven. Requirements veranderen met de tijd en dus moet ook je programma mee veranderen. Als je programma af is kan je verwachten dat je het nog vaak zult moeten aanpassen en verbeteren. Maar maak het programma natuurlijk wel zo goed mogelijk.



Figuur 2: Hoe goede code tot stand komt.

Een ander belangrijk punt van goed werken is van te voren plannen. Doe dit echter niet te veel. Er komt een punt waarop je meer aan het gokken en raden bent over hoe het eruit komt te zien dan dat je het daadwerkelijk plant. Het ontwerpen moet dus samengaan met het implementeren om elkaar van feedback te kunnen voorzien. Te veel van te voren plannen, en je ontwerp raakt te veel van de werkelijkheid. Te veel coden, en je programma wordt te onoverzichtelijk en ongestructureerd. Zoek de gulden middenweg die voor jezelf goed werkt.

Opslaan

Één van de belangrijkste tips tot nu toe: zorg dat je je werk vaak opslaat en backups maakt van grote projecten. Het overkomt de beste programmeurs dat de computer per ongeluk uitvalt en al je werk weg is en je weer opnieuw moet beginnen. Opslaan is heel gemakkelijk en bespaart een hoop werk als er toevallig iets mis gaat.

Dit geldt niet voor jou

Maar natuurlijk gelden al deze gewoontes niet voor jou en heb je niks aan deze tips. Je programmeert al zo lang dat jij in grote lijnen goed gedocumenteerd werkt. Maar houd er rekening mee dat slechte code vaak een ophoping is van meerdere kleine afkortingen en concessies, die door luiheid of haast even zijn overgeslagen. Kijk dus de volgende keer dat je aan je programma werkt even een keer van een afstandje naar je code, misschien valt je iets op.

Bronnen

[1] PEAR coding standard (2009)
http://pear.php.net/manual/en/standards.php

Advertentie Technolution

Micro vs. monolitische kernels

Michel Brinkhuis
Redacteur I/O Vivat

MICROKERNEL, MONOLITISCHE-KERNEL, LINUX, MINIX

Hét debat tussen de grondleggers van Linux en Minix

De basis van elk besturingssysteem is de kernel. Een kernel kan op meerdere manieren worden 'vormgegeven'. Iets wat zo'n twintig jaar geleden leidde tot een online discussie tussen de grondlegger van Linux: Linus Torvalds en de maker van Minix Andrew: Tanenbaum. Een discussie die wereldwijd bekend werd in de ICT-wereld als 'The Tanenbaum-Torvalds debate', en soms zelfs nog weer even oplaait omdat één van de betrokkenen erover naar buiten treedt.

Kort samengevat: in 1992 plaatste Tanenbaum in de comp.os.Minix-nieuwsgroep een post met de titel 'Linux is obsolete'. In de post pleitte Tanenbaum voor het gebruik van een microkernel gebaseerd systeem in plaats van de door Linus in Linux gebruikte monolitische kernel. Al snel volgde een discussie tussen Torvalds en Tanenbaum. Volgens sommigen een flamewar, maar ook aan inhoudelijke argumenten ontbrak het niet.

Kernels

Voordat we de discussie kunnen bekijken, is het handig als duidelijk is wat een microkernel anders maakt ten opzichte van een monolitische kernel. Allereerst, een kernel. De kernel heeft de verantwoordelijkheid over een aantal taken binnen het besturingssysteem. Het gaat hierbij om het management van interruptions, beheren van fysiek geheugen, beheren van virtueel geheugen en 'scheduling'. Dat laatste is het beheren van de processen, waarbij de kernel bepaalt wanneer welke taak uitgevoerd mag worden.

Verschil microkernels en monolitische kernels

Een microkernel is, zoals de naam al doet vermoeden, een kernel met een beperkte functionaliteit. Dit in tegenstelling tot de monolitische kernel, waarbij de kernel meer verworven is met overige delen van het systeem. Beperkt in de zin dat zoveel mogelijk zaken als apart proces zullen worden uitgevoerd binnen het systeem. Wanneer we kijken naar bijvoorbeeld drivers, dan zijn drivers in een systeem dat op een microkernel draait aparte processen. Bij een monolitische kernel vormen de drivers een geheel met de kernel. Je zult al wel vermoeden dat, wat drivers betreft, een stabieler systeem haalbaar is met een microkernel. Immers, wanneer een driver zou crashen is dat slechts een enkel proces dat crasht. Anders is het bij een monolitische kernel, waarbij een driver niet als zijnde apart proces kan crashen, maar in zijn crash andere systeemonderdelen ook kan mee laten crashen.

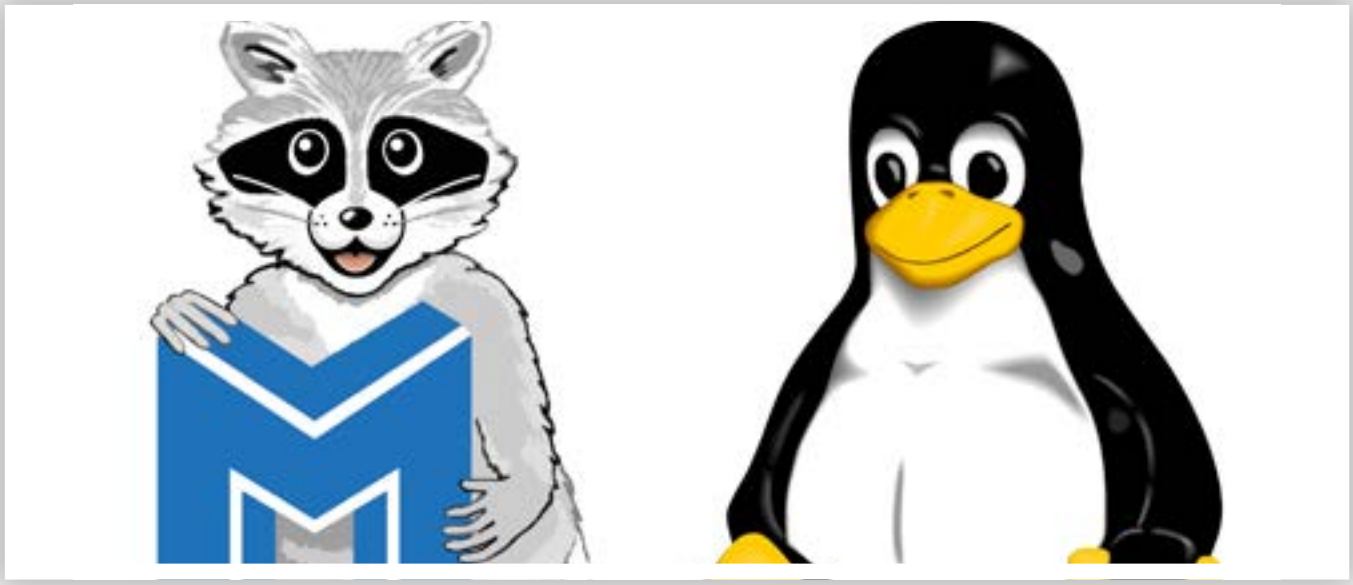
Velen zien tevens de grootte van de code als een van de voordelen van het gebruik van een microkernel. Doordat bij een microkernel minder taken door de kernel worden afgehandeld, maar plaatsvinden in de 'user space' zal de source code van een microkernel in theorie uit minder regels code bestaan. Dat komt de stabiliteit ten goede, zo stellen voorstanders. Immers, alle fouten halen uit een klein stuk code van een paar duizend regels is nog wel te doen. Maar als die broncode tien- of honderdduizenden regels code telt, dan wordt zoiets een heel wat lastigere opgave.

Bij een microkernel is het risico dat fouten optreden in de user space dan wel groter, omdat deze als logisch gevolg uit meer regels code zal bestaan dan bij een monolitische kernel. Echter, de user space bestaat uit vaak uit een aantal lagen, waarbinnen afzonderlijke processen draaien. Een fout in één van de processen heeft geen invloed op de kernel, die gewoon zal blijven draaien. De impact van een fout is dus in de meeste gevallen minder groot bij een microkernel, in tegenstelling tot bij een monolitische kernel.

Uit statistische onderzoeken is gebleken dat per 1000 regels code er zo'n vijf tot tien errors optreden. Bij code voor drivers ligt dit aantal iets hoger, het gaat hier per 1000 regels code om tussen de 35 en 70 errors. Vanuit dit opzicht bekeken is het dus zeker zinvol om zo min mogelijk code te hebben in het meest essentiële onderdeel van het systeem.

De stabiliteit van het systeem is één van de punten waar Andrew Tanenbaum in zijn discussie met Torval Linus op ingaat. Tanenbaum is tot op dit moment nog steeds bezig met onderzoek naar de

Microkernels zijn nog bij lange na niet 'mainstream'



microkernel, en werkt daarbij binnen de Vrije Universiteit in Amsterdam, aan een 'self healing' systeem. Hierbij kan een driver worden herstart, zonder dat een gebruiker het zelfs hoeft te merken. Doel van Tanenbaum om is middels deze technologie, welke verderop in dit artikel in nader detail zal worden besproken, een besturingssysteem te maken welke jaren achter elkaar zonder te crashen kan draaien.

Het is zinvol om zo min mogelijk code te hebben in de kernel

De discussie

Op woensdagmiddag 29 januari 1992 plaatst Andrew Tanenbaum in de nieuwsgroep comp.os.Minix de posting met als titel 'Linux is obsolete', waarin hij Linux op twee fronten aanvalt. Allereerst de discussie microkernel vs. Monolitische kernels. Tanenbaum stelt dat het enige argument vóór monolitische kernels snelheid is, en dat er in dat jaar genoeg bewijs is dat microkernels net zo snel kunnen zijn als systemen gebaseerd op een monolitische kernel. 'Microkernels have won.', zo luidt de kort maar krachtige conclusie van Tanenbaum. 'Linux is a monolitic style system. That is a giant step back in to 1970s.' Daarnaast gaat Tanenbaum in op de portabiliteit van Minix. Portabiliteit is hoe makkelijk de software is aan te passen zodat het ook op een ander type processor draait.

Nog dezelfde dag komt er in de nieuwsgroep een reactie van Torvalds. 'True, Linux is monolitic, and I agree that microkernels are nicer.' Volgens Torvalds is het echter zo dat de discussie niet zozeer uitsluitend uitsluitsel geeft over hoe goed een kernel is. Torvalds verwijst daarbij naar de Minix kernel, die alleen multithreading ondersteund wanneer er eerst hacks zijn toegepast. Tanenbaum ziet op dat moment het nut niet in van een multithreaded bestandssysteem. "When there is only one job active, the normal case on a small PC, it buys you nothing and adds complexity to the code. (…) It is only a win when there are multiple processes actually doing real disk I/O."

Ook wint Linux het op het punt van beschikbaarheid en ook is Linux meer portable dan Minix, vindt Torvalds. Weliswaar de kernel is niet portable, maar vanwege de open source-licentie en het feit dat de code vrijelijk beschikbaar is, is iedereen vrij de code aan te passen om Linux portable te maken. Volgens Linus kunnen programma's veel makkelijker worden geport naar Linux dan naar Minix.

Tanenbaum vindt het een verkeerde keuze dat Linux wordt gemaakt om te kunnen draaien op maar één type hardware, namelijk die van Intel. "An OS itself should be easily portable to new hardware platforms." Torvalds geeft Tanenbaum gelijk op het punt dat Minix portable is. 'But you can rewrite that as "doesn't use any features", and still be right.' Volgens Torvalds is de Linuxkernel maar een klein onderdeel van een heel system. 'And all of that source is portable.'

Echter, na een paar 'post-wisselingen' tussen Torvalds en Tanenbaum schrijft Torvalds: 'I over-reacted.', en geeft hij aan een wat minder 'bittere' brief te sturen naar Tanenbaum.

Daarmee leek de discussie op dat moment voorbij, maar dat de postwisseling in de nieuwsgroepen één van de bekendste discussies uit de OS-historie is geworden, bleek wel uit de jaren die erop volgden. Zo werd er onder meer een boek over de discussie gepubliceerd, en wakkerde Torvalds in 2006 de discussie weer aan met een post op het Real World Tech-forum. De discussie is wereldwijd zo bekend dat er zelfs een Wikipedia-pagina aan gewijd is genaamd 'The Tanenbaum-Torvalds Debate'`.

Tussenvorm

De discussie tussen Torvalds en Tanenbaum gaat over monolitische kernels versus microkernels. De 'kernelmarkt' is wel wat groter dan alleen deze twee varianten. Linux mag dan wel gebruikmaken van een monolitische kernel, bij Windows en Mac OS ligt dit anders. Beiden maken gebruik van een hybride kernel, een kernel waarbij zowel elementen van een monolitische kernel als van microkernels te herken-

nen zijn. Windows is 'hybride' sinds de komst van Windows NT. Microsofts' oorspronkelijke doel was Windows NT op een volwaardige microkernel te baseren. Maar doordat de prestaties die hiermee behaald konden waren niet voldeden aan de eisen van het bedrijf, is besloten om meer systeemservices van de user space naar de kernel te verplaatsen. Hierbij gaat het onder meer om de apparaatstuurprogramma's. Windows is vandaag de dag object georiënteerd van opzet. In dit ontwerp zijn processen, drivers, threads etc. weergegeven als objecten, welke allemaal beheerd worden door de 'object manager'.

Mac OS X maakt gebruik van de Mach-kernel, een microkernel ontwikkeld door de Carnegie Mellon universiteit. Ook hier zijn vanwege de prestaties drivers teruggebracht in de kernel, en is dus ook geen sprake meer van een volwaardige microkernel.

Uit onderzoek aan de TU Wenen is gebleken dat prestaties tegenwoordig geen argument meer hoeven te zijn om te kiezen voor iets anders een een microkernel. Dat brengt ons ook in de gelegenheid iets verder te kijken. Wanneer de microkernel absoluut niet meer 'bloated' (d.w.z. elementen bevat die ook in de user space zouden kunnen draaien) zou de microkernel zelfs in de processor geïntegreerd kunnen worden. Dat maakt nog betere prestaties, en de beste optimalisatie voor elke processor mogelijk voor elke processor, zo stelt onderzoeker Benjamin Roch van de TU Wenen.

Een 'self healing' kernel

Andrew Tanenbaum doet onderzoek naar het bouwen van betrouwbare en veilige besturingssystemen, en naar zijn idee zijn microkernels een goed gereedschap om dat doel te kunnen gebruiken. Het besturingssysteem Minix is inmiddels aangekomen bij versie 3, Minix3. Het besturingssysteem bestaat logischerwijs ook uit een microkernel, waarbij zo'n beetje alle functionaliteit draait in de 'user space'. Het besturingssysteem is te beschouwen als zijnde een systeem dat bestaat uit meerdere lagen.

De kernel is de basislaag, met daarboven een laag waarin de drivers draaien. Elke driver draait daarbij in z'n eigen proces, zodat dankzij een slechte driver niet de stabiliteit van het systeem als geheel in het geding kan komen. Boven de drivers-laag draait een laag met services. Hierin zitten een file-server (het deel van het besturingssysteem dat het wegschrijven en lezen van bestanden mogelijk maakt) en de zogenoemde 'reincarnation server'. Deze laatste zorgt voor de zelfhelende krachten van het systeem. Deze software roept periodiek alle lopende processen aan. Komt er geen reactie, dan wordt het proces vervangen door een nieuwe instantie van dat proces. Op deze manier zal een proces niet lang kunnen 'vastlopen', en hoeft het de gebruiker niet eens op te vallen dat een deel van het systeem is vastgelopen. Dat heeft ook zo z'n voordelen vanuit usability-perspectief bekeken. Stel je eens een omgeving voor waarin niets meer vast lijkt te lopen!

Bovenop deze service-laag is de applicatie-laag te vinden. Dit is de bovenste laag, en hierin draaien dan ook alle programma's. Programma's hebben geen directe toegang tot de kernel of de hardware. Datzelfde geldt feitelijk voor alle componenten van het systeem. Toegang tot de kernel wordt verkregen via communicatie met de onderliggende laag, via een 'top-down flow' naar de kernel.

Conclusie

Een discussie die zo'n twintig jaar geleden werd gevoerd, is eigenlijk tot op vandaag de dag nog niet beslist. Waar zowel Apple met Mac OS X als Microsoft met Windows wel hebben gekeken naar microkernels, maar daar omwille van prestatieoverwegingen toch niet volledig voor hebben gekozen, is Linux tot op heden volledig gebaseerd op een monolitische kernel. Microkernels zijn dus nog bij lange na niet 'mainstream' te noemen, maar dat wil niet zeggen dat het concept door de grote OS-bouwers helemaal aan de kant geschoven is. Een aantal jaar geleden startte Microsoft namelijk het onderzoeksproject Singularity. Een in C++ geschreven besturingssysteem welke gebruik maakt van een zuivere microkernel. Vooralsnog is het gebleven bij een onderzoeksproject, maar wie weet zal de opgedane kennis ooit ingezet worden om de kernel van Windows rigoureus op de schop te nemen. Maar of dat gezien de complexiteit van besturingssystemen tegenwoordig nog te verwachten valt, en op welke termijn, dat is lastig te zeggen. In die tussentijd is het echter niet geheel onwaarschijnlijk dat de 'Torvald-Tannenbaum debate' nogmaals zal opwaaien, door een publicatie of uitspraak van een van de kernelvisionairen.

Bronnen

Monolithic kernel vs. Microkernel
http://www.vmars.tuwien.ac.at/courses/akti12/journal/04ss/article_04ss_Roch.pdf

The H-Open: Andrew Tanenbaum's Minix 3
http://www.h-online.com/open/features/Andrew-Tanenbaum-s-Minix-3-746388.html

Tanenbaum-Torvalds Debate: Part II
http://www.cs.vu.nl/~ast/reliable-os/

Open Sources: Voices from the Open Source Revolution
http://oreilly.com/catalog/opensources/book/appa.html

Wikipedia: Kernel
http://nl.wikipedia.org/wiki/Kernel

Onderwijsvernieuwingen Scandinavië

Rom Langerak
Opleidingsdirecteur Informatica

Sinds april 1992 is dr. ir. Rom Langerak universitair docent bij de Formal Methods and Tools groep van de faculteit EWI. Romanus (Rom) werd op 1 februari geboren in Dordrecht en ging naar het Christelijk Lyceum aldaar. Hij haalde op de Universiteit Twente met lof zijn studie Toegepaste Wiskunde, waar hij afstudeerde op een onderwerp over Databases. Het is dan ook niet vreemd dat hij na zijn afstuderen ging promoveren bij de toenmalige faculteit Informatica. Na zijn promoveren in 1992 bleef hij bij de faculteit werkzaam.

Rom houdt van literatuur, filosofie, gitaar spelen, biljarten en Taekwondo. Sinds september 2009 is hij de nieuwe opleidingsdirecteur Informatica, een taak die hij met liefde zal gaan uitvoeren om zo het onderwijs voor zowel studenten als docenten

Na een wat onduidelijke start (toen de bachelors nog breed genoemd werden, terwijl later bleek dat ze nieuw waren) begint de vernieuwing van het bachelor onderwijs steeds duidelijker contouren te krijgen. Het basisidee van de vernieuwing wordt gevormd door het concept “module”: een module is een samenhangende eenheid van 15 EC, die je in zijn geheel moet volgen en halen. Dat lijkt een simpel idee, en dat is het ook, maar zoals zo vaak in het leven zijn het de kleine dingen die veel betekenen. Een belangrijk effect van de module is dat het de versnippering tegen gaat (het “sprokkelen” van vakjes), die er al snel toe leidt dat er vrijwel niemand meer nominaal studeert. Bij een module kun je niet besluiten dat je een onderdeel dat je wat minder bevalt, op de lange baan schuift: bij een module is het alles of niets! Een tweede belangrijk effect is dat modules leiden tot een grotere flexibiliteit. Je kunt vrijelijk met modules combineren, omdat een module per definitie een betekenisvolle eenheid is. Dat betekent dat je niet al als scholier in 5 of 6 vwo je hele leven hoeft vast te leggen. Als je gaat studeren, heb je in elk geval een initieel idee van wat je wilt, en daar kun je flexibel op blijven voortborduren door on-the-fly modules te kiezen op momenten dat je ook inderdaad de kennis en ervaring hebt om te weten wat zo’n keuze inhoudt.

Uiteraard laat het moduleconcept nog allerlei ruimte voor andere verbeteringen. En bij dat soort vernieuwingen is het een goed idee om inspiratie op te doen in het buitenland. Ik had het geluk deel uit te maken van een delegatie van 14 UT’ers die begin maart een vierdaagse rondreis door Scandinavië hebben gemaakt. Het was een gemengd gezelschap: de rector, onze decaan, een aantal beleidsmedewerkers, wat opleidingsdirecteuren, en twee studentes die de zaak perfect organiseerden. We gingen op bezoek bij de Universiteit van Aalborg, bij de DTU in Kopenhagen, en bij de Aalto Universiteit in Helsinki. We hebben er allerlei inspirerende ideeën opgedaan, met name op het gebied van projectonderwijs, wiskunde onderwijs, grote projecten, en ontwerpprojecten in de Master. En we kwamen er ook achter dat we in Twente best goed bezig zijn, dat ze op andere plekken soms met problemen kampen die wij allang achter de rug hebben, en dat er in de wereld ruimte is voor het Twentse model dat hier langzaam gestalte krijgt.

Zo’n tripje moet je niet zien als een toeristisch snoepreisje. Het is een hectisch gebeuren, het is een doorlopend gewissel van hotels, taxi’s en vliegtuigen, en er is geen enkele tijd om iets te zien. Van Aalborg hebben we helemaal niks gezien, en in Kopenhagen hebben we een wandeling (met twee biertjes) van zo’n honderdvijftig meter gemaakt. In Helsinki kwamen we om half twaalf aan in het hotel, en toen we volgens goed Nederlands gebruik nog een laatste biertje wilden, bleek de hotelbar al dicht. Gelukkig is onze decaan Ton Mouthaan iemand met veel initiatief, en hij had al gauw een soort rare Duitse biertent opgespoord, op steenworpafstand van het hotel. Erg gezellig leek het er niet (zelfs niet voor Finse begrippen), maar dat veranderde toen er een bandje het podium beklom. Accordeon, gitaar en bas, gespeeld door drie jongens die neefjes van de Leningrad Cowboys konden zijn. Ze speelden vrijwel alles wat we vroegen! Al gauw stond letterlijk het hele Nederlandse gezelschap op de dansvloer, het werd een onvergetelijke avond. Later bedacht ik dat die muzikanten misschien wel studenten waren, Finse langstudeerders, die onbedoeld het Twentse vernieuwingsproces de beslissende impuls hebben gegeven!

VerCors project

Marieke Huisman
Formal Methods and Tools

PROGRAMMAVERIFICATIE, SPECIFICATIE, CONCURRENCY, PROGRAMMALOGICA, SEPARATION LOGIC

Programma's met meerdere executiethreads, maar zonder fouten

Oktober vorig jaar heb ik een beurs van 1,3 miljoen euro ontvangen van de European Research Council - een instituut van de Europese Unie om toponderzoekers te financieren - voor het VerCors project, voluit: Verification of Concurrent Data Structures. Doel van VerCors is om de komende 5 jaar verificatietechnieken te ontwikkelen voor datastructuren die speciaal gemaakt zijn om in een concurrent programma te gebruiken. Met het geld van deze beurs kan ik een aantal mensen aan stellen (2 AiO's en 2 post docs) om dit onderzoek mee samen te doen. In dit artikel leg ik in het kort uit wat de achtergronden en de doelen zijn van het project en hoe we deze doelen gaan realiseren.

Verificatie van Multithreaded Programma's

De eerste vraag is natuurlijk waarom de verificatie van concurrent datastructuren, dat wil zeggen datastructuren die speciaal geschikt zijn voor multithreaded programma's, überhaupt een belangrijk onderwerp is. Het is een algemeen geaccepteerd gegeven dat het schrijven van correcte multithreaded software nog een graadje moeilijker is dan het schrijven van correcte sequentiële software. Voor een sequentieel programma hoef je immers alleen maar na te denken over hoe de verschillende instructies elkaar opvolgen en daardoor invloed op elkaar hebben. Voor een programma waarin meerdere executiethreads tegelijk bezig zijn, moet je als programmeur in gedachten bijhouden hoe al deze threads elkaar beïnvloeden. Je kunt er nooit van uit gaan dat een bepaalde instructie zal gebeuren voor een instructie in een ander thread - tenzij je deze volgorde expliciet afdwingt in je programma. Als je een beetje zoekt op internet kun je een groot aantal software bugs (met serieuze gevolgen) vinden die veroorzaakt werden door concurrency fouten. Een bekend voorbeeld hiervan zijn de problemen met de Therac 25. Dit was een bestralingsapparaat, waarvan de besturingssoftware multithreaded was. Door een fout in het programma konden soms twee threads tegelijkertijd schrijven op de geheugenlocatie waar de stralingsdosis werd opgeslagen, met als gevolg dat hier extreme waarden kwamen te staan en patienten een hoge overdosis straling ontvingen.

Als je zoekt op internet kun je een groot aantal bugs (met serieuze gevolgen) vinden

Door problemen zoals met de Therac 25 is er lange tijd gedacht dat concurrency te moeilijk was om correcte software te ontwikkelen. Als je persé wilde dat je software correct zou functioneren, dan moest je dat maar sequentieel doen; dan was het al moeilijk genoeg om te laten zien dat de software inderdaad correct was. Dit standpunt is echter niet meer vol te houden: de eisen aan software worden steeds hoger en langzamerhand bereiken we het einde van Moore's law: fysieke beperkingen zorgen er voor dat processorsnelheden niet meer elke 18 maanden verdubbeld kunnen worden. De oplossing hiervoor is het gebruik van chips met meerdere processoren. Om deze multi-core chips optimaal te gebruiken, moeten je programma's hier echter ook op ingericht zijn. Of te wel: het ontwikkelen van en nadenken over multithreaded software is niet langer te vermijden.

Het onderzoek binnen het VerCors project richt zich op het ontwikkelen van technieken om te kunnen zeggen of een programma zich correct gedraagt. Het idee is dat een programmeur een logische beschrijving geeft van het gedrag van het programma (dit wordt de formele specificatie genoemd) en dat vervolgens verificatietechnieken gebruikt worden om te controleren of de implementatie hier ook aan voldoet. Een methode die vaak voor dit soort problemen gebruikt wordt, is het gebruik van een model checker. Het basisprincipe van model checking is dat een volledige toestandsruimte van een systeem gegenereerd wordt en dat vervolgens gekeken wordt of alle bereikbare toestanden van het programma de gewenste eigenschappen hebben. Echter, in het geval van concurrent software stuit dit op veel problemen, omdat de toestandsruimte erg groot of zelfs oneindig is. Daarom gebruiken we binnen het VerCors project een andere methode, namelijk verificatie door middel van een programmalogica.

Geschiedenis van programmaverificatie

Het idee om programma's te verifiëren met behulp van logische bewijsregels stamt al uit de jaren 60. Robert Floyd en Tony Hoare ontwikkelden de zogenaamde Floyd-Hoare logica: een verzameling regels om over correctheid van een programma te redeneren. Ze introduceerden zogenaamde programmacorrectheids-triples `{P}S{Q}`, waarbij `P` een preconditie, `Q` een postconditie en S een programma is. Zo'n triple `{P}S{Q}` moet je als volgt lezen: als je programma `S` uitvoert in een toestand waar preconditie `P` waar is, en als programma `S` termineert, dan zal in de eindtoestand postconditie `Q` waar zijn. Vervolgens ontwikkelden ze bewijsregels waarmee je correctheids-triples voor samengestelde instructies kan opbreken in triples voor de onderdelen hiervan. Om bijvoorbeeld te bewijzen dat `{P} if C then S1 else S2 {Q}` geldt, is het voldoende om te bewijzen dat voor de beide branches `{P ^ C}S1{Q}`, respectievelijk `{P ^ ¬C} S2{Q}` geldt. Sinds het eind van de jaren 90 worden deze principes in verschillende tools toegepast om te redeneren over bijvoorbeeld sequentiële Java of C programma's. Waar de regels van Floyd en Hoare voor een simpele, ideale programmeertaal ontwikkeld zijn, zijn voor deze tools de logica's aangepast aan de de belangrijkste specifieke kenmerken van deze programmeertalen, zoals side-effects in expressies, excepties, en return statements.

Toen Floyd en Hoare deze bewijstechnieken introduceerden wilden mensen dit ook graag uitbreiden naar concurrent programma's. Midden jaren 70 kwamen Susan Owicki en David Gries met een uitbreiding van Floyd-Hoare logica voor concurrent programma's. Het belangrijkste onderdeel van hun verificatiemethode was dat je voor elke atomaire stap in je programma moest laten zien dat deze geen invloed had op de correctheid van andere threads. Dit resulteerde echter in zo'n grote hoeveelheid bewijsverplichtingen dat deze methode niet praktisch toepasbaar was.

Uiteindelijk is er pas een paar jaar geleden een methode gevonden om op een praktische manier over concurrent programma's te redeneren. Hiervoor wordt separation logic gebruikt. Dit is een uitbreiding van Floyd-Hoare logica die oorspronkelijk ontwikkeld was om te redeneren over programma's met pointers. Een belangrijk kenmerk van separation logic is dat de specificaties expliciet geheugenlocaties gebruiken en met name dat ze expliciet zeggen dat twee onderdelen van een programma op verschillende geheugenlocaties werken. Dit maakt het geschikt om op een praktische manier over concurrent programma's te redeneren: als je weet dat twee threads op verschillende geheugenlocaties werken, weet je automatisch dat ze elkaars correctheid niet beïnvloeden en hoef je dit niet meer te verifiëren.

Verificatie van multithreaded Java programma's

De afgelopen jaren heb ik (samen met Clément Hurlin en Christian Haack) een variant van separation logic ontwikkeld die speciaal geschikt is om over concurrent Java programma's te redeneren. Specifiek voor onze variant is dat we gebruik maken van zogenaamde permissies: we weten niet alleen of een thread een bepaalde geheugenlocatie gebruikt, maar ook of een thread op deze locatie mag schrijven, of alleen mag lezen. Correctheid van de bewijsregels garandeert dat een programma alleen geverifieerd kan worden met de bewijsregels als het geen data races bevat. Belangrijk is dat onze logica speciaal voor Java ontwikkeld is: we kunnen onder andere redeneren over het dynamisch creëren en beëindigen van threads, en over reentrant locks. Ook voor concurrent programma's geldt dat de logica aangepast moet zijn aan de specifieke kenmerken van de programmeertaal om over echte programma's te kunnen redeneren. Op dit moment zijn we druk bezig om tool support te ontwikkelen voor deze logica, zodat we het kunnen gebruiken om daadwerkelijk programma's te gaan specificeren en verifiëren – tot nu toe deden we dat altijd met pen en papier.

Binnen het VerCors project gaan we deze redeneertechnieken verder uitbreiden. De bedoeling is om uiteindelijk een verzameling gespecificeerde en correct bewezen implementaties van concurrent datastructuren te ontwikkelen. Deze datastructuren zijn dan de bouwstenen waaruit een concurrent programma opgebouwd kan worden en met behulp van onze verificatietool kan een software-ontwikkelaar dan makkelijk controleren of hij deze bouwstenen op een juiste manier gebruikt. Hiervoor moet ook de programmalogica verder uitgebreid worden: we moeten in staat zijn om te redeneren over andere concurrency- en synchronisatieconstructies, zoals bijvoorbeeld futures,

Als je meer wilt weten over het VerCors project

Op de pagina van het VerCors project (http://fmt.ewi.utwente.nl/research/projects/VerCors) vind je de volledige projectbeschrijving, een hoog-niveau presentatie van het project en alle technische resultaten.

In de NVTI nieuwsbrief van 2011 staat een samenvatting van de belangrijkste resultaten tot nu toe. De technische details en bewijzen zijn hier weggelaten. Dit artikel is beschikbaar via mijn homepage: http://wwwhome.ewi.utwente.nl/~marieke/papers.html.

Als je echt alles wilt begrijpen, dan is er ook een journal paper beschikbaar waarin alle technische details en bewijzen worden gegeven. In het Mastervak Program Verification worden de belangrijkste principes over het verifiëren van multithreaded programma's uitgelegd, onder andere aan de hand van dit artikel.

barriers, compare-and-swap instructies en reader-writer-locks. Daarnaast is het bedoeling om de verificatie van het concurrency-specifieke gedeelte los te koppelen van de verificatie van de functionele specificatie, dat wil zeggen de beschrijving van het gewenste gedrag. Op deze manier kun je laten zien dat als je de synchronisatie-strategie van een datastructuur wijzigt, door bijvoorbeeld meerdere locks te gebruiken voor onderdelen van de data, in plaats van één lock die alle data beschermd, dit wel invloed kan hebben op de performance, maar niet op de correctheid van een datastructuurimplementatie.

De komende vijf jaar verwachten we veel interessante onderzoeksvragen op te lossen

Om de methode ook praktisch bruikbaar te laten zijn, zullen we veel aandacht besteden aan automatisering. We zullen speciale algoritmes ontwikkelen om de bewijsverplichtingen te controleren en eventuele tegenvoorbeelden te genereren. Daarnaast zullen we ook technieken ontwikkelen om specificaties te genereren. De ervaring heeft geleerd dat om een programma daadwerkelijk te kunnen verifiëren er veel details expliciet gespecificeerd moeten worden. Maar veel van deze specificaties zijn standaard en kunnen automatisch gegenereerd worden, zodat de programmaontwikkelaar zich kan richten op het specificeren van de eigenschappen die essentieel zijn voor de correctheid van het programma.

Op de langere termijn is het de bedoeling om de volgende problemen aan te pakken: het uitbreiden van de logica voor de verificatie van zogenaamde lock-free programma's, waarbij soms wel een data race mag voorkomen, zolang deze maar "goedaardig" is; het uitbreiden van de logica naar gedistribueerde systemen, waarbij het geheugen van de verschillende sites consistent gehouden moet worden; en het uitbreiden van de logica naar andere programmeertalen.

Kortom, de komende vijf jaar verwachten we een heleboel interessante onderzoeksvragen op te lossen. Het lijkt niet reëel om te denken dat het resultaat over vijf jaar voor iedereen bruikbaar zal zijn, maar ik verwacht wel dat het een belangrijke stap hiernaar toe is. Een aantal jaar geleden dacht men ook dat verificatie van seqentiële programma's iets voor hobbyisten was, maar ondertussen zijn er commerciële tools beschikbaar die programmalogica-gebaseerde technieken gebruiken om fouten te vinden in programma's. Ik ben er van overtuigd dat over een aantal jaren zulke tools ook beschikbaar zijn voor concurrente programma's en dat de tools en technieken die we in VerCors gaan ontwikkelen, aan de basis hiervan zullen staan.

Advertentie
Thales

Werking/beveiligingOV-chipkaart

David Huistra
Redacteur I/O Vivat

TRANS LINK SYSTEMS, MYFARE CLASSIC, WERKING OV-CHIPKAART, HACKEN OV-CHIPKAART, INDELING OV-CHIPKAART, FRAUDEURS OPSPOREN.

Hoe werkt hij en waar komt alle trammelant vandaan?

We kunnen vandaag de dag bijna overal in Nederland met de OV-chipkaart reizen en de VVD wil dat de landelijke dekking van de OV-chipkaart nog dit jaar volledig is. Hoewel de OV-chipkaart over het algemeen wordt gezien als een goed streven, zijn er steeds meer partijen die zich afvragen of het systeem met de huidige kaart wel werkt. De media meldt dat het kinderlijk eenvoudig is om de OV-chipkaart te hacken en recentelijk hebben een aantal vervoersbedrijven openlijk aangegeven geen vertrouwen meer te hebben in Trans Link Systems, de uitgever van de kaart. Maar hoe kan de OV-chipkaart eigenlijk worden gehackt en is het inderdaad zo eenvoudig? Om deze vragen te beantwoorden zullen we eerst gaan kijken naar de werking van de OV-chipkaart.

De hardware van de kaart

De OV-chipkaart is in feite weinig meer dan een stuk geheugen dat door een kaartlezer kan worden uitgelezen en worden beschreven. Dit geheugen is opgedeeld in een aantal sectoren die elk apart worden beschermd door twee sleutels, genaamd A en B. Op de kaart kan worden ingesteld voor welke operaties op het geheugen welke sleutels nodig zijn, bijvoorbeeld A om te lezen en B om te schrijven.

De MIFARE Classic, de chip in de OV-chipkaart, beschikt over een geheugen van 4kB en is opgedeeld in 32 sectoren van 64 bytes en 8 sectoren van 256 bytes. Voor elke sector worden er 16 bytes gereserveerd voor de twee sleutels en de 16 allereerste bytes van de kaart worden gebruikt voor het serienummer. De OV-chipkaart maakt gebruik van een vaste indeling wat betreft welke informatie zich in welke sectoren bevindt en deze volledige indeling is op internet simpel op te zoeken. Zo blijkt je huidige saldo zich altijd in sector 39 te bevinden en staat de verjaardag van de eigenaar altijd in sector 22.

Op de kaart bevindt zich ook een kleine processor die voor de interactie van het geheugen met de buitenwereld zorgt. Deze proccessor ondersteunt een aantal instructies die het mogelijk maken om het geheugen van de kaart te lezen en te schrijven. De instructies zijn echter zo geprogrammeerd dat ze lees- en schrijfoperaties alleen uitvoert als de juiste sleutel wordt meegestuurd. Ook is het zo ingesteld dat sleutels zelf nooit kunnen worden uitgelezen en dat het serienummer van de kaart niet kan worden overschreven. Hiernaast zorgt de processor voor het versleutelen en ontcijferen van het dataverkeer met de kaartlezer, aangezien zonder encryptie het hacken zo simpel zou zijn als het afluisteren van het dataverkeer. De processor haalt de nodige stroom uit de inductie die aanwezig is bij kaartlezers en zal op deze manier dus nooit 'op' raken!

Voordat we aangeven hoe de OV-chipkaart kan worden gehackt, oftewel hoe de sleutels van sectoren kunnen worden achterhaald, zullen we eerst aangeven op welke manier er normale interactie met de kaart plaatsvindt en hoe de kaartlezer de sleutels normaal bepaalt. De werking van dit systeem is natuurlijk niet vrijgegeven, maar we kunnen het principe van dergelijke systemen wel aangeven.

Normale communicatie met de kaart

Als iemand bij een bus of op een station incheckt, zal de kaartlezer de juiste sleutels moeten weten om bij bijvoorbeeld je saldo te kunnen komen. De sleutels voor elke kaart in een tabel op elke kaartlezer opslaan is hierbij natuurlijk geen realistische optie, aangezien deze systemen dan bijgewerkt zouden moeten worden wanneer er nieuwe kaarten worden gemaakt, om het nog maar niet te hebben over de ruimte die nodig is om een dergelijke tabel op te slaan. Daarom worden vaak de sleutels op locatie bepaald.

Hierbij wordt normaal gesproken gebruik gemaakt van het serienummer van de kaart en een privésleutel die alleen bij Trans Link Systems bekend is. Over deze twee elementen wordt een hash berekend en deze hash wordt in een bepaald algoritme gestopt dat vervolgens de correcte sleutels voor elke sector kan opleveren.

Mensen thuis zullen niet de beschikking hebben over deze privésleutel of de werking van dit algoritme en kunnen daardoor niet simpel de sleutels berekenen. De reden dat er een hash wordt gebruikt komt vanwege het feit dat men wil voorkomen dat de werking van het algoritme en de privésleutel kunnen worden achterhaald door veel data te verzamelen. Toch blijken er genoeg mensen wel degelijk bij de informatie op de kaart te kunnen komen; hoe doen ze dat?

Het hacken van de kaart

Door gebruik te maken van de relatief kleine sleutels, het afluisteren van transacties en 'gewoon proberen' zijn er inmiddels algoritmes geschreven die binnen 300 queries naar de kaart elke sleutel kunnen achterhalen, wat resulteert op een sleutel achterhalen binnen een minuut. Op de werking van deze algoritmes zullen we hier niet ingaan, aangezien deze complex zijn en vaak wijzigen om sneller te worden. Deze algoritmes zijn echter gewoon opensource en door iedereen te downloaden en te gebruiken. Door herhaaldelijk gebruik van dit algoritme kunnen zo beide sleutels van alle sectoren worden achterhaald en kan vervolgens alle niet expliciet beveiligde data (zoals sleutels) worden gelezen en veranderd.

Elke sleutel binnen één minuut te achterhalen

Door een ontwerpfout van de OV-chipkaart is voor zowel het lezen als het wijzigen van het saldo van de kaart alleen de B sleutel van sector 39 nodig. Zo hoeft er zelfs maar één sleutel achterhaald te worden voordat het saldo dat op de kaart staat kan worden gewijzigd, wat dus binnen een minuut kan.

Zelf hacken

Voor het aanpassen van de informatie heb je een kaartlezer nodig, welke tegenwoordig vanaf 35 euro zijn aan te schaffen. Eenmaal in bezit van een dergelijke kaartlezer kan er een programma van internet worden gedownload dat al het werk voor je doet. Namelijk de sleutels bepalen en de gebruiker de mogelijkheid geven om de data te lezen en/of aan te passen.

Zo is er een programma genaamd 'OV Saldo – Lees / Schrijver' dat slechts over twee functionaliteiten beschikt, een lees knop en een schrijf knop, waarbij je zelf het saldo dat wordt weggeschreven kan bepalen.

Andere mogelijkheden zijn bijvoorbeeld het zelf inchecken in een bus of op een station. Met speciaal hiervoor gemaakte programma's kan je vanaf je computer met kaartlezer thuis al in de bus inchecken. Om het inchecken bij de bus te simuleren kan zelfs het geluid dat de officiele kaartlezer maakt van internet worden gedownload en bijvoorbeeld op een mobiel worden afgespeeld bij het binnenlopen van de bus.

De data op de kaart

De data die op de OV-chipkaart blijkt te staan is natuurlijk afhankelijk van wat soort kaart je hebt; zo heb je de persoonlijke kaarten en de anonieme kaarten. We zullen hier alleen naar de persoonlijke kaart kijken.

De informatie op de kaart is te verdelen in 3 categorieën, namelijk: identificatie, producten en reisgeschiedenis. Onder identificatie wordt het serienummer verstaan, maar ook de verloopdatum van de kaart, het bedrijf dat hem uitgegeven heeft en zelfs een banbit die aangeeft of de kaart verbannen is.

Aangezien niet alle sectoren van de OV-chipkaart worden gebruikt mogen bedrijven een van deze blocks claimen om hier hun eigen informatie zoals een abonnement in op te slaan. Van deze producten wordt dan vaak het productnummer en verloopdatum in deze sector opgeslagen. Bij de NS is er zelfs een derde parameter, namelijk of het product eerste of tweede klasse betreft.

Ten slotte wordt er ook nog reisinformatie op de kaart opgeslagen in de vorm van de laatste tien tot twaalf reizen die zijn gemaakt. Van elke reis wordt bepaalde informatie opgeslagen: het station van in- en uitchecken, het tijdstip hiervan en het bedrag dat hiervoor in rekening is gebracht. Maar waarom wordt deze informatie allemaal op de kaart gezet?

Naast het feit dat je hierdoor de reisgeschiedenis van de kaart bij de oplaadpunten op stations kan zien, is er nog een belangrijke andere reden. Voor het reizen met het openbaar vervoer moet er naast een bedrag voor het aantal gereisde kilometers ook een basistarief van 79 cent worden betaald. Bij een reis waarbij moet worden overgestapt hoeft dit echter maar eenmaal te worden betaald, zolang er altijd binnen 35 min na het uitchecken ergens anders word ingechecked.

De kaartlezers in bussen blijken echter

niet constant in contact te staan met de databases van TLS. Ze slaan alle informatie tijdelijk op en zullen slechts eenmaal per dag contact opnemen en alle infomatie doorgeven. Om er toch voor te zorgen dat iemand niet tweemaal een instaptarief hoeft te betalen, kan de kaartlezer via de reisgeschiedenis zien of de kaart minder dan 35 minuten geleden is uitgechecked, waarna hij niet opnieuw het basistarief voor reizen zal afschrijven.

Constante verschillen zullen gaan opvallen

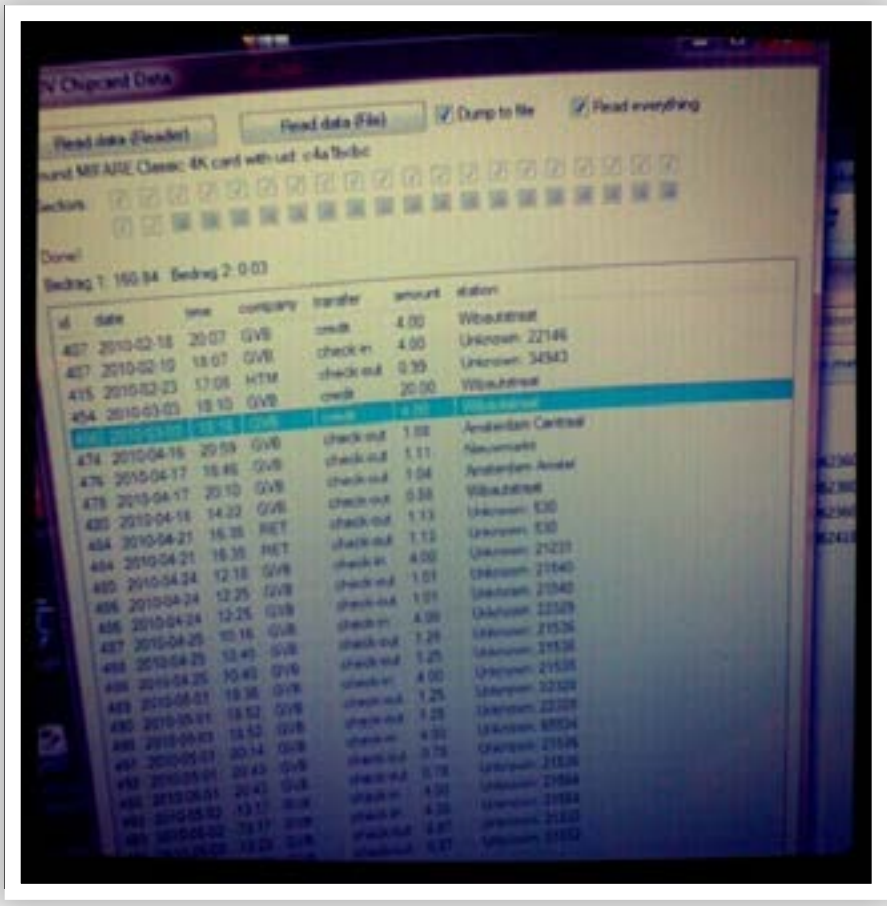
Hoewel er ook fouten aan de kant van de vervoersbedrijven kunnen plaatsvinden, bijvoorbeeld een bus die een dag de informatie niet opstuurt, zullen constante verschillen tussen de systemen van TLS en de kaart gaan opvallen. Als je in dit geval een persoonlijke OV-chipkaart hebt ben je makkelijk op te sporen. TLS zegt verder mensen met anonieme kaarten te kunnen opsporen door middel van de camera beelden die ze tot hun beschikking hebben.

Fraudeurs opsporen

Naast dat de informatie van passagiers op de OV-chipkaarten wordt opgeslagen heeft TLS zelf ook een systeem dat alle reisinformatie bijhoudt. Dit systeem synchroniseert eenmaal per dag met alle OV-chipkaartsystemen van de verschillende vervoerbedrijven en kan zo controleren of er inconsistenties bestaan tussen de data die de kaart doorgeeft en de informatie in hun eigen systeem.

Het was al snel duidelijk dat de OV-chipkaart te kraken was. De eerste keer gebeurde dit echter in een laboratoriumopstelling en gaf TLS aan dat dit niet haalbaar zou zijn voor een particulier. Inmiddels blijkt dat ook NPX, de maker van de OV-chipkaart, al in een vroeg stadium heeft aangegeven dat de beveiliging van de MIFARE Classic niet voldoende was en dat er een andere kaart van tien cent per kaart meer moest worden aangeschaft, maar dit advies is door TLS niet opgevolgd.

Het blijkt dat het hacken van de OV-chipkaart niet erg ingewikkeld is met de juiste apparatuur en software. Om echter consequent met een gehackte kaart te reizen lijkt lastiger te zijn, aangezien TLS beweert samen met de politie consequent mensen met gehackte kaarten op te sporen. TLS gaat er op het moment van uit dat een hoge pakkans mensen ervan moet weerhouden om de kaart te hacken. TLS is van plan de huidige kaart pas in 2015 te vervangen en hopelijk zal dat systeem lastiger te kraken te zijn.



Figuur 1: Dump van de inhoud van een OV-chipkaart

Bronnen

De OV-chipkaart wiki
http://OV-chipkaart.pc-active.nl

Topicus


Wouter Spoelstra
Topicus

SOFTWARE, ONTWIKKELOMGEVING, HERGEBRUIK, TOPICUS

Software hergebruik bij agile ontwikkelomgevingen

Copy-pasten, als student gebruik je het al en in het bedrijfsleven wordt het nog steeds veelvuldig toegepast. Ondanks dat iedereen altijd aangeeft dat dit natuurlijk niet de bedoeling is, is deze vorm van hergebruik vaak wel het snelste. Op dat moment natuurlijk, want fouten worden rustig mee gekopieerd. Een oplossing hiervan is het generieker definieren van een specifiek stuk code, zodat dit in een breder spectrum bruikbaar kan zijn. Echter, hoe weet je van te voren waar je het nog meer nodig gaat hebben?

Voordat deze vraag beantwoord kan worden, moet de code geanalyseerd, geoptimaliseerd en tenslotte geformaliseerd worden. Echter, met de komst van agile ontwikkeling zijn er weer nieuwe vragen ontstaan, aangezien de focus nu niet meer ligt op geformaliseerde processen en de documentatie hiervan, maar juist op mensen en communicatie. Hiermee wordt het interessant om uit te zoeken hoe dit proces nu het beste vorm kan krijgen en welke factoren hierbij een rol spelen.

Grote multinationals als IBM en HP hebben in het verleden gehele business units opgezet om software componenten te creëren en te onderhouden. Het lastige van deze opzet is dat kennis over de componenten verzameld moet worden bij andere business units, wat na verloop van tijd steeds moeilijker wordt, zeker aangezien het hergebruik vaak slechts een secundair proces is.

Naast deze organisatorische factoren zijn diverse andere factoren van invloed op software hergebruik. Om alle factoren goed te kunnen analyseren is er een model gedefinieerd, waarin onderscheid gemaakt wordt tussen een vijftal re-use levels, 15 re-use factoren en een assessment component. De re-use levels beschrijven de niveaus waarop software hergebruik plaats kan vinden, variërend van ad-hoc tot optimaliserend. De factoren beschrijven de verschillende invalshoeken en met behulp van de assessment component kunnen diverse verbeterpunten worden gedefinieerd.

Dit model is binnen verschillende business units van Topicus getoetst op correctheid en volledigheid. De cellenstructuur van Topicus, waarbij kleine units zich richten op specifieke marktsegmenten, leent zich uitstekend voor toepassing van dit model: naast de vele standaard componenten, worden er ook echte marktspecifieke componenten ontwikkeld, wat leidt tot een potentiëel hoog niveau van hergebruik.

Naar aanleiding van het onderzoek zijn diverse verbeterpunten gevonden. De belangrijkste is het toevoegen van een extra variabele om de schaalbaarheid van een re-use factor expliciet te meten, om zo de toepasbaarheid te vergroten. Daarnaast blijkt dat agile organisaties niet de processen als belangrijkste factor zien, maar juist de bewustwording bij het individu om hergebruik mogelijkheden te herkennen en te benutten. Het herkenningsproces begint bij het analyseren van de klantbehoefte en werkt door in de projecten. Ook bleek dat er behoefte is aan tool support om communicatie tussen projecten te verzorgen, echter zijn hier slechts beperkte mogelijkheden voor.

Wouter Spoelstra is student Bedrijfsinformatie technologie geweest aan de Universiteit Twente. Tijdens zijn afstudeeropdracht heeft hij in opdracht van Topicus onderzoek gedaan naar software hergebruik in agile ontwikkelomgevingen.

Topicus is een jong en innovatief ICT bedrijf met ca. 260 medewerkers in zowel Deventer als Enschede. Topicus realiseert identificieert zich niet alleen met hoogwaardige softwareoplossingen maar vooral ook door het automatiseren van gehele ketens in de sectoren zorg, onderwijs en finance centraal staan.

topicus

x86 versus ARM


Bas Stottelaar
Redacteur I/O Vivat

X86, ARM, INTEL, SOC

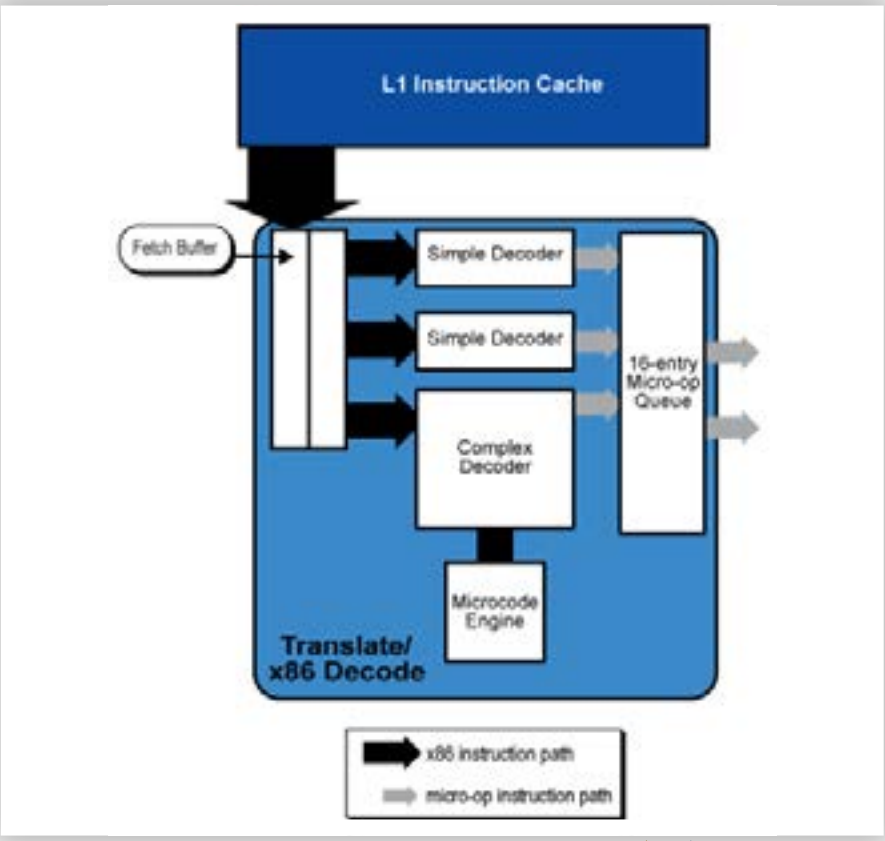
Een ware strijd of gewoon marktwerking?

Onder de ict'ers hoor je er haast niet meer bij als je geen smartphone(s) hebt. Even snel je mail checken, iets op het internet opzoeken of een spelletje spelen: allemaal taken die we 'vroeger' nog op de computer deden. Nu smartphones en tablets steeds krachtiger worden maar ook gelijktijdig energiezuiniger, lijkt er een strijd te zijn begonnen tussen de meest gebruikte architecturen voor het mobiele- en desktopsegment: x86 en ARM.

Toch hebben beide platformen twee verschillende mogelijkheden en doelen. Op je mobiele telefoon wil je geen programma's gaan compileren en je desktop wil je er niet elke keer bij pakken als je op internet wilt. Kortom, de markten zitten elkaar niet compleet in de weg, maar er is wel een verschuiving zichtbaar.

Fabrikanten spelen hier ook al op in. Intel probeert met haar Atom-processoren de ARM Cortex-gebaseerde processoren uit te dagen. Maar is er wel een strijd gaande? En wat is het voordeel van ARM boven x86? Dit artikel gaat daar op in.

Architectuur

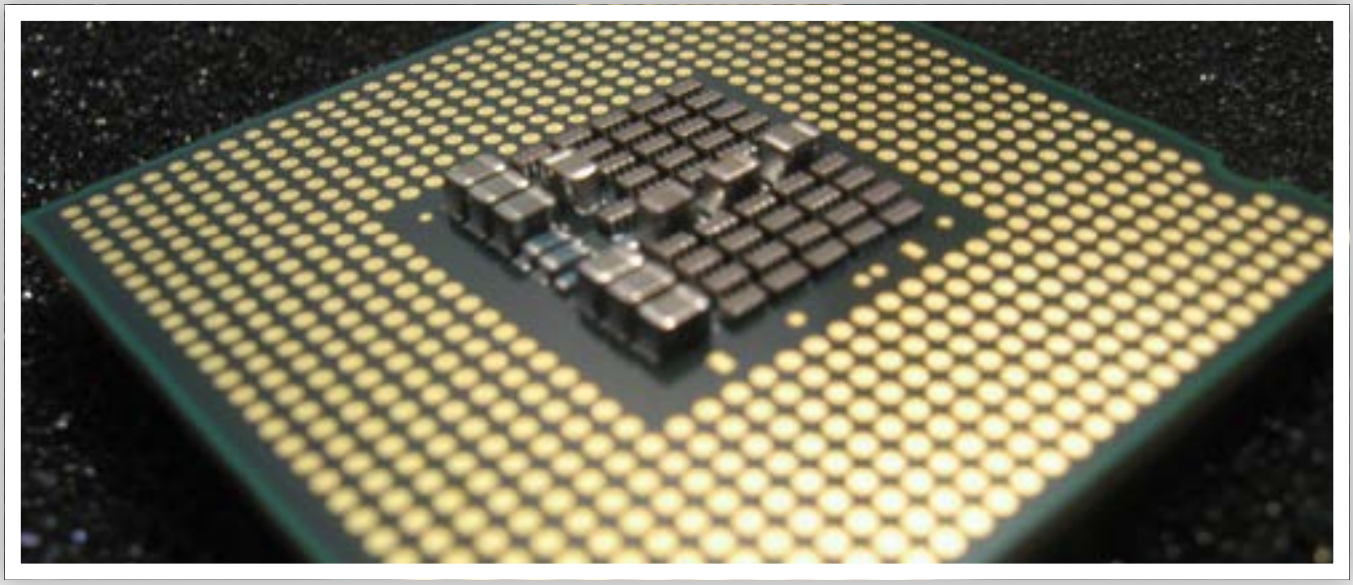


x86 is een complex instruction set computer (CISC). ARM is een reduced instruction set computer (RISC). De reden waarom hier voor gekozen is, heeft te maken met het doel van de inzet van vroeger. Embedded processoren zoals de ARM werden vroeger veel gebruikt op plekken waar de programma's eenvoudig waren en specifiek voor die ene toepassing. De processor hoeft dus geen rekening te houden met legacy code. Instructies kunnen dus direct uitgevoerd worden door de hardware, zonder een microcode en hardware om instructies te vertalen naar microcode.

Intel gebruikt wel een vertaler en microcode in haar processoren om instructies te vertalen. Dat heeft als groot voordeel dat een instructie hetzelfde kan blijven, maar de implementatie ervan op hardware-niveau anders kan zijn. Overigens gebruikt Intel sinds de 80486 ook concepten van een RISC-processor. Veelgebruikte eenvoudige instructies (zoals MOV) gaan dan niet door de vertaler, maar worden direct uitgevoerd. Het x86-platform is ook log omdat het juist legacy code ondersteunt. Hierdoor moeten 'oude' ontwerpkeuzes ook nog geïmplementeerd worden.

Dat legacy code niet per definitie draait op nieuwe ARMs is ook voordelig voor het aantal transistoren. De ARM2 (ARMv2) heeft er maar 30.000 en de ARM6 (ARMv3) maar 35.000. Bij Intel had de 8086 er alleen al 29.000 en de 80286 al meer dan 130.000!

ARM-instructies zijn altijd 32-bits. Dat heeft als nadeel dat een instructie die erg eenvoudig is toch meer ruimte nodig heeft. Er is wel een 16-bits vari-



ant (Thumb Mode), maar die beperkt het aantal instructies weer. Ook omdat ARM een RISC-processor is, heeft deze vaak meer instructies nodig om dezelfde berekening uit te voeren als een x86-processor. Hierdoor is een gecompileerd programma in de praktijk vaak tot 20% groter dan de x86-variant. Intel ondersteunt variabele instructielengtes waardoor codes compacter zijn.

Omdat de rekentaken sterk veranderd zijn de afgelopen jaren, zijn er ook specifiek voor die ene toepassing ontworpen worden waar het extra energiezuinig moet zijn, of een complete system-on-a-chip (SoC).

Intel staat geen licenties af voor x86-architectuur

ciale instructies aan zowel de x86-architectuur als de ARM-architectuur toegevoegd. Te denken valt bijvoorbeeld aan instructies om efficiënt met kommagetallen te rekenen, of met vectoren. Maar ook voor audio- en videobewerking zijn instructies toegevoegd. Virtualisatie wordt ook steeds vaker gebruikt op desktopsystemen. Om een gastsysteem te voorzien van fatsoenlijke prestaties hebben zowel Intel als AMD virtualisatie-instructies toegevoegd aan het x86 platform. De originele set met instructies is de 808. Deze is in de loop der jaren forst uitgebreid (SSE1-4, MMX, 3DNow, AMD64). ARM doet een soortgelijk iets en noemt dit ARMv1-v7, maar omdat ze geen compatibiliteit garanderen tussen verschillende versies, is het ook efficiënter.

Licentie

Er is een groot verschil in het recht op gebruik van de architecturen. ARM is open en geeft fabrikanten licenties om de architectuur te gebruiken. Dat betekent dat elke chipbakker zelf een processor gebaseerd op ARM mag ontwerpen, wat als groot voordeel heeft dat een fabrikant in staat is om een eigen draai aan de processor te geven. Hierdoor kan zo'n processor bijvoorbeeld

Intel staat geen licenties af voor het gebruik van de x86-architectuur. Toch zijn er een aantal andere fabrikanten die ook x86-processoren maken. Dat zijn AMD en Via die, vanwege oude overeenkomsten, dit wel mogen. AMD heeft geluk gehad door de 64-bit uitbreiding te ontwerpen voor de x86-architectuur, waardoor ze van Intel de licentie mochten houden.

Stel dat je als fabrikant een processor nodig hebt, dan kun je enkel een 'complete' x86-processor kopen van Intel, AMD of Via. Bij ARM koop je het ontwerp en produceer je zelf de processor. Omdat hierdoor veel ontwerpen beschikbaar zijn en je als fabrikant zelf zou kunnen produceren, zijn de ARM-processoren veel goedkoper dan x86-processoren. Het is daarom ook logisch dat er veel van deze processoren in mobiele apparaten terug te vinden zijn.

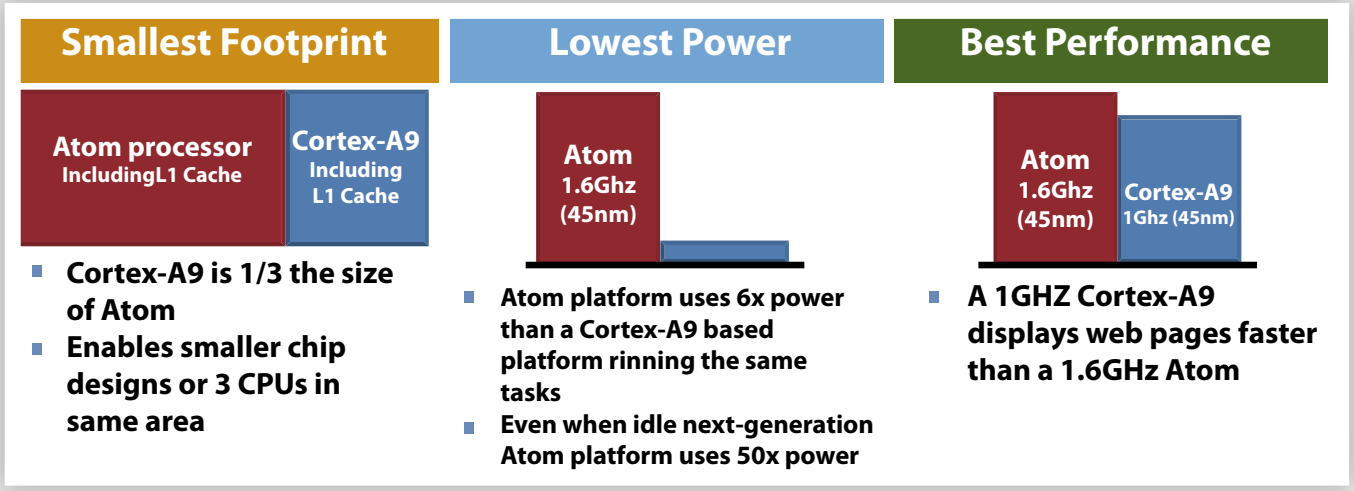
Benchmark

Omdat er veel verschillende uitvoeringen van de processoren op de markt zijn, is het niet mogelijk om een algemene benchmark te presenteren. Om toch een indruk te geven van wat soortgelijke processoren kunnen neerzetten, zijn er wel benchmarks beschikbaar van de huidige topmodellen die enigszins op elkaar lijken. Het gaat dan om de ARM Cortex A9 1Ghz processor en de Intel Atom 1,6Ghz.

Een ander groot voordeel van ARM is dat het goed mogelijk is om SoC te produceren. Marvell heeft eind 2010 een

Wist je dat:

- 5% van de Amerikanen een tablet heeft? In Groot Britannië is dit net iets meer dan 1,5%.
- 90% van de embedded-markt (router, telefoons, automotive etc) een ARM-processor bevat?
- Wereldwijd 1,7 miljard smartphones in gebruik zijn?
- Er totaal 4,7 miljard computers op aarde zijn?
- Het One-laptop-per-child-project ook een ARM variant heeft?
- Er ruim 15 Linux desktopdistributies zijn die op ARM processoren draaien?
- Apple ook ARM-architectuur in de iPhones en iPads gebruikt?



tripple core ARMv7 gebaseerde SoC geintroduceerd samen met USB3.0 ondersteuning, HDMI en een Direct X GPU die in staat is om 200 miljoen driehoeken per seconde te berekenen (MT/s). Ter vergelijking: een Playstation 3 is in staat om 250MT/s te berekenen en de iPhone 4 kan 28MT/s. Dit opent een nieuwe wereld voor multimediatoepassingen

In idle-state verbruikt een Intel Atom 100mW

Toekomst

De groei van het aantal mobiele apparaten wereldwijd is nog lang niet stil komen te staan. De komst van tablets doet de vraag naar energiezuinige processoren groeien met steeds meer geïntegreerde mogelijkheden. De kans dat je router of modem thuis met een ARM-processor uitgerust is, is tegenwoordig erg groot. Projecten om Linux-besturingssystemen te porten naar andere architecturen beginnen meer aanhang te krijgen.

Recentelijk heeft Microsoft aangekondigd om Windows 8 ook voor mobiele apparaten met een ARM-processor beschikbaar te maken. Er komen steeds meer mobiele apparaten bij, waaronder tablets. Energiezuinigheid is hierbij belangrijk en dat is iets wat door de ontwerpvrijheid bij een ARM-processor gemakkelijker is. In idle-state verbruikt een Intel Atom processor nog 100mW, terwijl een ARM processor rond de 1mW verbruikt.

Dat applicaties (en besturingssystemen) steeds vaker onder meerdere architecturen draaien brengt ook nadelen met zich mee. Programmeurs moeten op dit moment code nog vaak voorzien van extra patches om programma's te porten. Stuurprogramma's zijn vaak nog lastiger omdat deze erg afhankelijk zijn van low level architectuur. Verder heeft Intel al wel 64-bits processoren, maar ARM ziet er tot op heden niet de noodzaak van in.

Op technologisch vlak kan er ook nog het een en ander veranderen. Het probleem waar veel processoren mee te maken hebben is current leakage. Dat is een verschijnsel waarbij energie 'lekt' bij het schakelen van een transistor. Tegenwoordig zijn transistoren grofweg 32nm groot en is de energiebehoefte om een transistor te schakelen erg afgenomen. Echter is het zo dat de lekkage juist toeneemt en dat er meer energie verloren begint te raken door het lekken dan het schakelen. Meer transistoren betekent ook meer verlies. Het gebruik van hogere kwaliteit materiaal (waaronder high-k dielectric) voor de productie van transistoren moet deze lekkage voorkomen. Intel claimde recentelijk dat ze met Tri-Gate technologie transistoren kunnen produceren op 22nm die 50% minder energie nodig hebben.

Conclusie

Dat het aantal ARM-processoren aan het groeien is, moge duidelijk zijn. Benchmarks wijzen uit dat de ARM Cortex al erg dicht in de buurt komt van de Intel Atom. Het aantal transistoren in een ARM is veel minder dan in een Intel. Hierdoor is een ARM energiezuiniger.

Een Intel heeft altijd een basisinstructieset gehad, de 8086. Daarop zijn veel uitbreidingen gekomen. Dat heeft als voordeel dat legacy code ondersteund wordt, maar als groot nadeel dat de processor log is. ARM kent verschillende versies en heeft geen ondersteuning voor legacy code.

Vanwege de licentie is het als fabrikant 'voordeliger' om ARM te gebruiken en zelf een chip te bakken die specifiek ontworpen is voor dat ene apparaat, dan dat je een dure Intel-processor moet gebruiken. Maar gaat het over echte rekenkracht, dan heeft Intel nog steeds een streepje voor.

Er staat nog heel veel te wachten en de intrede van ARM is pas net begonnen. Intel heeft dit gemerkt en gooit de Atom-processor in de strijd. De strijd is nog maar net bezig en nog lang niet over. Wat in ieder geval zeker is, is dat de mobile era pas net begonnen is!

Bronnen

Achtergrond: ARM-processor technologie (2011)
http://nl.hardware.info/reviews/1994/achtergrond-arm-processor-technologie

Mirror: The Coming War: ARM versus x86 (2010)
http://vanshardware.com/2010/08/mirror-the-coming-war-arm-versus-x86/

ARM vs. Atom: The battle for the next digital frontier
http://www.infoworld.com/d/hardware/arm-vs-atom-battle-next-digital-frontier-762

Advertentie
QualityOnline

Leven op het web


Rick van Galen
Redacteur I/O Vivat

WEB, LIFEHACKING, JAVASCRIPT, CLOUD

Hebben wij desktopapplicaties nog wel nodig?

Desktopapplicaties zijn oud en versleten, zo lijkt het. Men loopt niet meer warm voor een nieuwe e-mailclient of RSS-lezer: alles gebeurt op het web! Maar zijn deze webdiensten al zo volwassen dat je geen desktopapplicaties meer nodig hebt?

Web 2.0 is inmiddels een enigszins gepasseerde term. Deze werd gehangen aan toepassingen op het web die het beter mogelijk maakte voor verschillende mensen, waar dan ook op de wereld, om te communiceren. Deze Web 2.0-toepassingen waren mogelijk door vernieuwde webstandaarden die het mogelijk maakten om in plaats van oude statische hypertext, dynamische applicaties te maken met dynamische inhoud - technologieen die vaak worden geschaard onder AJAX. Daarnaast hebben de grote adoptie van Firefox, Opera, Chrome en recentelijk Internet Explorer 9 er voor gezorgd dat veel webapplicaties gebruik kunnen maken van een voorschot op HTML5- en CSS3-specificaties. Deze nieuwe standaarden specificeren een aantal nieuwe technologieën om de functionaliteit en het uiterlijk van websites op hetzelfde niveau te krijgen als een desktopapplicatie.

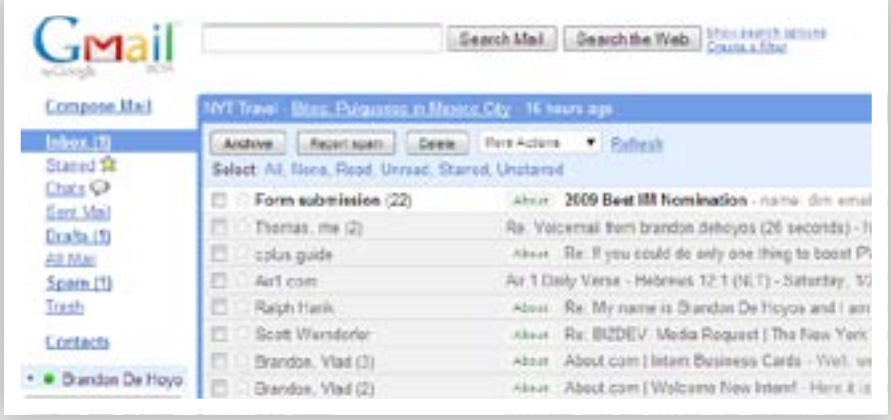
Door deze vooruitgangen in webtechnologie zijn er veel bedrijven ontstaan die zich hebben gespecialiseerd in het maken van webapplicaties die desktopapplicaties vervangen. Als de wijdverspreidheid van de benodigde technologie nog niet toereikend is, kan men applicaties schrijven in Flash, dat op bijna alle platformen van de eindgebruikers aanwezig is. Het voordeel van webapplicaties is de onafhankelijkheid van je werkplek. Als je een pc, tablet of telefoon hebt kun je simpelweg inloggen op je webdienst, en het werk dat je wil bereiken is beschikbaar vanaf de servers van de provider.

Er is zo'n groot aanbod aan desktopvervangende webapplicaties dat het wellicht mogelijk is om alle desktopapplicaties van de hand te doen en puur op het web verder te gaan. Dit idee wordt bijvoorbeeld interessant gebruikt door 'nieuwe' besturingssystemen Chrome OS en Joli OS: beide op Linux gebaseerde systemen die als portal dienen voor verscheidene webapplicaties. Voor dit artikel heb ik geprobeerd voor zover mogelijk met webapplicaties mijn zaken te doen. Hoewel ik een heel eind kwam, bleek dat ik als informaticastudent toch tegen beperkingen aanliep.

Mail en agenda

Er zijn op het web een groot aantal webapplicaties beschikbaar, maar er is er één die er met kop en schouders bovenuit steekt: Google Mail. Hoewel concurrerende platformen vandaag de dag een behoorlijk beter indruk maken dan een aantal jaar geleden, heeft Google bij de introductie van Gmail een standaard gezet waar andere webmailclients niet goed overheen komen. De dominantie van Google als web- en agendadienst zorgt bovendien voor uitstekende compatibiliteit met verschillende aantal andere diensten. Concurrenten als Windows Live Mail, Calendar van Microsoft en Calendar en Zoho Mail kunnen niet zowel de volwassenheid van de webinterface, het niveau van comptabiliteit

LucidChart is een echte aanrader



Figuur 1: Screenshot van Google Mail.

compatibiliteit, het spamfilter als de hoeveelheid beschikbare ruimte evenaren.

De Universiteit Twente biedt een oude Exchange 2003-server waar ik de afgelopen drie jaar gebruik van heb gemaakt. In de loop van de tijd ben ik groot fan geworden van Outlook als client, maar zodra deze niet beschikbaar was, werd ik aangewezen op een ernstig gedateerde webinterface. Mijn overstap naar Google Mail en Calendar was dan ook verademend. De webinterfaces van zowel de mail als de agenda zorgen ook voor een meer dan adequate vervanging van Outlook.

Eén ding dat Google nog niet voor elkaar heeft is takenmanagement. Gelukkig biedt de dienst Remember The Milk een handige plugin voor Google Calendar die het takenmanagement een stuk gebruiksvriendelijker maakt. Het werken met de combinatie van Google Mail, Calendar en Remember The Milk is een adequate vervanging van Outlook.

Notities

Notities op een desktop gaan niet veel verder dan het gebruiken van een texteditor, en de opgeslagen textbestanden ongestructureerd over verschillende mappen verspreid zijn. Een ware notitie-applicatie biedt veel meer opties om notities te organiseren en te synchroniseren. Evernote is een online dienst waar je notities bij kunt houden, kunt organiseren in verschillende mappen, en notities rijk kunt opmaken.

Documenten

Naast mail en agenda biedt Google ook Google Docs aan - een bekende dienst om online te tekstverwerken, spreadsheets bij te houden en presentaties te maken. Iedereen is vrij goed bekend met Google Docs - en daarom besloot ik gebruik te maken van Zoho Docs, de grote concurrent van Google Docs. Zoho Docs biedt ook een rijke interface om tekst op te maken, die toereikend moet zijn voor iedereen behalve de zwaarste Office-gebruikers. Ook biedt het uitstekend functionerende real-time

Figuur 2: Screenshot uit Zoho writer.

Platformkeuze

Hoewel het grote voordeel van het gebruik van webapplicaties is dat je overal je werk kunt openen, heb ik toch een paar aannames gemaakt voor het platform. De browser werd toegestaan om een Flash-plugin en PDF-plugin te hebben. Deze plugins zijn voor vrijwel alle platformen beschikbaar - alleen iOS-apparaten zijn beperkt daarin. Dit opende iets meer opties voor webapplicaties om te draaien, zoals SlideRocket en LucidChart.

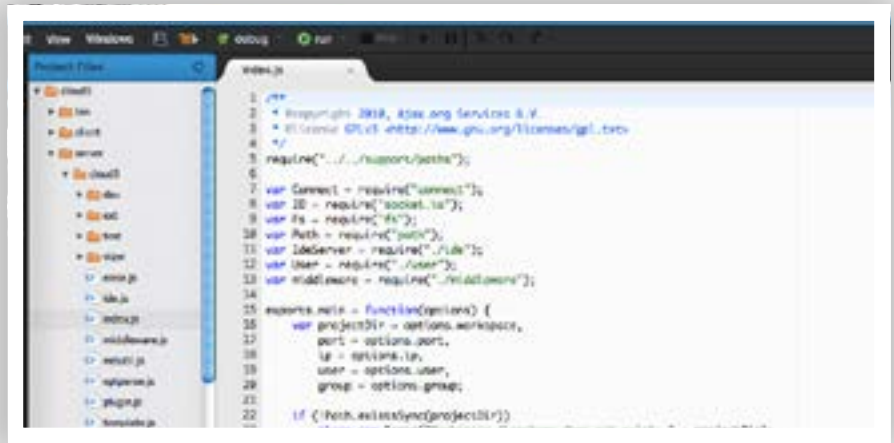
Als browser heb ik Google Chrome gebruikt. Niet alleen heeft Chrome standaard een PDF- en Flash-plugin, maar is het ook één van de snelste webbrowsers die beschikbaar is. Naast deze voordelen beschikt Chrome ook over uitgebreide synchronisatiefunctionaliteit, waarbij alle instellingen en extensies kunnen worden gesynchroniseerd. Al mijn extensies, wachtwoorden en dergelijke waren beschikbaar op mijn PC, laptop en bij Inter-Actief. Bovendien biedt Google met haar Chrome Web Store een gecentraliseerde plek om webapplicaties in je browser te integreren. Ondanks deze keuze voor Chrome werken de hier beschreven applicaties prima met Firefox en moderne Internet Explorers.

Webspelers in de cloud.

Op het moment van schrijven verschijnen in het streamen van muziek via het web een aantal grote partijen op het toneel. Amazon kondigde haar Cloud Player aan, Google haar Google Music Beta en Apple is bezig met een iTunes-streaming dienst. Al deze diensten komen overeen in de zin dat het de bedoeling is dat je er muziek naar uploadt en dat vervolgens via de webspeler of mobiele applicaties streamt. Het is dus niet als Grooveshark waarbij de muziek al aanwezig is: het is zelfs zo dat de online spelers een portal zijn naar de winkels van de respectievelijke bedrijven.

Eclipse in de cloud

Bestaande online IDE's zijn ontwikkeld als klein opensource project, of door een kleine startup. De Eclipse Foundation is nu echter ook begonnen met het maken webgebaseerde IDE, gebaseerd op in Java geschreven serversoftware gecombineerd met JavaScript client-code. Voorlopig is het het doel om een JavaScript-IDE op te leveren, en daarna door te ontwikkelen als completer platform. Misschien dat dit platform de eerste succesvolle webgebaseerde ontwikkelomgeving kan worden!



Figuur 3: Screenshot van Cloud9 IDE.

samenwerkingsmogelijkheden om tegelijkertijd met een aantal mensen aan hetzelfde document te werken.

Zoho Docs is beter dan Google Docs. De opmaakopties in de tekstverwerker zijn veel uitgebreider: lijsten, insprin-gen en andere opmaak zijn in Zoho Docs beter uitgewerkt met een makkelijkere interface. Ook zijn er opties voor meerdere kolommen, speciale tekens, webgebaseerde presentatie-applicaties mooie animaties tussen slides inzetten.

Het is in de informatica ongebruikelijk om academische papers te schrijven met een standaard office-pakket. Doorgaans gebruikt men hier LaTeX voor. Het is verrassend hoe goed de online LaTeX-editors zijn - aanraders zijn Verbosus en LaTeX Labs (die integreert met Google Docs). Helaas bleek geen enkele van de

Voor ontwikkeling moet men het voorlopig nog offline zoeken

betere ondersteuning voor vergelijkingen. Het importeren en exporteren naar Microsoft Office liep vanuit Zoho Docs ook vele malen beter dan vanuit Google Docs. Als laatste is de interface van Zoho Docs overzichtelijker, gedetailleerder en zet het meer de focus op het document dat getikt moet worden in plaats van menubalken. Voor de spreadsheet- en presentatietoepassingen van Zoho geldt ook dat deze veel uitgebreider en completer aandoen dan Google Docs, zonder in te boeten op de samenwerkingsfunctionaliteit.

Voor presentaties heb ik echter niet Zoho Docs gebruikt, maar de dienst SlideRocket. SlideRocket biedt een in Flash geschreven applicatie aan die een heel rijke interface biedt om presentaties te maken. Het voordeel dat SlideRocket heeft over andere webdiensten is dat het makkelijk Flashobjecten als video's kan implementeren, en widgets als Twitterstreams en polls in de presentatie zetten. Bovendien kan het als enige online LaTeX-editors in staat om zowel BibTeX als custom class-bestanden voor een andere opmaak te gebruiken. Omdat ik deze voor mijn werk wel nodig had, heb ik daarvoor een offline applicatie gebruikt.

Bewerken van afbeeldingen

Er zijn heel sterke afbeeldingsbewerkers. De firma Aviary heeft een aantal applicaties in gebruik om afbeeldingen mee te bewerken. Er zijn applicaties voor bitmap-creaties, vectorcreaties en fotobewerking. Hoewel geen enkele applicatie in de buurt komt van Adobe Creative Suite of Corel's producten, voldoet het prima voor enkele basisoperaties. Het grote probleem van het online bewerken van afbeeldingen is dat afbeeldingen altijd verplaatst moeten worden naar andere documenten. Of je nu een screenshot aan het bijwerken bent, de Vivat-cover aan het maken bent, of een website ontwerpt - de afbeeldingen zul je op de goede plek moeten krijgen. He-

laas betekent dat in de praktijk dat je je bestanden eerst moet uploaden naar de cloud om ze te bewerken, ze na bewerking weer downloaden vanuit de cloud en dit bestand weer moeten uploaden naar een andere clouddienst. Niet alleen is dit werk dat er buiten de browser moet gebeuren, maar is het ook nog eens erg onhandig.

Ontwikkeling

Als informaticastudent moet je aan ontwikkeling doen. De afgelopen tijd ontwikkelde ik in Java EE voor mijn werk, in Python/HTML voor het ontwerpproject en in Java/Promela voor mijn bacheloronderzoek. Het web is in potentie een uitstekende plek om je codewerk te doen: alle code veilig opgeslagen in de cloud, mogelijk Google-Docs-achtig tegelijkertijd code kunnen bewerken

Figuur 5: Screenshot van SlideRocket.

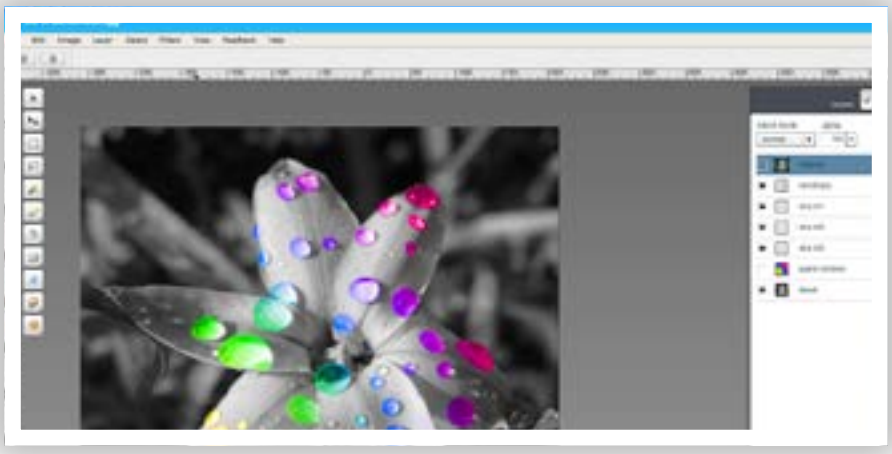
probeert dit wel te zijn, en heeft een omgeving die naast een code-omgeving ook een beperkte Platform-as-a-Service aanbiedt: het is mogelijk om PHP-programma's te draaien op de servers van Kodingen om te testen. Helaas is Kodingen nog in bètafase en functioneert het nog niet vlekkeloos. Het is een omgeving die voor webdevelopment in de toekomst nog veel kan betekenen, zeker als het bedrijf zijn beloftes om Git, Ruby en Python te ondersteunen ook gaat nakomen.

Je kunt binnen je webbrowser prima overleven

libraries die automatisch bugfixes kunnen krijgen, en het direct testen van webapplicaties in de browser zijn mogelijkheden die in de toekomst mogelijk moeten zijn.

De huidige online IDE's zijn echter niet geschikt hiervoor. Er wordt wel gewerkt hieraan. Cloud9 is een online IDE waarin PHP, Python, Ruby, HTML en JavaScript geschreven kan worden. Het is primair bedoeld als JavaScript-IDE, omdat het de browser kan gebruiken om dynamisch code te verifiëren. Verder biedt het dichte integratie met GitHub om code makkelijk te delen met anderen. Het is echter geen vervangen van een complete omgeving. Kodingen

Voor ontwikkeling moet men het voorlopig nog offline zoeken. Geen enkele webomgeving komt verder dan een beperkte deelverzameling van webframeworks, en kan daarmee dus maar beperkt in worden gezet voor webdevelopment en al helemaal niet voor ontwikkeling van andere zaken. Voorlopig is men daarvoor nog op NetBeans, Eclipse of Visual Studio aangewezen.



Figuur 4: Screenshot van Aviary.

Diagrammen

Een gedeelte van software-ontwikkeling kon ik gelukkig wel op het web doen. LucidChart is online diagramsoftware waarmee men klassendiagrammen, control flow-diagrammen, mockups en andere zaken kan maken. Het lijkt qua ontwerp meer op het Mac-programma OmniGraffle Pro dan Microsoft Visio. Een van de prettig verschillen met Visio is dat LucidChart niet voor je probeert na te denken: het weet dat je vormpjes en pijltjes wil tekenen en niet veel meer dan dat. Voor de meest complexe diagrammen is een betaald account nodig, maar voor de meeste diagrammen werkt het uitstekend. Een aanrader!

Muziek

Muziek in de cloud is iets dat de laatste paar jaar in een stroomversnelling is geraakt. Je muziek in de cloud opslaan en het overal luisteren waar je dat wil is een aantrekkelijke gedachte. Spotify is waarschijnlijk de populairste dienst, maar hoewel het een clouddienst is, is het geen webdienst (spotifyontheweb.com werd pas beschikbaar tijdens het schrijven van dit artikel) en viel dus af voor de test. Sociaal netwerk Last.FM biedt een betaalde dienst aan om onbeperkt muziek te streamen, maar biedt geen prettige interface aan om dit te ondersteunen. Mijn vereiste om een goede webspeler te hebben bracht mij bij Grooveshark: Grooveshark laat haar gebruikers muziek uploaden en dat met elkaar delen. Het bedrijf zorgt daarbij voor de licenties, in ruil voor het tonen van advertenties of de betaling van een maandelijks bedrag. Als je een maandelijkse bijdrage betaalt, kun je ook nog eens de nummers naar je telefoon streamen.

Grooveshark is een adequate vervanger van desktopspelers, met prettige faciliteiten om je afspeellijst te managen, het opzetten van 'radiokanalen' met bepaalde categorieën muziek (denk aan 'klassiek' of 'eighties dance'), en met integratie van sociale netwerken. Het grootste voordeel van Grooveshark is dat er veel meer muziek beschikbaar is dan je zelfs op je computer kunt opslaan. Van obscure Nederlandse muziek tot grote dancestampers uit de Top 40: alles is er te vinden, en mocht dat toch niet lukken kun je altijd je eigen muziek uploaden. Helaas heeft Grooveshark maar beperkte controle over de kwaliteit van de nummers die het aanbiedt, omdat deze door gebruikers geüpload worden. Daarom zijn sommige nummers van slechte kwaliteit of komt de tag-informatie toch niet overeen met het nummer dat je hoort.

Belangrijke nieuwe webtechnologieën Op een rijtje

De belangrijkste nieuwe technologieën die interactievere applicaties mogelijk maken:

Local storage

De traditionele manier voor browsers om data op te laten slaan is met cookies: kleine bestandjes die door de browser voor websites beschreven kunnen worden. Aan cookies kleven echter een aantal beveiligingsproblemen en tekstbestandjes zijn bovendien een primitieve manier om data op te slaan. Met Local Storage wordt een flexibele maar meer afgeschermde omgeving geïntroduceerd: websites kunnen lokaal bij de gebruiker een simpele relationele database beschrijven. Dit maakt het mogelijk om veel webapplicaties zonder internettoegang functioneel te houden.

WebSockets

Als een webpagina geladen is, kan deze met JavaScript verbinding maken met een server om data uit te wisselen (dit wordt vaak met de term AJAX aangeduid). Dit gaat altijd om eenvoudige berichtjes die heen en weer gestuurd worden, maar een webpagina heeft nog geen mogelijkheid om een permanente verbinding te openen naar de webserver. Met WebSockets wordt het wel mogelijk om een TCP-verbinding naar een webserver te openen. Dit maakt verbindingen met webservers sneller en betrouwbaarder, en maakt het beter mogelijk om websites sneller te laten reageren en nog dynamischer te maken.

WebWorkers

JavaScript wordt altijd in een enkele thread op een browser uitgevoerd. Aangezien webapplicaties steeds meer JavaScript gebruiken en ook zware berekeningen moeten uitvoeren, betekent dit dat een website kan 'hangen' als er zware operaties worden uitgevoerd, simpelweg omdat er geen JavaScript afgehandeld kan worden als deze op een berekening aan het werken is. Als een browser WebWorkers implementeert is het mogelijk om JavaScript 'multithreaded' te maken: JavaScript-operaties kunnen in een andere thread worden uitgevoerd, zodat zware berekeningen geen websites meer laten hangen.

Notifications

Een voordeel van desktop-applicaties is dat ze via popups aan kunnen geven wanneer er bijvoorbeeld nieuwe chatberichten binnenkomen of als bestanden geconverteerd zijn. Webbrowsers hebben dat voordeel normaal gesproken niet, maar Google Chrome implementeert sinds versie 9 een standaard om deze notificaties weer te geven. Mozilla heeft aangegeven deze op termijn ook te gaan implementeren, wanneer dit een W3C-standaard wordt.

Canvas

Het <canvas>-element in HTML5 maakt het mogelijk om een gedeelte in de browser vrij te betekenen. Dit maakt het mogelijk om animaties te maken die voorheen alleen mogelijk waren met Adobe Flash en Microsoft Silverlight. In I/O Vivat 26.2 staat een tutorial over hoe dit kan.

WebGL

Een uitbreiding op de <canvas>-tag om het beter mogelijk te maken 3D-beelden weer te geven is WebGL. Deze is gebaseerd op OpenGL, een 3D-bibliotheek voor allerlei platformen, en maakt gebruik van hardwareacceleratie op de platforms waar het op draait.

Deze technologieën worden goed ondersteund in Chrome en Firefox 4, maar het zal nog een tijd duren voordat dit gemeengoed is in alle webbrowsers.

Conclusie

Bijna twee weken lang heb ik alleen maar webapplicaties gebruikt, tenzij het niet anders kon. Voor de 'gewone' dingen ging dat best prima: met genoemde applicaties (en nog een aantal anderen) kun je de meeste de zaken die je gewoon op internet doet prima uitvoeren vanuit de webbrowser. Documenten bewerken, mails versturen, muziek luisteren: dat ging allemaal prima - en dat terwijl ik een grote hoeveelheid diensten niet eens heb uitgeprobeerd. Niet-algemene toepassingen zijn op het web echter nog niet te vinden. Met name software-ontwikkeling is nog een heikel punt, maar ook het bewerken van LaTeX-bestanden kwam nog niet helemaal goed uit de verf. De conclusie is dan ook dat je met alleen een webbrowser behoorlijk prima kunt overleven, als je er maar niet heel speciale dingen mee wil doen. In het algemeen viel het me mee hoe weinig ik office-applicaties, muziekspelers en e-mailprogramma's miste. Veel van de hier genoemde applicaties - met name LucidChart, SlideRocket en Zoho Docs - zal ik dan ook in de toekomst vaak blijven gebruiken.

Bronnen

Overzicht van nieuwe functionaliteit in HTML5, CSS3 en JavaScript (vooral voor Chrome)
html5rocks.com

Orion, het webgebaseerde IDE-project van de Eclipse Foundation
http://mmilinkov.wordpress.com/2011/01/11/introducing-orion/

LucidChart
lucidchart.com

Evernote
evernote.com

SlideRocket
sliderocket.com

Zoho Docs
docs.zoho.com

Cloud9
cloud9ide.com

Grooveshark
listen.grooveshark.com

Verbosus
verbosus.com

Chrome OS
google.com/chromebook/

Van de voorzitter

Jacco Roest
Voorzitter Inter-Actief

Jacco Roest werd geboren op 17 februari 1989 in Hardenberg. Hij is opgegroeid in Dedemsvaart, wat bekend is van de Dedemsvaria. De middelbare school rondde hij af in Hardenberg met als profiel N&T.

Hij begon zijn carrière in het onderwijs op basisschool ' De Regenboog' in Dedemsvaart. Zijn middelbareschooltijd beleefde hij op het Vechtdal College te Dedemsvaart en daarna in Hardenberg. Uiteindelijk heeft hij z'n VWO diploma gehaald in zes jaar met als profiel N&T.

Daarna de keuze voor Enschede, een gezellige stad, gezellige mensen en gezellige studievereniging. Hij is daar bezig met de studie Bedrijfsinformatietechnologie. En hij zet, na een jaar een Informaticus als voorzitter, de traditie voort als een voorzitter die de studie BIT doet.

Naast zijn studie heeft Jacco zich ingezet in verschillende commissies, waaronder de ECie, ICI (nu Kick-IT) en de CoLeX.

Het einde is in zicht

Het klinkt heel cliché, maar het voelt toch echt wel zo. Op het moment van schrijven zijn we al op zoek naar opvolging, op zoek naar leden die commissies willen doen die begin volgend collegejaar starten en dat terwijl je dat in mindere mate mee gaat maken, omdat je half oktober klaar bent met besturen. Iedereen leeft nu naar de zomervakantie toe en kijkt vooruit naar wat ze volgend jaar willen doen, kijken naar hun persoonlijke (studie-)planning. Zo moet ik zelf ook gaan kijken wat ik ga doen, qua vakken, maar ook in hoeverre ik iets actiefs wil doen en waar.

Je merkt dat het einde in zicht is, omdat we op het moment bezig zijn met het zoeken naar een kandidaat-bestuur, op zoek naar opvolgers die ook deze geweldige ervaring mee kunnen maken, mensen met een eigen visie en die hun eigen beleid gaan voeren. Je ziet dat dit bij de besturen om je heen ook gebeurd. Sommige besturen zijn net gewisseld, maar ook een deel van de besturen wisselt begin volgend collegejaar uit. Er zijn al een aantal kandidaat-besturen gepresenteerd, iets wat wij ook aan het eind van dit collegejaar willen.

Verder zijn wij op zoek naar mensen voor commissies die volgend jaar starten en die wij als contactpersonen van het bestuur eigenlijk niet meer zo veel zien. Dat zijn de kleinere commissies die een evenement neer zetten, zoals de skireiscommissie of een commissie voor het Twents kampioenschap programmeren. Maar ook voor de studiereis zijn we op het moment op zoek naar leden. Na de geweldige reis naar de Verenigde Staten van vorig jaar willen we graag begin collegejaar 2012/2013 weer een reis neerzetten.

En eigenlijk voelt dit vlak na de halfjaarlijkse algemene leden vergadering (ALV) een beetje vreemd. Je hebt deze vergadering aangegrepen om te laten zien aan de ALV hoe ver je bent gevorderd met betrekking tot het beleidsplan dat we aan het begin van het jaar gepresenteerd hebben. Tijdens de vergadering krijg je een stukje feedback over hoe je als bestuur het eerste half jaar door bent gekomen en waar wij nog beter op kunnen letten. Een belangrijk punt, ook met betrekking tot de kabinetsplannen waar ik in mijn vorige column over sprak, is het vervroegen van het wisselmoment van het bestuur. Het wisselmoment ligt op het moment op half oktober en wij zijn bezig om te inventariseren wat er allemaal bij komt kijken om het wisselmoment naar het begin van het collegejaar te halen.

Zoals gezegd, het einde is in zicht. Natuurlijk is dit jammer, want het bestuur is geweldig om te doen en ik ga dit met veel enthousiasme overdragen. En in de volgende Vivat hoop ik dat er een nieuw bestuur aan het roer staat bij deze geweldige vereniging en dat alle commissies gevuld zijn!

Inter-Actief

Op bezoek bij: Quinity

Voornaam Achternaam
Redacteur I/O Vivat

QUINITY,INHOUSE-DAG,VERZEKERINGSPRODUCTEN

Your Partner in eBusiness Solution Delivery

Wellicht heb je wel eens online een verzekering afgesloten. Grote kans dat je aanmelding werd verwerkt door een systeem van Quinity. Quinity is een e-business dienstverlener, gevestigd in Utrecht. Men houdt zich voornamelijk bezig met verzekeringsproducten. Daarvoor heeft het bedrijf een eigen systeem ontwikkeld, de Quinity Insurance Solution (afgekort: QIS).

QIS is een volledig geïntegreerde webbased polis- en schadeadministratiesysteem voor verzekeringsmaatschappijen. In Nederland is Quinity daar erg succesvol mee, met een marktaandeel in de verzekeringsbranche van zo'n 90%.

Op 19 mei organiseerde Quinity een inhousedag. Een hele dag werken volgens de werkwijze van Quinity, aan een case zoals ze bij Quinity aan de 'lopende band' voorbij komen: maak een functioneel ontwerp voor de inrichting van een verzekeringsproduct in QIS. Daarbij moet worden gekeken naar met name drie aspecten, namelijk: welke producten worden aangeboden? Hoe zitten die producten in elkaar? En hoe wordt de premie berekend? De basis voor een functioneel ontwerp is te vinden in een datamodel. Dit datamodel is een representatie van hoe de database van het systeem eruit dient te zien. De materie is behoorlijk complex.

Via een opdrachtbespreking en ontwerpsessies met de klant, een directeur van een fictieve verzekeringsmaatschappij, moesten de wensen van de klant in kaart worden gebracht. Met de informatie van de klant kon vervolgens een datamodel en een functioneel ontwerp worden opgesteld. Dit werd tussentijds een aantal malen met de klant besproken, waarna het eindresultaat gepresenteerd moest worden aan de klant.

De inhousedag is niet alleen een mooie gelegenheid om de werkwijze te zien binnen een deelproject, maar is ook een mooie gelegenheid om kennis te maken met Quinity als bedrijf. Quinity heeft zo'n 115 medewerkers in dienst, welke verspreid zitten over twee mooie (aangrenzende) panden. Dat maakt dat je bij Quinity binnen een kleinschalige organisatie werkt. Een jonge en dynamische organisatie, welteverstaan. Dat uit zich in veel aspecten. De gemiddelde leeftijd is 30 jaar en Quinity is een projectorganisatie. Opdrachten worden dus uitgevoerd in projectteams. De grootte van zo'n team kan erg verschillen, teams bestaan gemiddeld uit 5-8 mensen.. De samenstelling van een team wisselt vaak per opdracht, waardoor je dus met veel collega's samen zult werken. Iedereen kent elkaar, wel zo gezellig. Op kantoor heb je in principe geen vaste werkplek, iedereen werkt er op flexwerkplekken in zowel kleinere kantoortjes als in grote open ruimtes.

Maar er zijn meer leuke aspecten aan het werken bij Quinity, waaruit blijkt dat Quinity geen 'standaard' bedrijf is. Zo wordt er 's middags altijd samen aan grote tafels geluncht, is er een borrelcommissie die twee keer per maand een borrel organiseert, is er een feestcommissie en zijn er verschillende sportgroepen.

Quinity biedt haar werknemers ook veel opleidings- en doorgroeimogelijkheden. Wanneer je begint als consultant of software engineer volg je een uitgebreid opleidingsprogramma, en wordt je begeleid door een coach en vaktechnisch begeleider. Dat maakt het mogelijk dat je al snel kunt doorgroeien naar bijvoorbeeld de rol van senior consultant / software engineer, teamleider of projectleider.

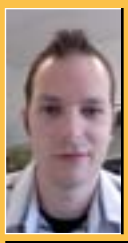
Wil je kennismaken met het bedrijf? Kijk dan eens op www.quinity.com en www.werkenbijquinity.nl. Daarnaast organiseert men regelmatig inhousedagen. Na je studie kun je bij Quinity aan de slag als software engineer of consultant. Ook zijn er regelmatig interessante afstudeeropdrachten beschikbaarvoor laatstejaars informaticastudenten. Interesse? Dan kun je je CV en cijferlijsten voorzien van een korte motivatie mailen naar jobs@quinity.com. Voor meer informatie kun je bellen met Tessa van Rijnsoever of Gerda Kamphof, beiden zijn bereikbaar via het telefoonnummer 030-2335999.

ENIAC

CλaSH

Column: Van de voorzitter

CλaSH

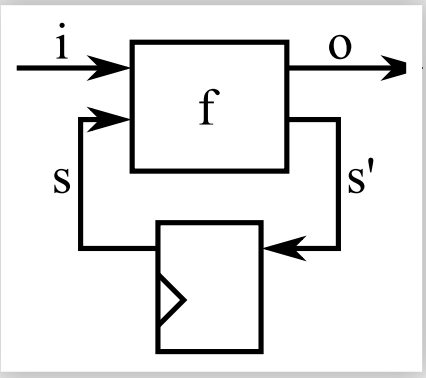


ChristiaanBaaij & Matthijs Kooijman
ENIAC

CλASH, HASKELL, CAES, HARDWAREBE-SCHRIJVING

Functioneel Hardware ontwerpen in CλaSH

De technologie achter het fabriceren van microchips is de afgelopen decennia met veel grotere stappen vooruit gegaan dan de methodologieën en talen voor het ontwerpen van hardware. Het gevolg hiervan is dat we nu meer transistoren op een chip kunnen zetten dan dat we nuttig kunnen gebruiken. Om aan te geven wat de huidige staat van hardware-ontwerptalen is trekken we een vergelijking met programmeertalen: VHDL en Verilog, de standaard hardware ontwerptalen zijn vergelijkbaar met high-level assembly talen. Net als in een high-level assembly taal zijn er wel wat trucjes om bepaalde dingen sneller op te schrijven, maar moet er nog steeds op een heel laag en heel gedetailleerd niveau ontworpen worden. Net als met programmeren in assembly zijn complexe ontwerpen niet meer behapbaar voor de hardware ontwerper omdat hij geen beschikking heeft over abstractiemechanismen. In de software-wereld worden inmiddels hogere talen (voornamelijk objectgeoriënteerde talen) gebruikt om de beschikbare CPU kracht optimaal te benutten, maar in de hardware wereld zijn VHDL en Verilog nog steeds de meest gebruikte talen.



Figuur 1: Mealy Machine

Een functionele taal, [...], is daarom heel geschikt als ontwerptaal voor hardware [...]

Hardware en Functionele talen

Om de abstractie-mechanismen die softwareprogrammeurs al tijden gewend zijn nu ook aan te bieden aan hardware ontwerpers is er binnen de CAES[1] vakgroep van de Universiteit Twente de hardwarebeschrijvingstaal CλaSH[2] ontwikkeld. Deze ontwerptaal is gebaseerd op de functionele programmeertaal Haskell[3]. Er is gekozen voor een functionele taal omdat hardware en functies hetzelfde model volgen. Functioneel programmeren is een paradigma dat het 'berekenen' ziet als de evaluatie van een wiskundige functie. Het legt de nadruk op het toepassen van functies, in tegenstelling tot de imperatieve en objectgeoriënteerde programmeerstijlen (Java, C/C++) die de nadruk leggen op het veranderen van de toestand (geheugen) van de onderliggende hardware.

Combinatorische schakelingen (hardware zonder geheugen) zijn eenvoudig wiskundig (als functies) te beschrijven. Net zoals de toepassing van een functie op een bepaald argument altijd hetzelfde resultaat geeft, levert een schakeling bij een bepaalde invoer ook altijd dezelfde uitvoer.

Een functionele taal, waar de nadruk dus ligt op functies en waar het toepassen van functies de basisoperatie is, is daarom heel geschikt als ontwerptaal voor hardware omdat het evaluatiemodel van hardware en functioneel programmeren erg op elkaar lijken. Functionele talen hebben als bijkomend voordeel dat ze beschikken over vele abstractiemechanismen die het mogelijk maken om zeer generieke ontwerpen te maken die aan de hand van allerlei parameters aanpasbaar zijn.

Een voorbeeld van zo'n abstractiemechanisme is dat functionele talen functies zelf als "first-class citizens" van de taal beschouwen. Dat betekent dat functies zelf zowel het argument als het resultaat van een functie kunnen zijn (deze laatste functie noemen we dan "hogere orde"). In het voorbeeld verderop zullen we gebruik maken van dit zeer handige abstractie-mechanisme. Merk op dat een functie als first-class waarde in de uiteindelijke hardware natuurlijk niet mogelijk is, dus de CλaSH compiler zoekt voor je uit welke functies precies waar gebruikt worden, waardoor er toch hardware gemaakt kan worden van een beschrijving met hogere orde functies.

Hardware en Geheugen

Ondanks dat het evaluatiemodel van

functionele talen en combinatorische schakelingen perfect bij elkaar aansluiten, bevat de meeste (interessante) hardware echter ook geheugenelementen. Het geheugen zal vaak ook het resultaat van de hardware beïnvloeden, waardoor de uitvoer bij een bepaalde invoer niet meer altijd hetzelfde zal zijn. Om met het concept 'geheugen' om te gaan binnen CλaSH hebben we teruggegrepen naar een oud en zeer bekend model van hardware: de Mealy machine. Een schematische weergave van de Mealy machine is te zien in Figuur 1.

De Mealy-machine bestaat uit een combinatorische schakeling, f, die twee invoerwaarden heeft: de externe input, i, en de huidige waarde van het geheugen, s. Het resultaat van deze Mealy machine bestaat uit twee delen: de output, o, en de nieuwe waarde, s', die in het geheugen geschreven moet worden. We beschrijven een functie f in CλaSH zoals te zien is in Code Listing 1. Deze voorbeeld functie geeft van een rij invoergetallen steeds de som van het huidige getal en het vorige getal. Merk op dat CλaSH code ook gewoon geldige Haskell code is. Dus hoewel we geen ruimte hebben om de syntax van CλaSH uitgebreid te behandelen, kun je je Haskell kennis direct toepassen (of van de gelegenheid gebruik maken om eens naar Haskell te kijken, dat is zeker de moeite waard!).

Hierbij geeft het State keyword aan welk argument en welk deel van het resultaattupel met het geheugenelement zijn verbonden. In CλaSH wordt alle hardware die geheugen bevat dus in de vorm, zoals in Code Listing 1 getoond, op geschreven. De CλaSH compiler zal er dan voor zorgen dat de geheugenelementen in de uiteindelijke hardware worden gezet. Via dit Mealy-machine model hoeven we dus niet af te wijken van het evaluatie-model van functioneel programmeren en blijft het beschrijven van de hardware dus consistent met de achterliggende theorie.

```
f(State s) i=(State s', o)
  where
    (s',o) = (i, s + i)
```

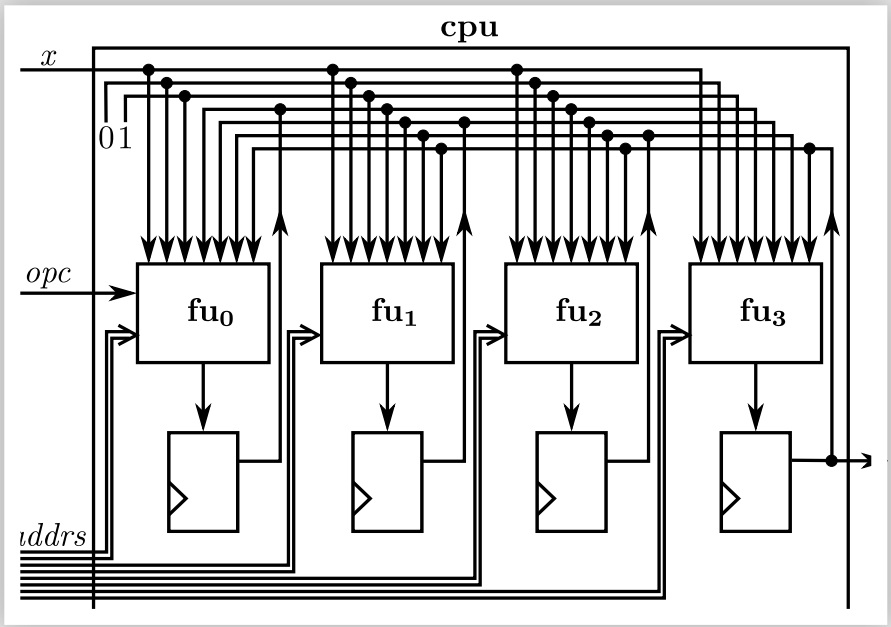
Code Listing 1: Mealy-Machine.

Een CPU ontwerp

Om meteen in het diepe te springen en goed te laten zien welke abstractiemechanismen er allemaal in CλaSH beschikbaar zijn, gaan we meteen over tot het ontwerpen van een simpele CPU. Dit voorbeeld zal gebruik maken van de verschillende abstractiemechanismen die we tot onze beschikking hebben binnen CλaSH, zoals: hogere-orde functies, partial application, lambda-expressies, pattern matching, en type afleiding. Een schematische weergave van de CPU die we willen ontwerpen is te zien in Figuur 2.

De processor zal bestaan uit 4 executie-eenheden (function units), fu_0 … fu_3, die elk een binaire operatie zullen uitvoeren. Elke executie-eenheid heeft 7 data inputs en twee adres inputs. De twee adressen geven aan welke van 7 data inputs als operanden van de functie gebruikt moeten worden. Deze 7 data inputs zijn: een externe input x, twee constanten (0 en 1) en de uitgangswaarden van de 4 executie-eenheden (uit de vorige kloktik). De uitvoer van de processor als geheel is de oude waarde van de executie-eenheid fu_3. De executie-eenheden fu_1, fu_2, en fu_3 voeren een vooraf vastgestelde functie uit. Executie-eenheid fu_0 heeft daarentegen een extra opcode ingang, opc, zodat er uit verschillende functies gekozen kan worden. De "instructies" van deze CPU bestaan dus uit de opcode en de 8 adressen, die bijvoorbeeld uit een extern geheugen afkomstig zouden kunnen zijn.

Code Listing 2 laat de volledige code zien die de eerdergenoemde CPU beschrijft. We zullen hier stap voor stap doorheen lopen. Op regel 1 en 2 worden twee type-aliassen aangemaakt, waar de eerste type-alias Word een alias voor 16-bit signed integers is en de tweede type-alias CpuState een alias voor het geheugenelement van



Figuur 2: CPU Ontwerp

```
1  type Word = Signed D16
2  type CpuState = State (Vector D4 Word)
3
4  fu op inputs (a1, a2) =
5    op (inputs!a1) (inputs!a2)
6
7  fu1 = fu (+)
8  fu2 = fu (-)
9  fu3 = fu (*)
10
11 data Opcode = ShiftL | Xor | Equal
12
13 multiop ShiftR = shiftR
14 multiop Xor    = xor
15 multiop Equal  = \a b -> if a == b then 1 else 0
16
17 fu0 opc = fu (multiop opc)
18
19 {-# ANN cpu TopEntity #-}
20 cpu :: CpuState
21  -> (Word, Opcode, Vector D4 (Index D7, Index D7))
22  -> (CpuState, Word)
23 cpu (State s) (x,opc,addrs) = (State s', out)
24  where
25   inputs = [x,0,1,s!0,s!1,s!2,s!3]
26   s'     = [ fu0 opc inputs (addrs!0)
27            , fu1 inputs (addrs!1)
28            , fu2 inputs (addrs!2)
29            , fu3 inputs (addrs!3)
30            ]
31   out    = vlast s
```

Code Listing 2: Simple CPU

de CPU geeft. Het geheugen element bestaat uit een vector van vier elementen van het type Word.

Op regel 4 en 5 staat de 'template' functie voor elke executie-eenheid. De beschrijving van fu maakt gebruik van het feit dat functies zelf argumenten mogen zijn van een functie. Het eerste argument van fu, op, is namelijk de binaire operatie die de executie-eenheid uit moet voeren. Het tweede argument is een vector met de zeven data inputs, het derde argument is een tupel met daarin de twee adressen. Wat de fu functie doet (regel 5) is twee operanden selecteren uit inputs en de operatie op hierop toepassen. Merk op dat fu tevens een polymorfe functie is: het kan omgaan met inputs van verschillende lengtes en verschillende element types.

Nu we een 'template' hebben voor de executie-eenheden kunnen we overgaan tot het definiëren van fu_1, fu_2, en fu_3 (regels 7 t/m 9). Hier geven we elke executie-eenheid een eigen operatie mee die hij moet uitvoeren. Merk op dat fu drie argumenten heeft, maar nu maar op één argument toegepast wordt. Dit betekent dat fu_1, fu_2, en fu_3 ook nog functies zijn die nog twee argumenten verwachten.

Om de laatste executie eenheid, fu_0, te definiëren, moeten we eerst het Opcode type (regel 11), en een functie multiop (regel 13 t/m 15) maken. De multiop functie kan gegeven een opcode een bepaalde functie teruggeven. De multiop functie maakt gebruik van zogenaamde Pattern Matching, waarbij de versie van de functie gebruikt wordt waarbij het pattern (tussen de functienaam en het = teken) overeen komt met de daadwerkelijke argument. Indien het eerste argument van multiop dus gelijk is aan ShiftR zal hij de functie shiftR teruggeven (let op het verschil in hoofdlettergebruik).

De waarde van de Equal case van multiop is een zogenaamde lambda-abstractie. Dit is een anonieme functie, die in dit geval de twee argumenten a en b heeft. Op regel 17 definiëren we dan eindelijk fu_0, welke dus een extra argument, een opcode, meekrijgt.

Alle componenten worden uiteindelijk samengevoegd in de cpu functie (regel 20 t/m 32). Op regel 19 geven we tevens aan de cpu de top-entiteit is en dus de top van de functie hiërarchie is. De top-entiteit mag niet polymorf of hogere orde zijn. Om er voor te zorgen dat de functie cpu niet meer polymorf is moeten we hem voorzien van een typesignatuur. Dit doen we op regel 20 t/m 22, waarbij we dus ook het gebruik van de CpuState type alias terug zien. Omdat het type van cpu nu is gegeven, kan de compiler tevens de concrete types afleiden voor alle andere functies die gebruikt worden door cpu.

Hoewel dit een vrij simpel CPU-ontwerp is, geeft het toch aan welke abstractie mogelijkheden er beschikbaar zijn in CλaSH en hoe compact een dergelijk ontwerp opgeschreven kan worden. De CPU kan nu gesimuleerd worden met een Haskell compiler of interpreter of naar VHDL vertaald worden met de CλaSH compiler.

Verder met CλaSH

Wil je meer weten over CλaSH, of wil je zelf eens spelen met de code of de compiler, ga dan naar de CλaSH website[4], waar je de compiler en verdere documentatie kan vinden. Tevens is het functioneel hardware ontwerpen een actief onderzoeksgebied binnen de CAES vakgroep: ben je geïnteresseerd in een (master)opdracht, kom dan rustig eens langs bij Jan Kuper[5] of Christiaan Baaij

Bronnen

[1] **CAES: Computer Architecture for Embedded Systems**

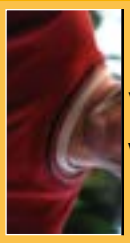
[2] **CλaSH: CAES Language for Synchronous Hardware**

[3] **The Haskell Programming Language**
http://www.haskell.org

[4] **CλaSH**
http://clash.ewi.utwente.nl

[5] **Jan Kuper**
email: j.kuper@ewi.utwente.nl

ENIAC: Van de voorzitter



Berend van den Brink

Voorzitter ENIAC

Berend van den Brink is voorzitter van ENIAC: de ENschedese Informatica Alumni Club. ENIAC is de alumnivereniging voor oud-studenten Informatica, bedrijfsinformatietechnologie en Telematica aan de Universiteit Twente.

Voor slechts € 5,- per jaar kan je al lid worden van deze club. Je krijgt dan in ieder geval de Vivats die jaarlijks verschijnen (meestal zo'n 4 stuks, maar niet helemaal per kwartaal) en uitnodigingen voor de activiteiten die we organiseren (meestal per mail). Daar mag je dan vervolgens (veelal gratis!) aan deelnemen. En al doe je maar eens in de paar jaar ergens aan mee, die € 5,- kan toch bijna iedere informatica-alumnus wel missen? Zo houd je toch nog wat binding met je wetenschappelijke roots en af en toe contact met vrienden uit je studietijd.

Berend van den Brink
voorzitter@eniac.utwente.nl

Laatst ben ik weer eens voor een dagje op de UT geweest. Voor vele lezers van de I/O-Vivat waarschijnlijk "dagelijkse kost". En dan bedoel ik natuurlijk met name de vele Inter-*Actief*-leden die deze Vivat ook lezen. Voor andere lezers zal het mogelijk lang geleden zijn dat ze de UT gezien hebben. Ik kom er niet vaak, maar toch wel regelmatig en nog steeds verandert er steeds weer wat op de Campus. Directe aanleiding was de promotie van Machiel van der Bijl, oud ENIAC-bestuurder en voor mij ook oud-collega, in de relatief nieuwe Prof.dr. G. Berkhoffzaal (voorheen Waaier 4). Een in vele opzichten unieke gebeurtenis. Natuurlijk allereerst voor Machiel, want promoveren doe je niet iedere dag. Gefeliciteerd!

Maar voor Prof. dr. ir. Arend Rensink was het ook de eerste promotie waar hij als "Prof." bij zat, in de rol van promotor. En voor de rector, Prof. dr. ir. Brinksma was het de eerste keer dat hij als rector en promotor de bul mocht tekenen. En ik weet vrijwel zeker dat het ook de eerste promotie is geweest waarbij tijdens de felicitatie van de promovendus een gloedvol betoog werd gehouden voor het werven van ENIAC-actievelingen. Dank je, Arend! Als het nou niet lukt, weet ik het niet meer.

Rond de promotie had ik diverse afspraken en heb ik mensen gesproken over de toekomst van ENIAC. Gelukkig overheerst de mening dat een vereniging voor INF/BIT/TEL-alumni er zeker moet zijn en dat het de moeite waard is daar energie in te steken. Gelukkig, dan heb ik dat de afgelopen jaren in ieder geval niet voor niets gedaan. Ook hebben we gesproken over de bekendheid van ENIAC en van Inter-*Actief* en mogelijkheden van verdergaande samenwerking. Natuurlijk is een alumnus wat anders dan een student, maar een alumnus is natuurlijk wel altijd student geweest. Nou biedt Inter-*Actief* tijdens de studie zoveel directe voordelen dat eigenlijk iedere student uit de doelgroep lid wordt. Mogelijk dat we die binding en de daaruit voortvloeiende naamsbekendheid van Inter-*Actief* beter kunnen benutten dan we nu doen.

Misschien staat ENIAC niet bij iedereen even positief bekend. Daar kan je je bij neerleggen en dan houdt het op een gegeven moment vanzelf op. Of je doet er wat aan. En wat er ook gezegd wordt, ENIAC is altijd een vereniging geweest waar ruimte is voor initiatieven van leden. Dus vind je dat ENIAC beter kan? Beter moet? Doe er wat aan. Mis je een activiteit? Organiseer er hem maar. Het bestuur biedt je alle ruimte en ondersteuning. Wil je iets met je vroegere doegroep? Een activiteit die openstaat voor alle leden? Het bestuur staat achter je, met ondersteuning in de communicatie en een financiële bijdrage. Laten we er met z'n alle voor zorgen dat ENIAC weer een bruisende vereniging wordt.

Vrij snel na mijn "dagje Enschede" kwam er een mail waarmee drie oud-bestuurders van Inter-*Actief* hun belangstelling voor een bestuursfunctie kenbaar maakten. Gelukkig. Want Francis heeft na vele jaren in het bestuur te kennen gegeven dat hij aan het eind van het jaar wil stoppen. En met een nieuw bestuur kan ENIAC er weer fris tegenaan. Met drie nieuwe actievelingen zijn we goed op weg, maar natuurlijk kunnen we nog veel meer impulsen gebruiken. Doe je mee? Meld je snel aan of lees eerst de oproep verder in dit katern en kies iets wat je aanspreekt. En natuurlijk wil ik ook graag je aandacht vestigen op het artikel van de scriptieprijs-winnaars Christiaan Baaij & Matthijs Kooijman over "Functioneel hardware ontwerpen in CλaSH", verderop in dit ENIAC-katern.

Tenslotte wil ik jullie vragen of je 26 november al in je agenda hebt gezet? Dan organiseert de UT een groots opgezette alumnidag omdat de UT dit jaar 50 jaar bestaat. Een goede gelegenheid voor een "dagje Enschede" en het ontmoeten van oude bekenden.

Op bezoek bij: Shell

Floor de Jong
External Sourcing Coordinator at Shell

SHELL, AFSTUDEREN, ICT, TaCIT, CONSULTANCY

Interview Floor de Jong

Ditmaal zijn we op bezoek geweest bij Floor de Jong. Floor heeft BIT gestudeerd aan de Universiteit Twente en is nu werkzaam bij Shell. Naast haar studie BIT heeft Floor een jaar in het bestuur van Inter-*Actief* gezeten (bestuur 26), veel op het gebied van voorlichting georganiseerd, meerdere jaren in de BIT onderwijs kwaliteit commissie gezeten, het Skill Certificate van de Student Union opgezet en ze is op studiereis naar India geweest.

Kun je ons wat meer over jezelf vertellen?

Ik ben dus Floor de Jong, sinds kort 27 jaar en ik werk sinds 2 jaar bij Shell. Daarvoor heb ik al 9 maanden afgestudeerd bij Shell, hoewel dat in een heel ander onderdeel was dan waar ik nu werk. Het afstuderen was echter zo goed bevallen dat ik dacht: 'hier wil ik wel blijven hangen' en gelukkig was het bij Shell ook zo goed bevallen dat ik hier in maart 2009 aan de slag kon. Ik woon tegenwoordig in Voorburg met mijn vriend en werk in Rijswijk.

Hoe ben je bij Shell terecht gekomen?

Ik had tijdens mijn studie nooit het idee om bij Shell af te studeren. Waar ik me destijds vooral op oriënteerde was consultancy en dan specifiek strategy consultancy. Ik ben bij zo'n beetje alle strategy houses langs geweest, maar daar kun je niet afstuderen. Ik ben bij Shell terecht gekomen via het bestuur van Inter-*Actief*. Het bestuur van Inter-*Actief* werd uitgenodigd voor een besturendag waar ik er achterkwam wat er allemaal op het gebied van ICT bij Shell plaatsvindt. Vervolgens werd mij door een Ebbinge recruiter aangeraden om eens bij Shell te kijken.

Figuur 1: Floor op haar werkplek

Ik heb me daarna ingeschreven voor de business challenge van Shell, Gourami. Nadat ik online mijn CV had ingevuld werd ik uitgenodigd voor een interview en toen ik daar doorheen was kon ik in principe ingedeeld worden in een case. Ze bleken echter geen ICT cases te hebben bij die versie van Gourami, dus ik zou een case moeten doen die weinig met mijn achtergrond te maken zou hebben. Daarom boden ze mij aan om in plaats van een case een stage bij Shell te komen doen, waarop ik vroeg of ik ook bij Shell kon afstuderen en dat bleek te kunnen! Op dat moment baalde ik er eigenlijk flink van, aangezien ik me verheugd had op Gourami, maar achteraf gezien ben ik erg blij mee, aangezien ik denk dat een stage je veel meer inzicht geeft in de werking van een bedrijf dan een business course.

Wat heb je voor afstudeer opdracht gedaan?

Bij Shell zijn er in de loop van de tijd allerlei programma's ontwikkeld, bijvoorbeeld de interne vacaturebank. De applicatie support van deze programma's is uitbesteed aan een Indiaas bedrijf. Bij applicatie support moet worden gedacht aan bugs oplossen, nieuwe functionaliteit toevoegen maar ook mensen die ergens niet uit komen hulp bieden, oftewel helpdesk.

Het bleek erg lastig om de medewerkers in India aan te sturen. De cultuur is daar extreem anders, de werkprocessen zijn anders en de manier waarop zij succes meten is ook heel anders. Als je bijvoorbeeld in Nederland aardig bent tegen klanten van de helpdesk wordt je hulp vaak al snel als goed beoordeeld.

In India is het veel meer tot in detail gespecificeerd hoe succes moet worden gemeten. Ze meten bijvoorbeeld hoeveel tickets ze binnen krijgen, hoe snel ze worden opgelost, hoeveel er naar een volgend niveau (bijvoorbeeld naar medewerkers die werkelijk het programma onderhouden) doorgaan en dat botst nog wel eens met de Nederlandse manier van werken.

Voor mijn afstudeeropdracht werd ik gevraagd om een besturingsmodel te ontwikkelen voor de relatie met een dergelijk bedrijf. Dit is in feite een raamwerk wat beschrijft wat voor rollen er bij zo'n relatie nodig zijn, bijvoorbeeld een relatiemanager en een applicatie specialist binnen je eigen organisatie en teamleiders bij de externe partij.

Heb je veel aan je studie gehad?

Als je bij Shell aan de slag gaat met een IT-achtergrond, dan wordt je niet iemand die erg de technologie induikt, die echt gaat programmeren of ontwikkelen. Je wordt iemand die gaat nadenken over IT op een hoger niveau. Tijdens mijn afstudeeropdracht heb ik geen enkele regel broncode gezien, maar ik heb wel met de applicatiebeheerders en hun bazen gepraat . Het gaat er dus om dat je snapt waar het over gaat, dus dat je weet wat programmeren is en wat een architectuur is, maar vooral dat je op een niveau hoger kan kijken en dat allemaal in verband met elkaar kan brengen en daar is BIT een zeer geschikte opleiding voor.

Als graduate wordt je bij Shell als een talent gezien en er zijn mensen die nadenken over wat jij voor Shell kan betekenen in 10 jaar en hoe ze jou hier het beste bij kunnen steunen en welke cursussen je zou moeten volgen etc. Je wordt bij Shell echt behandeld als een waardevolle werknemer en ze willen graag je potentie zo goed mogelijk benutten en dat geeft best een tof gevoel!

Waar ben je op het moment mee bezig?

Ik ben op het moment bezig met mijn derde grote klus bij Shell en ga al bijna met mijn vierde beginnen. Het is bij Shell ook de bedoeling dat je in korte tijd bij meerdere functies langsgaat. Op het moment ben ik 'External Sourcing Coordinator for P&T TaCIT'. Hoewel dit erg chique klinkt komt het er op neer dat ik mensen inhuur voor projecten.

We hebben (in een versimpelde versie) bij Shell twee IT organisaties. De 'standaard IT afdeling' die gaat over dingen zoals applicatie support en de IT infrastructuur. Daarnaast heb je TaCIT, oftewel Technical en Competative IT en dit is de tak die zelf programma's ontwikkelt en Shell anders maakt dan bijvoorbeeld BP of ExxonMobil.

Een van de punten waarin Shell de afgelopen 30 jaar uitblinkt, is het vinden van olie. Shell doet overal ter wereld testen om te zoeken naar olie en gas en dit levert petabyte's aan data op, welke worden geïnterpreteerd door onze supercomputers. Hoewel concurrenten dit natuurlijk ook doen, blinkt Shell uit bij het vinden van olie en gas vanwege de supergoede interpretatie software die door TaCIT wordt ontwikkeld.

Mijn functie is om externe mensen te zoeken en in te huren voor het ontwikkelen van dergelijke software. Dit is daarom geen goed voorbeeld van wat je mijn studie kan doen, maar er is ook nog een andere reden dat ik hier zit. TaCIT in zijn huidige vorm bestaat namelijk pas sinds 2010 en de 'tools, policy's en processes' van deze functie, dus hoe zoeken we mensen, welk tarief moeten ze krijgen etc, zijn nog niet opgesteld, dat mag ik nu doen en daarna ga ik naar mijn volgende baan.

Waar heb je een hekel aan en wat vind je geweldig binnen Shell?

Waar ik een hekel aan heb is de bureaucratie binnen Shell, er zijn ontzettend veel policies en soms lijkt de situatie bijna niet werkbaar en dat is iets waar je tegen moet kunnen, helaas niet een van mijn sterkste kanten.

Wat ik echter super leuk vind aan Shell is het internationale karakter en dan bedoel ik niet eens dat je veel gaat reizen, want dat geldt lang niet altijd. Ik zit nu in een kamer met een Ier, een Portugees, een Iraniër, een Canadees en mijn baas uit Indonesië. Dat is echt ontzettend leuk, maar dat heb je pas door als je hier werkt. Verder is de work-life balance erg goed bij Shell en uiteindelijk ook een van de redenen dat ik niet bij een consultancy bedrijf aan de slag ben gegaan.

Bedankt voor het Interview!

IMMORALITY

05.10.11

GROLSCH VESTE

Immorality

Immorality
Symposium Commissie

IMMORALITY, SOCIAL ENGINEERING, HACKING, PRIVACY

De duistere kant van IT?

Op 5 oktober houdt Inter-*Actief* weer een studentensymposium: Immorality. Het thema deze dag zal 'de duistere kant van IT' zijn. Als IT-studenten beseffen we ons natuurlijk maar al te goed dat het digitale iets moois is. Toch zijn wij de laatsten om te ontkennen dat er ook een andere kant aan zit. Deze andere kant wordt, nu computers prominenter en prominenter aanwezig zijn in de samenleving, steeds invloedrijker. Daarom is het interessant en belangrijk om hier eens een dag aandacht aan te besteden.

Natuurlijk is 'de duistere kant' een zeer breed begrip. Het eerste waar u aan denkt is waarschijnlijk hacking en de bijbehorende security, maar de duistere kant van IT is veel groter. Zo zullen privacy, wetgeving en social engeneering een aantal onderwerpen zijn die besproken zullen worden. Over de exacte invulling kunnen we nog niet veel bekend maken, maar één van de sprekers zal David Nieborg zijn. Verderop in dit katern staat een interview met hem.

Hacking

Zoals gezegd is dit waarschijnlijk het meest voor de hand liggende onderwerp binnen de immorele kant van de IT, dus wie zijn wij om er geen aandacht aan te schenken? Beide kanten van de medaille zullen daarbij aan bod komen, dus zowel het hacken als het beveiligen daartegen. Het is van het grootste belang dat het web goed beveiligd is. Stel je voor dat iedereen bij je online bankieren account kan. Dit mag natuurlijk niet, maar het is verbazingwekkend hoeveel sites slecht beveiligd zijn. Hoe kan dit en hoe kan dit probleem opgelost worden?

Privacy

Hoe vaak plaats je je privégegevens op het internet? En hoe gaan bedrijven hier mee om? Je wil natuurlijk niet dat er met deze gegevens rondgestrooid wordt, maar daar heb je zelf maar weinig controle over. Hoe bedrijven de veiligheid van je privégegevens waarborgen en hoe de wetgeving op dit gebied in elkaar zit is de vraag die wij ons stellen op Immorality.

Social engineering

Je kunt je technisch nog zo goed beveiligen, maar is de mens niet de zwakste schakel van de beveiliging van je systemen? Social engineering is de kunst van het misbruiken van de mens en zijn psyche om op die manier een systeem te kraken. Dat klinkt wellicht wat onduidelijk, maar het voorbeeld van e-mails die op naam van een bank worden verzonden en de ontvanger vragen zijn wachtwoord op te sturen, is heel concreet. Uiteraard zijn er vele andere manieren.

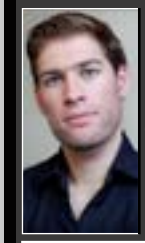
Overige informatie

Met dit interessante, prikkelende thema denken we een fantastisch symposium neer te gaan zetten. Naast boeiende lezingen zullen er een aantal bedrijven aanwezig zijn met een stand en een aantal werknemers. Deze bedrijven komen speciaal voor jou naar het symposium en daarmee vormt Immorality een uitstekende gelegenheid om op zoek te gaan naar een bachelor- of afstudeeropdracht, of gewoon voor een eerste kennismaking. Uiteraard zul je de hele dag voorzien worden van koffie, thee, frisdrank en lunch, en is er een borrel achteraf. En dat alles voor een heel schappelijke prijs. We zien je graag, de 5e van oktober!

Bronnen

www.immorality.nl
Datum: 5 Oktober 2011
Grolsch Veste
Kosten: 5 euro

Interview:David Nieborg



David Nieborg
Game-onderzoeker

GAMEMAKERS,GAMES,FEITELIJKE ASPECTEN, GEWELD, MEDIA

Zijn games wel zo onschuldig?

David Nieborg is 30 jaar oud en is cum laude afgestudeerd aan de Universiteit Utrecht met het onderwerp interactie tussen populaire cultuur en het Amerikaanse leger. Hij woont momenteel in Amsterdam, waar hij in juni promoveert aan de Universiteit van Amsterdam op het gebied van politieke economie van de game industrie. Daarnaast is hij journalist voor het NRC next, schrijft hij columns over computergames en is hij adviseur over social media. Hij zal op 5 oktober 2011 spreken op het symposium Immorality.

Wordt er in de game-industrie gekeken naar morele aspecten in een videospel?

Gamemakers richten zich meer op de feitelijke aspecten in een spel. De nadruk ligt meer op hoe wapens werken en hoe groot een bominslag moet zijn, maar kijken daarbij minder naar de morele kant. Je schiet vrijwel altijd als blanke Amerikaanse soldaat een terrorist neer, maar er wordt geen aandacht geschonken aan waarom deze man terrorist is. Je bent ook nooit de terrorist in singleplayer spellen. In multiplayer spellen kan je dan nog wel de terrorist spelen, maar dan mist er een verhaal achter het terrorist zijn.

Maar houden bedrijven er dan zelf rekening mee?

Er zijn zeker gameontwikkelaars die hiermee experimenteren. Maar de ironie is een beetje dat gamemakers nog wel bereid zijn dit aspect op te nemen in hun spellen, maar dat het de media en de ouders zijn die dit niet verantwoord vinden. Mensen vinden het geen probleem als je Schindler's List kijkt, waar toch ook ethisch onverantwoorde dingen gebeuren. Maar als iemand een terrorist speelt is hij weer schietlustig.

Kijkend naar voorbeelden als de schietpartij in Alphen aan de Rijn, heeft de gamewereld dan invloed op de echte wereld?

Bij een schietpartij als in Alphen of bij Columbine kun je eigenlijk alleen maar algemene observaties doen. Er roept al vrij snel iemand dat de oorzaak bij de gewelddadige game ligt, terwijl dit nog helemaal niet aangetoond kan worden. Er zijn zo veel factoren die invloed hebben op deze mensen dat je niet kunt zeggen dat het aan gewelddadige games ligt. Eric Harris bleek achteraf een psychopaat te zijn, dat is een veel betere verklaring dan dat hij gewelddadige games speelt. Er zijn altijd drie punten bij dit soort problemen: je kunt het nooit voorkomen, er is geen eenzijdig profiel dat bij deze mensen hoort en er is niet één factor aan te wijzen waardoor dit komt. De FBI heeft dan ook gezegd dat de oorzaak niet bij games lag. Journalisten doen er daarom ook geen eer aan om dit probleem aan computergames toe te schrijven en moeten eerst het onderzoek afwachten.

De oorzaak van dit probleem is eerder toe te wijzen aan populaire cultuur. Na het voorval bij Columbine zijn in heel Amerika meer van dit soort schietpartijen voorgevallen. Er wordt dus eerder gekopieerd van andere gebeurtenissen. Nu kan het natuurlijk zijn dat dit gekopieerd wordt van games. Maar zulke schietpartijen hebben vaak eenzelfde patroon; de dader wil eerst anderen van hun leven nemen voordat ze hun eigen leven nemen. En zulk soort patronen en scripts vind je eerder terug in films dan in games.

Hebben games dan geen invloed op de echte wereld?

Games hebben wel degelijk invloed op de mens, maar dat ligt geheel aan de context. Als je gewoon speels en competitief een spel online aan het spelen bent heeft het schieten op personen geen invloed op je hersenen. Het is niet zo dat een klik op de knop meteen iets omschakelt in je hersenen waardoor je schietlustig wordt. Maar als je het in een andere context plaatst, bijvoorbeeld in de politie of het leger, worden schietspellen wel degelijk gebruikt om procedures te leren aan soldaten. Dan kan er ook gecontroleerd worden wat ze goed deden en wat beter kan. Er zijn bijvoorbeeld ook levels opgedoken van de compound van Al Qaida. Wie weet hebben de Navy SEALs dit level wel vele malen gespeeld voordat ze daadwerkelijk de inval hebben gedaan.

Dus games kunnen ook een boodschap overbrengen

Ook games kunnen inderdaad een boodschap overbrengen, maar dat is erg afhankelijk voor de personen die het spelen. Ik zou als ik kinderen had ze geen games laten spelen waar ze niet oud genoeg voor zijn, net zoals ik geen Schindler's List met ze zou kijken. Daarom worden er ook ratings aan games gegeven. Kinderen zijn daar nou eenmaal vatbaarder voor. Maar naar mijn idee hebben games niks te maken met de schietpartijen en maken ze mensen niet gewelddadiger.

Hartelijk bedankt voor het interview en tot op 5 oktober 2011.

VOLGENDE KEER IN I/O VIVAT

- COMPLEX ADAPTIVE SYSTEMS
- NFC (BETALINGSCHIPS)
- DNS FILTERING / PROTECT IP ACT

Advertentie
ASML