Ontwikkelen en kopiëren van software

TU/e technische universiteit eindhoven

Intreerede
13 juni 2003

prof.dr.ir. M.J.I.M. van Genuchten

ontwikkelen en kopiëren van software

/ faculteit technologie management

Intreerede

Uitgesproken op 13 juni 2003
aan de Technische Universiteit Eindhoven

ontwikkelen en kopiëren

van software

prof.dr.ir. M.J.I.M. van Genuchten

Inleiding

Mijnheer de Rector Magnificus, dames en heren,
Terwijl je thuis voor de televisie zit, word je gebeld op je mobiele telefoon. De vraag is of je direct een document kunt versturen naar een collega die morgen met de auto naar een klant gaat. Tijdens het gesprek regel je dit vanaf je pc via het Internet. Je bent blij dat je de pc kunt gebruiken, want een huisgenoot heeft zojuist de 'speler van de wedstrijd' geselecteerd en doorgegeven via een SMS. Nu wil hij weer terug naar de chat met zijn vrienden.

Deze situatie speelt zich niet af in een huis van de toekomst, maar is alledaagse realiteit. We beseffen niet altijd, dat de functionaliteit van de genoemde producten grotendeels gerealiseerd is in de vorm van software. Ieder van deze producten bevat ten minste enige honderdduizenden regels software.
De hoeveelheid software in producten neemt snel toe. Draadloos telefoneren, een kijkoperatie, communiceren met de rest van de wereld via Internet, leren in virtuele teams: het zou allemaal niet mogelijk zijn zonder software.
De gevolgen voor bedrijven en industrieën zijn groot. Voorbeelden zijn veranderingen in de computerindustrie tussen 1984 en 1994. IBM en Digital waren in 1984 marktleiders in deze industrie. In 1994 was hun positie overgenomen door bedrijven als Microsoft en Oracle, softwarebedrijven die tien jaar eerder nog nauwelijks op de kaart stonden.

De opkomst van software is dagelijks nieuws. Twee willekeurige voorbeelden uit de krant van 12 februari 2003. Het eerste bericht gaat over de filmindustrie en vermeldt het feit dat animaties inmiddels via software en niet meer met pen en papier worden gemaakt de verhoudingen in de tekenfilmindustrie drastisch heeft veranderd. Disney's positie wordt bedreigd door Pixar, een bedrijf dat in 1995 pas zijn eerste film uitbracht (Holson 2003). Een tweede voorbeeld uit de krant bericht dat software zelfs belangrijk wordt voor producten waarvan je het niet zou verwachten. Zo is het aannemelijk dat LED's

(Light Emitting Diodes) als lichtbron steeds populairder worden. Deze zijn immers zuiniger met energie en geven ruime mogelijkheden zoals natuurlijk licht en de optie om de kleur van licht te veranderen. Software is noodzakelijk om vele LED's aan te sturen en te synchroniseren (Feder 2003).
Deze voorbeelden zijn een toevallige greep uit de krant en zijn dus lang niet uitputtend. Maar ze laten zich lezen als tekens van een algemeen verschijnsel: bijna elke bedrijfssector zal aanzienlijk door software worden beïnvloed. Of zoals Humphrey (2002) het uitdrukt: 'Every company is a software company'.

De titel van mijn verhaal is 'Ontwikkelen en kopiëren van software'. Ontwikkelen van software blijkt lastig. Iedereen kent wel een voorbeeld van een project dat uit de hand is gelopen of een softwareproduct met fouten. In het eerste deel van mijn verhaal geef ik aan hoe een groot deel van deze problemen inmiddels wordt opgelost. Kopiëren of reproduceren van software is zo eenvoudig dat iedereen het kan. Een tweede identieke kopie is snel gemaakt en eenvoudig over de wereld verspreid, want de transportkosten zijn verwaarloosbaar. Dat is een belangrijk verschil met de reproductie van fysieke producten zoals huizen, auto's, aardappelen of elektronica. Kopiëren van software is onderwerp van het tweede deel van mijn rede. Het gaat hier met name om verreikende consequenties voor bedrijven en industrieën. De samenvatting van mijn rede staat schematisch in tabel 1.

tabel 1

Ontwikkelen en kopiëren van software.

	Ontwikkelen	**Kopiëren**
Handeling	Lastig en wetenschappelijk interessant – deel 1 lezing	Triviaal
Implicaties	Vergelijkbaar met andere disciplines	Veel onduidelijke consequenties voor bedrijven / economieën en wetenschappelijk interessant – deel 2 lezing

Het verhaal wordt afgesloten met een overzicht van het onderwijs van software management binnen de faculteit Technologie Management en de rol van virtuele teams daarin.

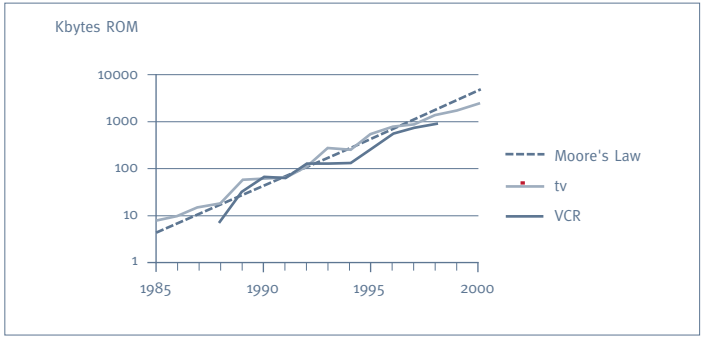
Toename van software

Allereerst een drietal definities. De eerste definitie betreft een omschrijving van software: 'computer programs, procedures, rules and possibly associated documentation and data pertaining to the operation of a computer system' (Reifer 1993). Een tweede definitie benadert software vanuit een andere invalshoek: 'software can be viewed as executable knowledge' (Humphrey 1989, pagina 18). Software is kennis die zo goed en precies is omschreven, dat een machine de kennis kan executeren. Deze definitie zal ik in het tweede deel van mijn betoog gebruiken.
Software heeft als belangrijke eigenschap dat het een abstract product is. Als je er één exemplaar van hebt, is reproduceren technisch gezien eenvoudig. In economische termen: 'marginale kosten van een additioneel product zijn nul' (Varian 2001).
Eerder introduceerde ik de term software management. En daarbij hoort de derde en laatste definitie: 'Software management richt zich op de ontwikkeling van softwareproducten en softwaremarkten'.

Er zijn twee belangrijke redenen voor de groei van de hoeveelheid software. Ten eerste blijkt software een effectieve en efficiënte technologie om functionaliteit voor een klant te realiseren. Wat dit betreft is het einde nog niet in zicht. Op veel gebieden liggen nog mogelijkheden tot verbetering. Ik noem er een paar: spraakherkenning, visualisatie in 3-d, medische toepassingen en besturing van voertuigen die zich 'bewust' zijn van hun omgeving (Weber 2003).
De tweede reden voor de groei van software wordt gegeven door snelle ontwikkelingen in hardware en in het bijzonder in chips. Deze groei wordt samengevat in de wet van Moore die stelt, dat de dichtheid van transistors op een chip iedere 18 maanden verdubbelt. Dit leidt ertoe, dat er steeds meer functionaliteit op zo'n chip gerealiseerd kan worden en dat er steeds meer software nodig is om die chip te benutten. De hoeveelheid software groeit snel en verdubbelt iedere twee jaar in vele producten. Als voorbeeld in figuur 1 de toename van software in televisies en videorecorders. Zoals uit de figuur blijkt, volgt de toename van de hoeveelheid software de wet van Moore op de voet.

figuur 1

Groei van software in tv's en VCR's en de wet van Moore (Aerts 2001).



De verwachting is dat de wet van Moore nog wel enige tijd zal blijven gelden. Een recente enquête onder fellows van de IEEE geeft aan, dat 52 procent van de respondenten verwacht dat de wet van Moore nog 5-10 jaar geldig blijft en dat 17 procent denkt dat de wet van Moore zelfs meer dan 10 jaar geldig zal zijn (Applewhite 2003).
Toenemende mogelijkheden van hardware zullen leiden tot het integreren van meer functionaliteit in apparaten en tot het exploreren van nieuwe functionaliteit. De hoeveelheid software in producten zal dus ook nog jaren in hoog tempo blijven toenemen.

Ontwikkelen van software

Hoe beheersen we de ontwikkeling van software? Jarenlang liet het antwoord zich eenvoudig samenvatten: we beheersen die ontwikkeling nauwelijks! Overschrijdingen op softwareprojecten waren normaal. Er werd niet eens de moeite genomen om de mate van overschrijding bij te houden (Heemstra 1989, Genuchten 1991). Een softwareproduct bevatte bij oplevering een onbekend aantal fouten dat gratis werd meegeleverd. Gebrek aan softwarekwaliteit had ernstige financiële en ook persoonlijke consequenties. Iedereen leed eronder, maar accepteerde de situatie. Blijkbaar kon het niet anders, was de gedachte.

Oorzaken waren deels structureel, deels te vermijden. Software is een abstract product en dat betekent dat een reproductieproces ontbreekt. Resteert een moeilijk voorspelbaar ontwikkelproces. In de komende alinea zal ik proberen enig begrip te kweken voor het feit dat software wel eens te laat is en er wel eens een fout in zit. Daarna gaan we kijken hoe software management kan bijdragen aan een betere beheersing van de ontwikkeling en een betere kwaliteit van software.

De hoeveelheid software is omvangrijk. Neem een softwareproduct van 1 miljoen regels code. Dat is ongeveer de hoeveelheid software in een medische scanner en slechts één procent van de software die nodig is om een intercontinentaal telefoongesprek mogelijk te maken. Een miljoen regels broncode op papier betekent zo'n 20.000 pagina's. In iedere regel op iedere pagina kan een fout zitten. Een miljoen regels code schrijven vraagt tussen de 40 en 400 mensjaren ontwikkelwerk, afhankelijk van de soort software. Logisch, dat er fouten ontstaan in zulke grote ontwikkelprojecten. Vergelijk dit eens met fouten in een wettekst of in een afstudeerverslag of krant. Bij een fout in de krant zijn de gevolgen vaak weinig ingrijpend: bij een eenvoudige verschrijving snapt de lezer het meestal nog wel. Bij een grove fout volstaat een rectificatie, 24 uur later. Software-ontwikkelaars zijn mensen en die maken dus fouten. Het gaat erom, dat we het softwareproces zo inrichten dat we fouten uit het product halen voordat dit aan de klant wordt geleverd.

Er zijn veel manieren om de ontwikkeling van software te verbeteren. De TU/e heeft op dit gebied een reputatie dankzij het werk dat onder meer door Dijkstra aan deze universiteit is uitgevoerd. Vanuit het oogpunt van software management is het werk dat door Watts Humphrey is gestart op het Software Engineering Instituut van groot belang (Humphrey 1989). Humphrey ontwikkelde op basis van zijn ervaring en eerder werk van Deming en Juran het Capability Maturity Model (CMM) voor software. Dit model geeft vijf niveaus van procesbeheersing: initial, repeatable, defined, managed en optimizing. Hoe hoger het niveau, hoe beter de beheersing van het software-ontwikkelproces. Inmiddels zijn we tien jaar verder en zien we in een groeiend aantal bedrijven de resultaten van de verbetering van softwareprocessen. Het is duidelijk, hoe de verbetering van software-ontwikkeling kan worden aangepakt. Via meetbare resultaten zijn de voordelen van deze aanpak aangetoond (Harter 2000, Vu 1997, Haley 1997, Humphrey 2002).

Twee voorbeelden: één waar ik zelf bij betrokken ben geweest en een recent voorbeeld uit de literatuur.
Het eerste voorbeeld betreft de ontwikkeling van tv's bij Philips in de jaren 90 (Rooijmans 1996). De hoeveelheid software in een toenmalige tv was gegroeid tot zo'n 500 k ROM. Het management raakte geïnteresseerd in de ontwikkeling van software, omdat het een knelpunt was geworden. Door het invoeren van een aantal verbeteringen zijn de kwaliteit van software en de voorspelbaarheid van de ontwikkeling nu verbeterd. Vier belangrijke verbeteringen waren de verbeterde specificaties en het wijzigingenbeheer, meer gedetailleerde plannen, een wekelijkse voortgangsrapportage en software inspecties. Inspecties zijn een gestructureerde manier van reviewen waarbij een groep van drie tot vijf ontwikkelaars zoveel mogelijk fouten uit een ontwikkeldocument probeert te halen (Fagan 1986). Zo'n ontwikkeldocument kan variëren van specificatie, ontwerp tot code. Tabel 2 geeft aan hoeveel fouten in inspecties zijn gevonden alsmede hoeveel fouten er in inspecties zijn gemist en pas later in tests zijn teruggevonden.

tabel 2

Voorkomen van fouten door inspecties (Rooijmans 1996).

	Inspecties			Test		
	Aantal inspecties	Aantal voorkomen fouten	Inspanning per voorkomen fout in mensuren	Aantal gevonden fouten	Inspanning per fout in mensuren	Aantal fouten per Kbyte ROM
Project 1	12	460	0.5	117	4	3.7
Project 2	20	543	0.7	330	1.6	2.8
Project 3	180	1170	2.1	825	3.1	1.7

Uit de tabel valt af te lezen, dat inspecties een efficiënte manier zijn om fouten uit software te halen. Het vinden en oplossen van fouten in inspecties kost minder tijd dan bijvoorbeeld testen. Uit de literatuur blijkt bovendien, dat inspecties effectiever zijn dan testen. Verder blijkt uit de tabel, dat in het derde project het grootste aantal inspecties is uitgevoerd, de meeste fouten in inspecties zijn gevonden en dat de foutdichtheid in de test belangrijk lager is dan in de twee andere projecten. Is hiermee het derde project een doorslaand succes? Helaas niet, want Moore had ons al weer ingehaald. Project 3 was 4 keer zo groot als project 2 en 16 keer zo groot als project 1. De overschrijdingen van de genoemde projecten worden gegeven in tabel 3.

tabel 3

Overschrijding van begroting in procenten.

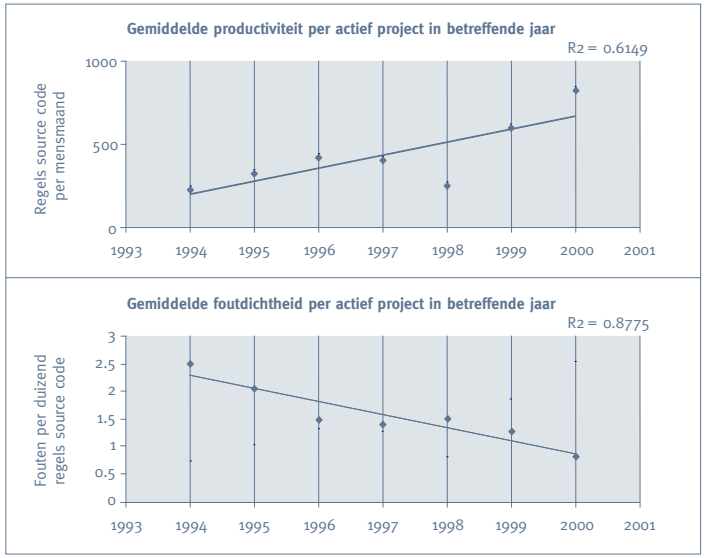
Overschrijding in procenten	Hoeveelheid software	Ontwikkelinspaning	Doorlooptijd van het project
Project 1	0	19	10
Project 2	20	30	14
Project 3	67	96	59

Tabel 3 geeft het probleem aan waarmee veel softwareprojecten kampen: het te ontwikkelen product is veel groter dan producten waarmee eerder ervaring is opgedaan. Methoden en technieken die bij kleine projecten voldoen, werken niet meer bij grote projecten. Ontwikkelprocessen zullen verder verbeterd moeten worden om aan steeds hogere eisen te kunnen voldoen.

Het tweede voorbeeld is zeer recent (McGarry 2002). De ervaringen van het SEAS laboratory worden hierin beschreven. Dit is een software-ontwikkeleenheid van 850 personen, betrokken bij de ontwikkeling van real-time software voor NASA. De organisatie opereert op niveau 5, het hoogste CMM-niveau. De empirische gegevens betreffen 30 projecten, uitgevoerd tussen 1984 en 2000. Zowel de productiviteit als de kwaliteit neemt toe met zo'n 10 procent per jaar. Het aantal fouten is afgenomen van 2.5 naar 1 per 1000 regels code. De verbeteringen tussen 1994 en 2000 staan afgebeeld in figuur 2.

figuur 2

Toename van productiviteit en afname van foutdichtheid.



Andere goede voorbeelden van systematisch werken aan verbetering van softwareproducten zijn te vinden in het werk van Humphrey (1989 en 2002) en Boehm en Basili (Boehm 2001). Het is niet meer de vraag of grootschalige software-ontwikkeling beheerst kan verlopen; een toenemend aantal organisaties toont aan dat dit kan. Het is eerder de vraag waarom veel organisaties de problemen nog steeds niet willen oplossen. Dat kan worden veroorzaakt door onwil, bijvoorbeeld omdat de betreffende softwareleverancier wordt afgerekend per gewerkt uur.

Het kan ook worden veroorzaakt door het feit dat opdrachtgevers, management en ontwikkelaars geen consequenties verbinden aan geconstateerde problemen en mogelijke oplossingen. Zij kiezen er dan voor om in de problemen te blijven zitten. Uiteindelijk zal de markt bedrijven belonen die hun ontwikkeling wel beheersen en zo softwareproblemen stelselmatig verkleinen.

Overigens blijven er ook in goede ontwikkelafdelingen interessante problemen genoeg om op te lossen. Bijvoorbeeld: de toegenomen productiviteit in het laboratorium van CSC is indrukwekkend maar niet toereikend om de groei in software volgens de wet van Moore te volgen. Dit fenomeen speelt op veel ontwikkelafdelingen: ondanks alle verbeteringen is de groep niet in staat de gewenste groei in software bij te houden. De oplossing is overigens om software niet meer te ontwikkelen maar te kopiëren, waarover later meer.
Welke wetenschappelijk interessante vragen zijn op dit moment relevant voor het beheersen van ontwikkeling? Een onderzoeksvraag waar ik mij de komende jaren op wil richten is:

Onderzoeksvraag 1
Welke processen, methoden en hulpmiddelen dragen aantoonbaar en meetbaar bij aan het effectief en efficiënt ontwikkelen van software?

Een korte toelichting op deze onderzoeksvraag: we gaan niet op zoek naar de silver bullet die alle problemen in één keer oplost. Die bestaat namelijk niet (Brooks 1987). Belangrijk in de onderzoeksvraag is de term 'meetbaar'. Lord Kelvin zei 100 jaar geleden al: "When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science" (Dunham 1983). Doel van mijn onderzoek is om via empirisch en kwantitatief onderzoek ertoe bij te dragen, dat software management 'advances to the stage of science'. Ik verwacht een vruchtbare samenwerking met collega's uit de onderzoeksschool BETA, die al een langere traditie hebben in het kwantitatief bestuderen van operationele processen (www.tm.tue.nl/beta).

Hoe gaan we dit onderzoek aanpakken? De vraagstellingen zullen zich niet beperken tot Nederland. De huidige voorzitter van NWO stelt in een interview, dat er in zijn ogen te veel groepen zijn die 'ervan uitgaan dat hun Nederlandse positie ook Nederlandse vraagstellingen rechtvaardigt' (Calmthout 2002). Voorgaande waarschuwing geldt zeker voor mijn vak; er zijn mij geen typisch Nederlandse softwarefouten bekend. Als onderzoeker in dit veld voel je je soms als de man aan de rivier zoals die wordt geschetst in het kinderboekenweekgeschenk van 2002.

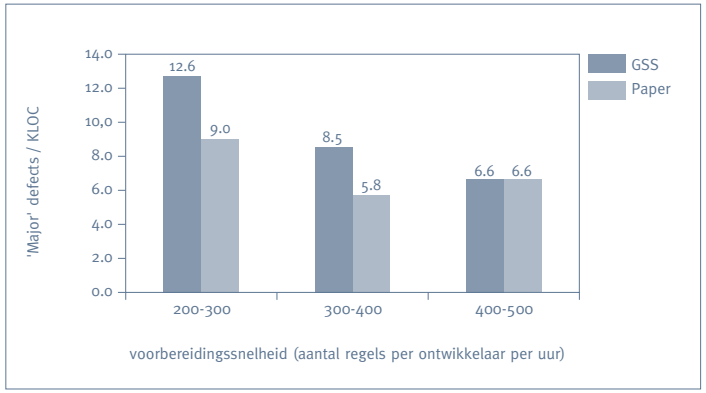
Een eindje voorbij het bord stond een man in het riet, een kleine, kale man. Hij had lange laarzen aan en stond met zijn armen over elkaar naar het water te turen.
Dunne Dirk zwaaide. 'Vreemde man, wat doe je daar?' riep hij.
'Ik vis,' antwoordde de vreemde man.
Tante Jo giechelde. 'Maar je hebt geen hengel, zo vang je niks'.
'Ik vis niet om te vangen, ik vis om te tellen' zei de vreemde man. 'Ik tel hoeveel vissen er in de rivier zwemmen en dan schrijf ik daar een boek over en dan word ik beroemd.'
'Leuk voor je', zei tante Jo.
'Ja. Dan schrijf ik: er zwemmen honderdzeven vissen in de rivier. En dan vertel ik het aan iedereen. En dan zeggen de mensen: 'O ja, is dat zo? Honderdzeven vissen? Dat wisten we niet. Wat knap van je dat je dat weet. We gaan meteen je boek kopen ...'
Boris vroeg: 'Hoeveel vissen zijn er hier?'
De vreemde man haalde zijn schouders op. 'Weet ik niet. Ik sta hier al dagen, maar ze zwemmen steeds weg en ze zwemmen steeds door elkaar heen. Ik moet steeds opnieuw beginnen met tellen. Zo komt mijn boek nooit af. Ik ben gek op boeken, ik heb er thuis wel tien. En ik heb ze allemaal gelezen. Allemaal!' (Kromhout 2002)

De gebruikte onderzoeksmethoden zullen afhankelijk zijn van de omstandigheden. In een aantal gevallen kan een bijdrage worden geleverd door precies te beschrijven wat wordt waargenomen, zoals de man doet met de vissen. Onderzoek kan ook ontwerpgericht zijn. Eerst iets ontwerpen of bedenken, en daarna in de praktijk toetsen of het werkt. Dit onder het motto: 'the best way to predict the future is to create it' (Alan Kay). Ik zie het als mijn taak in dialoog te zijn met de wetenschappelijke wereld door te blijven publiceren in toonaangevende, internationale tijdschriften. Tegelijk wil ik onderzoek doen dat relevant is voor de praktijk en uitgevoerd kan worden in dialoog met de praktijk. Onderzoek zal daarom in nauwe samenwerking met de praktijk gebeuren en veelal empirisch van aard zijn.
Het onderzoek zal uitgevoerd worden binnen de capaciteitsgroep I&T (Informatie & Technologie). Binnen de vakgroep ben ik in het gelukkige gezelschap van Rob Kusters en Jos Trienekens, twee ervaren onderzoekers op dit gebied. Ik verwacht tevens vruchtbare samenwerking binnen de groep I&T op het gebied van het modelleren van bedrijfsprocessen en architecturen. Samenwerking met de capaciteitsgroep product- en proceskwaliteit ligt eveneens voor de hand.

Graag geef ik vier voorbeelden van het soort onderzoek dat u kunt verwachten.
Ten eerste ondersteuning van inspecties met Group Support Systemen (GSS). Midden jaren 90 ontstond het idee dat inspecties wellicht beter ondersteund kunnen worden met Group Support Systemen (GSS), een vorm van software waarmee groepen worden ondersteund bij het uitvoeren van taken zoals vergaderingen (Nunamaker 1991). Dit idee is met succes in de praktijk gebracht door ontwikkelaars van Philips en Baan (Genuchten 1998). Hierna is het op grote schaal toegepast bij Baan (Genuchten 2001). Dit onderzoek maakte het mogelijk om de effectiviteit en efficiency van 102 traditionele en 87 GSS-inspecties te vergelijken. De resultaten staan in figuur 3.

figuur 3

GSS versus papieren inspecties (Genuchten 2001).



Twee conclusies uit dit onderzoek. In inspecties ondersteund door GSS worden 40 procent meer fouten gevonden dan in traditionele inspecties. En: dit voordeel blijkt snel te verdwijnen als het juiste proces niet meer wordt gevolgd. Een belangrijk deel van het inspectieproces is de voorbereiding die met de juiste snelheid dient te gebeuren. Als de voorbereiding sneller gaat dan 200-300 regels per uur, dan neemt de effectiviteit af, zoals te zien is in figuur 3. Dit leert ons weer eens, dat het automatiseren van een activiteit pas zin heeft als het proces volwassen is. Veel bouwers van softwarehulpmiddelen hebben deze les naar mijn mening nog onvoldoende geleerd. Dit doet vermoeden, dat empirisch onderzoek naar geavanceerde hulpmiddelen in onvolwassen softwareprocessen nog interessante resultaten kan opleveren.
Een tweede voorbeeld is onderzoek van Kusters en Trienekens (2002) naar de invloed van verbeteractiviteiten in software-ontwikkeling op de resultaten van het bedrijf. Het onderzoek is uitgevoerd bij een grote financiële organisatie. De resultaten geven aan, dat zowel de ontwikkelaars als de gebruikers overtuigd zijn van het nut van de verbeteractiviteiten. De waardering voor deze activiteiten neemt in de loop der jaren toe. De ontwikkelaars schatten het effect van de verbeteringen significant hoger in dan de gebruikers. Dit is opmerkelijk, omdat je zou verwachten dat de gebruikers het effect eerder opmerken.

Twee voorbeelden van reeds lopend onderzoek zijn een promotie-onderzoek naar software-ontwikkeling van productfamilies en onderzoek naar succesfactoren bij het verbeteren van software-ontwikkeling. Tot zover het ontwikkelen van software.

Kopiëren van software

Het kopiëren van software is zeer eenvoudig. De consequenties zijn verreikend en moeilijk te overzien. Duidelijk is dat de software-industrie in slechts 20 jaar een belangrijke industrie is geworden. Enkele indicaties: in 2001 was de wereldmarkt voor ‘packaged software’ zo’n 180 miljard dollar (Ghandi 2001). Het aantal softwarebedrijven met meer dan een miljard dollar omzet was in dat jaar gestegen van 10 naar 16 (Ghandi 2001). In deze telling worden alleen Independent Software Vendors (ISV’s) meegenomen. Dit zijn bedrijven die het merendeel van hun omzet uit software halen. Een bedrijf als IBM wordt in deze telling dus niet meegenomen, ondanks het feit dat dit een zeer grote softwareleverancier is. Een andere indicatie van het belang van software is dat op de lijst van 500 grootste bedrijven van de wereld 13 Independent Software Vendors staan (Fortune 2002). En dat voor een industrie die 30 jaar geleden amper bestond.

De veranderingen die software heeft gebracht in de computerindustrie en elektrotechnische industrie zijn in het midden van de jaren 90 beschreven door Ruud Gal en ondergetekende (Gal 1996). Inmiddels zijn we een aantal jaren verder. Het model is aangepast en zal nu worden gebruikt als illustratie van de gevolgen van software op industrieniveau. Het model schetst hoe de elektrotechnische industrie vijf fasen doorloopt. Het is geen technisch model, maar een economisch model van de ontwikkeling van softwaremarkten.

De eerste fase is de hardware-fase, waarin geen software in het elektrotechnische product zit. Een voorbeeld van zo’n product is een waterkoker. De tweede fase is die van embedded software. Een deel van de functionaliteit van het product wordt dan in de vorm van software gerealiseerd maar die software wordt als onderdeel van hardware verkocht. Een voorbeeld van een product in deze fase is een televisie of een kopieerapparaat. Software brengt in de embedded fase geen geld op, is dus voor algemeen management een kostenpost en wordt vaak ervaren als probleem.

De derde fase is die van de proprietary software: software wordt wel als afzonderlijk product verkocht, maar die software werkt alleen op hardware die door hetzelfde bedrijf wordt geleverd. Voorbeelden van producten in deze fase zijn professionele producten zoals medische scanners. De vierde fase is die van open systemen. Software van leverancier 1 werkt op hardware van leverancier 2, doordat er een standaard is ontstaan waaraan een belangrijk deel van de industrie zich houdt. Voorbeelden van producten in deze fase zijn het UNIX-platform sinds de jaren 90 en het DOS- / Windows-platform sinds midden jaren 80. Voor de klant is de keuze van hardware niet meer zo relevant. Voorbeeld: veranderen van PC-merk A naar merk B is minder ingrijpend dan een nieuwe tekstverwerker leren of een nieuw ERP-pakket implementeren. Invloed van softwarebedrijven in de opensystemen-fase is groot.

In 1996 eindigde hier ons model. Inmiddels komt er een vijfde fase in zicht: die van open source. Er ontstaat een beweging om software, inclusief source code, gratis ter beschikking te stellen aan iedereen die deze wil gebruiken. Het is begonnen als een filantropische hobby van een aantal ontwikkelaars. Open source werd beschouwd als een aardige manier om iets terug te doen voor de maatschappij (Applewhite 2003). Inmiddels draaien steeds meer computers op Open Source platform Linux. Het is een serieuze beweging waar onder meer een aantal grote leveranciers zich achter heeft gesteld. Deze bedrijven hopen geld te verdienen door afgeleide producten te verkopen of door services te leveren. Zo kiest Sony voor Open Source (Linux) als standaard platform voor zijn 'networked appliances strategy' (Lewis 2002). Inmiddels heeft een consortium van grote bedrijven zich achter LINUX CE gesteld. Tot deze bedrijven behoren onder meer Philips, Sony, Samsung en LG (Yoshida 2003).
De gevolgen zijn verreikend: consumentenelektronica is een zeer grote markt. Het verschijnen van een (wereldwijde) standaard maakt het voor softwarebedrijven mogelijk zich te richten op het ontwikkelen van toepassingen, zonder zich om details van de onderliggende hardware te hoeven bekommeren. Een samenvatting van het 5-fasen model staat in tabel 4.

tabel 4

Elektrotechnische industrie in beweging.

	Hardware	Embedded sw	Proprietary sw	Open systems sw	Open source sw
Voorbeeld van een product	Waterkoker	Televisie / GSM	Medische apparatuur	pc	Linux server
Drager van toegevoegde waarde	hardware	1 hardware 2 software	1 software 2 hardware	software	Service / andere producten
Prijs van software	Niet van toepassing	Gratis (per definitie)	oplopend	hoog	Terug naar nul?

Het is goed om ons twee dingen te realiseren. Ten eerste betreffen de veranderingen een zeer grote industriële sector. Ten tweede: de ontwikkelingen voltrekken zich in een zeer hoog tempo. Software bestaat pas sinds 1950 als embedded software. Proprietary software kennen we pas sinds het midden van de jaren 70, als open systems software sinds 1990. Vanaf 2000 dreigen sommige softwaremarkten alweer te verdwijnen, doordat software wellicht gratis wordt weggegeven in de vorm van open source.

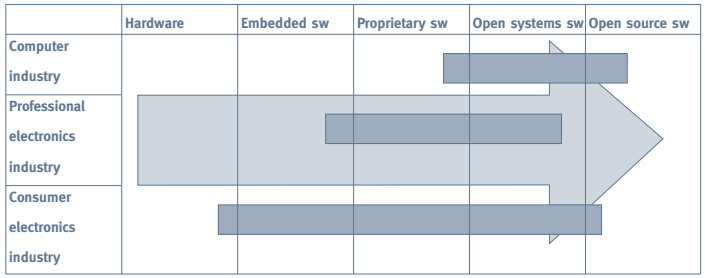
Wat betekent dit voor een bedrijf? In de embedded fase beperken de gevolgen zich tot de ontwikkelafdelingen. Bij de overgang van de embedded naar de proprietary fase verschijnt software voor het eerst op facturen en worden marketing en verkoop ermee geconfronteerd. Marketing- en salesregels veranderen. Tegelijk veranderen logistiek en fabricage. Tevens moet de strategie van het bedrijf worden aangepast aan de kenmerken van de software. In de fase van de open systemen is de software leidend geworden in de bedrijfsstrategie. Het is op dit moment onduidelijk, wat de gevolgen van open source zullen zijn voor bedrijven.

Klopt het hier geschetste model? In zijn huidige vorm is het model goed bruikbaar om veranderingen in de elektrotechnische industrie te beschrijven. De computerindustrie is bijna geheel aangeland in de open systems-fase en delen van de computerindustrie maken aanstalten om richting open source te gaan. Bedrijven die het belang van software hebben onderschat of nog steeds een eigen platform in de markt

hebben, zijn reeds verdwenen of zitten in moeilijkheden (Kerstetter 2002). Professionele elektronica bevindt zich in de proprietary en open systems-fase. Consumentenelektronica is zeer gespreid; producten als tv's zitten nog in de embedded fase. Microsoft probeert zijn invloed met open systemen uit te breiden naar de huiskamer (Greene 2002). Sony gaat een stap verder en zegt zijn strategie voor toekomstige producten te willen baseren op open source (Linux). De positie van de verschillende industrieën wordt samengevat in figuur 4.

figuur 4

Verschillende industrieën op weg van hardware naar software.



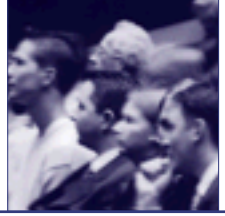
Het geschetste model heeft ook zijn tekortkomingen. Ik zal er twee noemen. Ten eerste: het model moet genuanceerd worden voor de verschillende industrieën. Zo zijn de auto- en telefonie-industrie erin geslaagd open systemen (voorlopig) buiten de deur te houden. Dit geeft aan, dat gevestigde industrieën wel degelijk de snelheid kunnen beïnvloeden waarmee fasen worden doorlopen. Ik verwacht niet, dat ze de gang door de beschreven fasen kunnen stoppen. Duidelijk is, dat de gevestigde software-industrie zich niet zomaar zal neerleggen bij het feit dat een deel van de software in de toekomst wordt weggegeven. Inmiddels heeft een groep van 13.000 softwarebedrijven in de VS zich al verenigd met het doel gebruik van open source binnen de Amerikaanse overheid tegen te houden (Levy 2003).
Een tweede tekortkoming is dat het model kwalitatief is. Onduidelijk blijft onder andere wat de gevolgen voor een bedrijf of industrie zijn, als de overgang van hardware naar software verkeerd wordt ingeschat. Gevolgen voor bedrijven en markten zijn moeilijk te specificeren. Hierdoor is het management van bedrijven in transitie moeilijk te overtuigen van de noodzaak tot actie. De relevantie van de

industrie en het tempo van veranderingen maken dit een interessant onderzoeksonderwerp voor onze faculteit Technologie Management. Het leidt tot de volgende onderzoeksvraag:

Onderzoeksvraag 2:
Met welke modellen kan ontwikkeling van softwaremarkten overtuigend worden beschreven en voorspeld?

De focus van mijn onderzoek zal liggen op de implicaties van software op ontwikkel- en bedrijfsniveau. Toch wil ik hier ook een aantal gevolgen van de geschetste ontwikkelingen noemen op het niveau van nationale en internationale economie. Software is een zeer interessant product voor een ontwikkelde en dichtbevolkte regio als West-Europa; ruimte- en energiebeslag zijn immers nihil. Verder is het mogelijk met een beperkte groep mensen veel toegevoegde waarde te genereren. Immers: reproductie van additionele producten is gratis en men kan blijven kopiëren en incasseren.
Europa en in het bijzonder Nederland hebben echter een zwakke positie op het terrein van software. Europa is netto importeur van softwareproducten. Europa importeert voor 155 miljard Euro en exporteert voor 117 miljard aan ICT-producten (ISTAG 2002). Het tekort op deze betalingsbalans is 38 miljard dollar. Het mag duidelijk zijn, dat dit tekort niet te compenseren is door services aan te bieden. Als we ervan uitgaan, dat 'een mensjaar ICT services' ongeveer 150.000 Euro waard is, betekent dit dat Europa meer dan 250.000 mensjaren services moet exporteren om betalingsevenwicht te bereiken. En dat in concurrentie met talenten en tarieven die landen als India en China bieden.

Wat Nederland betreft het volgende: het aandeel van de ICT-goederen in de totale import van Nederland is gestegen van 12 naar 19 procent tussen 1995 en 2000 (CBS 2002). Tegelijk is de groei in de uitvoer van ICT-producten (29 procent) achtergebleven bij de totale groei van de Nederlandse export (36 procent). Het aandeel van ICT aan de toegevoegde waarde van de bedrijvensector ligt onder de 2 procent, ver onder die van landen als de Verenigde Staten, het Verenigd Koninkrijk, Ierland en Finland. Nederland is netto importeur van softwareproducten (CBS 2002). Dat is niet verwonderlijk, aangezien Nederland een zeer beperkte software-industrie heeft. In de top 100 van ICT-bedrijven



staat de eerste Nederlandse softwareleverancier op plaats 42 (www.Computable.nl) en dat is Baan, inmiddels onderdeel van een Brits concern. De volgende Nederlandse softwareleverancier is Exact, op plaats 56.

In Nederland hebben servicebedrijven steeds een groot deel van de markt bepaald. Software servicebedrijven verkopen hun kennis in de vorm van uren of services maar niet in de vorm van software. Nadeel hiervan is, eenvoudig gesteld, dat uren niet te reproduceren zijn. Dit leidt ertoe, dat Nederland nauwelijks software services exporteert. Sterker nog, sinds 2001 is Nederland zelfs netto importeur van software services (CBS 2002). Nederland en Europa laten vooralsnog de kans voorbijgaan op een zeer interessante industrie voor dit ontwikkelde en dichtbevolkte deel van de wereld. In Nederland zien we dat de industriële productie afneemt. Een aantal partijen schijnt de indruk te hebben dat als er industrie verdwijnt, er vanzelf een kenniseconomie verschijnt. Dit is een illusie. Elkaar achter de dijken adviseren en uren verkopen leidt niet tot toegevoegde waarde, laat staan tot exporteerbare kennisproducten.

Wat kan Europa doen om een groter deel van deze interessante industrie te veroveren? Men dient te beseffen, dat de software-industrie een concurrerende industrie is. De Verenigde Staten zijn nummer 1 en koesteren die positie. Een land als China heeft software als hoogste prioriteit op zijn technologie-agenda gezet (Einhorn 2002). Twee veranderingen bieden Europa een kans: de opkomst van open source en het feit dat het belang van software toeneemt in industrieën waarin Europa van oudsher een sterke positie heeft.

Over open source: in open systemen leek het spel gespeeld met Amerikaanse softwarebedrijven als grote winnaars. Open source betekent, met alle onzekerheden, een nieuwe ronde met nieuwe kansen. Het businessmodel voor open source is nog onduidelijk, maar de markt voor betaalde applicaties op open source platforms is een nieuwe markt waar nieuwe spelers kansen hebben.
Een duidelijke kans ligt in industrieën waarin Europa sterk is en waarin het softwarespel nog gespeeld moet worden. In industrieën zoals de auto-industrie, medische elektronica en telecommunicatie worden open standaarden gezet. Dit zijn grote industrieën. Zo wordt de markt voor healthcare IT geschat op 21 miljard dollar in 2002 (Ragupathi 2002).

Het is daarbij duidelijk, dat software een groot deel van de toegevoegde waarde in healthcare IT zal vormen. Europa heeft de kans om via deze industrieën een rol in software te gaan spelen. Het is daarom noodzakelijk een deel van de kennis die in Europa beschikbaar is om te zetten in software en deze kennis in de vorm van producten te verkopen. Het is belangrijk voor Europese bedrijven en economieën om het in deze industrieën beter te doen dan eerder gebeurde in de computerindustrie. Alleen zo kunnen interessante softwarebedrijven, -banen en -inkomsten worden gegenereerd.
Het belang is duidelijk: als Europa opnieuw de boot mist, dan zal het handelstekort op ICT-gebied drastisch groeien. Ik blijf optimistisch, omdat applicatiekennis en vaardigheid om deze kennis om te zetten in software in Europa aanwezig is.

Onderwijs in software management

Ik hoop u er inmiddels van overtuigd te hebben, dat software management een relevant en interessant vak is. De vraag is hoe je dit vak kunt onderwijzen aan studenten technische bedrijfskunde.
Sinds 1998 ben ik als gastdocent betrokken bij het keuzevak software management. Het derde deel van mijn rede beschrijft de ervaringen in dat onderwijs van 1998 tot 2002 en de plannen voor de komende jaren.

Studenten interesseren voor een vak en het vak vervolgens goed geven wordt steeds belangrijker vanwege de te verwachten concurrentieslag om studenten tussen universiteiten en hogescholen. Bachelor-studenten zullen hun keuze naar verwachting nog grotendeels bepalen door geografische overwegingen en dicht bij huis studeren. Na hun bachelorexamen maken studenten opnieuw een keuze. Dit betekent, dat we niet meer alleen op de 18-jarige vwo-leerling moeten inzetten maar ook op andere doelgroepen. Daarbij draait het om de vraag of we studenten met drie jaar studie-ervaring kunnen houden, dan wel of we buitenlandse studenten met drie jaar universitaire opleiding kunnen krijgen.

Voor docenten is het van belang vast te stellen, dat het door de nieuwe mogelijkheden in het onderwijs interessanter wordt. Het leidt tot verandering van werk. Inhoud is veel eenvoudiger te verkrijgen en beschikbaar te stellen. Via Internet is een overvloed aan informatie beschikbaar. Een voorbeeld is het Open Course Ware- (OCW-)initiatief van MIT. Hierin stelt MIT al zijn 2000 cursussen gratis en on-line beschikbaar (www.ocw.mit.edu). Het is mijn ervaring, dat een willekeurige groep studenten binnen een week dingen over mijn vak weet die ik nog niet weet. Ik vind dat geweldig, want dat betekent dat ik ook iets leer. Dit wil niet zeggen, dat de docent overbodig is. De docent speelt een rol bij het aanbieden van structuur om de overdaad aan informatie te kunnen ordenen. De docent verandert van ‘the sage on the stage’ in ‘the guide by the side’.

Een andere, nieuwe mogelijkheid is collaboratief leren. Afgelopen jaar ben ik betrokken geweest bij ‘LEX, the learning experience’ van de Open Universiteit Nederland (www.lex.ou.nl). Samen met collega Doug Vogel van de City University in Hong Kong heb ik het onderwerp ‘collaboratief leren’ mogen vormgeven. In de praktijk van collaboratief leren is de docent lid van het team en hoeft hij dus niet al het werk alleen te doen. Collaboratief leren blijkt in bepaalde omstandigheden ook nog eens beter, sneller en leuker. Een voorbeeld is leren in een team over Italiaanse schilderkunst in een museum (www.lex.ou.nl). Een interessant voorbeeld op mijn vakgebied is het vak zoals dit door prof. Brinkkemper wordt gegeven aan de VU. Studenten informatica ontwikkelen niet alleen een softwareproduct maar ook het bedrijf dat daarbij hoort (www.netherware.nl).

Hoe hebben we de laatste jaren de veranderingen aangegrepen om onderwijs in software management aan studenten Technologie Management te verbeteren? Gegeven is dat genoemde studenten geen ervaring hebben met het ontwikkelen van software. In het vak software management streven we ernaar studenten te laten ervaren wat werken in een groot softwareproject betekent. Studenten bestuderen een onderwerp in een internationaal samengesteld, virtueel team van acht tot tien studenten. Het resultaat dient te worden gepubliceerd in de vorm van een website. Websites van 17 groepen van de leergang 2002 vormen samen een elektronisch boek dat te vinden is op http://hknet.tm.tue.nl .

figuur 5

Foto van virtueel team in actie.

De studenten komen uit verschillende landen en hebben elkaar nog nooit ontmoet in de reële wereld, maar wel in de virtuele wereld via

Internet en in videoconferenties. Taak is het onderwerp te analyseren en te beschrijven vanuit verschillende culturele, geografische en disciplinaire invalshoeken. Door in virtuele teams te werken leren studenten op verschillende terreinen:

- Ze ervaren wat het betekent om in een project te werken en een werkend product op te leveren, in dit geval een website. Dit geeft onmiddellijk de dynamiek die je ook terugvindt in een software-project: een groot team opgesplitst in kleine teams, die geografisch gespreid aan onderdelen van het grote geheel werken. En net als in de echte wereld, wordt ook hier gewerkt onder tijdsdruk.
- Ze leren over het software management-onderwerp dat ze bestuderen. Dit onderwerp kan bijvoorbeeld zijn: softwarefouten of ERP-implementaties.
- De studenten leren samenwerken in een virtueel team (Vogel 2001, Rutkowski 2002)
- Ze ervaren wat het is samen te werken met andere culturen. Dit is niet alleen van belang voor hun opleiding maar ook voor hun functioneren in hun carrière en de maatschappij. Studenten blijken niet alleen te leren over de cultuur van de teamleden maar ook over hun eigen cultuur (Rutkowski 2001).

Hoe gaan we verder met software management? Software management omvat onder meer zorgvuldig plannen, specificeren en implementeren. Dit zijn onderdelen die virtuele teams zullen uitvoeren. Een college over plannen gaat vooraf aan het zelf daadwerkelijk plannen van een deel van het softwareproject. Voor de student is het dus eerst leren en dan meteen doen. Het realisme van het project zal verder toenemen, omdat we het elektronische boek van jaar tot jaar verbeteren en uitbreiden. Aldus zullen studenten ervaren, wat het nut van documentatie en configuratiemanagement is. Een kwartier zoeken naar een document laat beter het nut van configuratie management zien, dan een college van een uur.

Tegelijk leren wij zelf meer over het functioneren van virtuele teams. Deze kennis is toepasbaar in virtuele teams in onderwijs maar ook daarbuiten. Het HKNet-project is één van de grootste virtuele teamprojecten in zijn soort. Er is ervaring met grote virtuele teams die analytische activiteiten ondernemen (Arkesteijn 2003). Bijzonder aan HKNet is dat een virtueel team van deze omvang bouwen of construeren tot taak heeft.

Dit leidt tot de derde onderzoeksvraag waar ik me op zal richten.
Onderzoeksvraag 3:
Hoe kunnen virtuele teams, in het bijzonder in onderwijs, effectief met ICT worden ondersteund?

Op dit terrein is samenwerking met de capaciteitsgroep Human Performance Management gestart. Bovendien loopt er inmiddels een promotie-onderzoek dat zich richt op de verbetering van onderwijsprocessen en ondersteunende software. Met het laatste punt is de cirkel weer rond, want software voor e-learning is in ontwikkeling en behoeft nog veel verbetering. Met een dergelijk betoog ben ik mijn verhaal begonnen.

Tot slot

Software is een mooi vakgebied waarin van alles gebeurt. Werken aan software management aan deze universiteit geeft mij de kans een maatschappelijke bijdrage te leveren via interessant onderwijs en onderzoek. Ik waardeer het zeer, dat ik deze kans krijg en ik wil daarvoor een aantal mensen van harte bedanken.

Ik dank het College van Bestuur en het bestuur van de faculteit Technologie Management voor het in mij gestelde vertrouwen. Ik bedank de studenten voor het enthousiasme in de colleges en het begrip als de verbinding met Hong Kong even niet tot stand komt. Studenten, ik verwacht dat jullie respect hebben voor echt werk in echte bedrijven. Een snelle analyse en een oppervlakkig advies is niet het doel van de studie. Het zal in mijn vak geen voldoende opleveren, niet op het tentamen en niet in de praktijk. Mijn doel is jullie te laten ervaren dat echt werk veel interessanter is, zowel in het onderwijs als in de praktijk.

De vakgroep informatie en technologie wil ik bedanken in de personen van Theo Bemelmans, Hans Wortmann en Wil van der Aalst. Van Theo en Hans heb ik veel geleerd tijdens mijn promotie en het is mooi de samenwerking te kunnen voortzetten. Ik ben 11 jaar weggeweest en er is in die tijd veel gebeurd. Veel waardering heb ik voor de prestaties van de groep in de laatste jaren, onder leiding van Wil van der Aalst. Het was een goede groep en het is een goede groep om in terug te keren.

I would like to thank the HKNet team for the experience of the last years. The team includes the students as well as the instructors Doug Vogel, Ann Rutkowski and Theo Bemelmans. In 1998 we had no idea what we started and if we are honest, we still do not know where we are going. We are working as friends and we are providing innovative education as well as interesting research. And all that is considered work.

Ik wil graag mijn ouders bedanken. Veel van wat belangrijk is, heb ik van jullie geleerd. Helaas is mijn vader twee jaar geleden overleden. Wij missen hem nog iedere dag. Ik wil vanaf deze plek ook Anja, Wouter en Guus bedanken. Werken is heel aardig, maar thuis is het echt leuk. Mijn mededeling dat ik een dag in de week op de TU/e zou gaan werken, leverde me thuis vijf minuten aandacht op en de vraag of een professor op de TU/e net zoiets is als een professor op Zweinstein. Ik heb dat bevestigend beantwoord met de toevoeging dat ik het toveren nog moet leren.

Tot slot wil ik u allen bedanken voor uw aandacht en aanwezigheid. Ik ben onder de indruk van het gezelschap dat zich hier vanmiddag heeft verzameld.

Referenties

1 Aerts, H., Mogilensky, J. 'Evolutionary stages of software growth', European SEPG 2000, 5 - 8 June 2000 in Amsterdam

2 Applewhite, A., 'Since you asked...', IEEE Spectrum, January 2003.

3 Arkesteijn, H., De Rooij, J., Eekhout, M., Van Genuchten, M., 'A large-scale virtual meeting in practice', te publiceren in Group Decision and Negotiation Support, 2003

4 Boehm, B., Basili, V.R., 'Software defect reduction top 10 list', IEEE Computer, January 2001

5 Brooks, F.P., 'No Silver Bullet: Essence and Accidents of Software Engineering'. IEEE Computer 20 (1987)

6 Calmthout, Scholtens, 'Ramen open, interview met Nijkamp, voorzitter NWO', Volkskrant, 6 april 2002.

7 CBS, Centraal Bureau voor de Statistiek, 'De digitale economie 2002', Voorburg/Heerlen 2002.

8 Dunham, J.R., Kruesi, E., 'The measurement task area', IEEE Computer, November 1983.

9 Einhorn, 'High Tech in China', Business week, October, 28, 2002.

10 Fagan, M. 'Advances in software inspections', IEEE Transaction on Software Engineering, 12(7), July 1986, pp. 741-755.

11 Feder, B., 'The fade-out of the light bulb; microchips burn cheaply, and more colorfully', New York Times, February, 12, 2003.

12 Gal, R., van Genuchten, M., 'Release the embedded software; the electronics industry in transition', International Journal on Technology Management, June 1996.

13 Genuchten, M. van, 'Why is software late? An empirical study of reasons for delay in software development', IEEE transactions on Software Engineering, 1991

14 Genuchten, M. van, van Dijk, C., Scholten, H., Vogel, D., 'Using Group Support Systems for software inspections', IEEE Software, May-June 2001, page 60-65

15 Genuchten, M. van, Cornelissen, W., and van Dijk, C., 'Supporting Inspections with an Electronic Meeting System', Journal of Management Information Systems, Winter 1997-98/Volume 14, No.3

16 Ghandi, P, Joseph, J., Seaberg, J., 'Beyond the billion dollar myth: a predictive view of the software industry', McKinsey company, November 2001.

17 Greene, Microsoft in the living room, Business week, xxxx

18 Haley, T.J., Software process improvement at Raytheon, IEEE Software, November/December 1997, Page(s): 33-41

19 Harter, D.E., Krishnan, M.S., Slaughter, S.A. , 'Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development' Management Science, April 2000.

20 Heemstra, F.J., Hoe duur is programmatuur?, Kluwer 1989.

21 Holson, L., 'As animation goes digital, Disney frets, New York Times, February, 12, 2003.

22 http://hknet.tm.tue.nl resultaten van het HKNet 5 project

23 http://www.computable.nl grootste Nederlandse ICT bedrijven

24 http://www.fortune.com Fortune 500 van grootste bedrijven

25 http://www.google.com toegang tot de rest van de wereld

26 http://www.netherware.nl resultaten college Brinkkemper aan de VU

27 http://www.lex.ou.nl Learning experience met ondermeer collaboratief leren

28 http://www.ocw.mit.edu Cursussen van het MIT

29 http://www.tm.tue.nl/beta onderzoekschool BETA

30 http://www.tm.tue.nl/ecis onderzoekschool ECIS

31 http://www.tm.tue.nl/it de vakgroep Information & Technology

32 http://www.tm.tue.nl/it/staff/mvgenuchten personal website Michiel van Genuchten

33 Humphrey, W.S., 'Winning with software: an executive strategy, Addison-Wesley, Boston, 2002.

34 Humphrey, W.S., Managing the software process, Reading MA: Addison Wesley, 1989

35 ISTAG, 'Software Technologies, embedded systems and distributed systems' IST Advisory Group, to be found at http://www.compas.eu

36 Kerstetter, J., Greene, J., 'Will Sun rise again?', Business week November, 25, 2002

37 Kromhout, R., Boris en het woeste water, Stichting collectieve propaganda van het Nederlandse boek, 2002.

38 Kusters, R.J. en Trienekens, J.J.M., and Hassoldt, W., On the business impact of software process improvement, In: Martin, D.C., Proceedings of the IEEE 26th international computer software & applications conference, IEEE Computing Society, Los Alamitos, 2002, ISBN 0-7695-1727-7, pp. 59-67.

39 Levy, S., forcing open Windows, Newsweek, January, 6, 2003

40 Lewis, P., 'Sony redreams its future', Fortune, November, 25, 2002

41 McGarry, F., Decker, B., Attaining level 5 in CMM process maturity, IEEE Software, November-December 2002.

42 Nunamaker, J.; Dennis, A,; Valacich, J.; Vogel, D.; and George, J. Electronic meeting to support group work, Communications of the ACM, 34(7), July 1991, pp. 40-61.

43 Raghupathi, W., Tan, J., Strategic IT applications in health care, Communications of the ACM, December 2002.

44 Reifer, D. (editor), 'Software management', IEEE Computer Society Press, 1993.

45 Rutkowski, Vogel, D., van Genuchten, M., Bemelmans, T., 'E-collaboration, the reality of virtuality', IEEE Transactions on Professional Communications, December 2002.

46 Rutkowski, A., Vogel, D., Bemelmans, T., Genuchten M. van, 'Group Support Systems and virtual collaboration, the HKNet project', Journal of Group Decision and Negotiation, No. 2, 2002.

47 Rooijmans, J., Aerts, H., Genuchten, M. van, 'Software quality in Consumer electronic products', IEEE Software, January 1996.

48 Varian, H., 'Economics of Information Technology', paper presented at Boccomi university in Milano, November 2001

49 Vogel, D., Genuchten, M. van, Lou, D., VerveenS., Eekhout, M. van, Adams, T., 'Exploratory research on the role of national and professional cultures in a distributed learning project', IEEE Transactions on Professional Communications, June 2001, page 114-125.

50 Vu, J., 'Software process imporvement journey (from level to level 5)', Software Engineering Process Group Meting, 1997.

51 Weber, M., Weisbrod, J., 'Requirements engineering in automotive development: experiences and challenges', IEEE Software, January-February 2003.

52 Yoshida, Junko, 'Japanese Manufacturers Back Off Proprietary OSes', EE Times, January 2003.

Curriculum Vitae

Aan de Technische Universiteit Eindhoven is per 1 augustus 2002 prof.dr.ir. M.J.I.M. van Genuchten benoemd als parttime hoogleraar software management bij de capaciteitsgroep Informatie & Technologie van de faculteit Technologie Management.

Michiel van Genuchten (1963) studeerde in 1987 af aan de Technische Universiteit Eindhoven. In 1991 promoveerde hij op het proefschrift 'Towards a software factory' aan diezelfde universiteit.
Sinds 1987 is hij werkzaam in het bedrijfsleven, onder meer bij Philips en Groupsupport, een softwarebedrijf dat hij midden jaren 90 heeft opgericht.
Zijn onderzoeksinteresse gaat uit naar de ontwikkeling van softwareproducten en softwaremarkten, alsmede naar ondersteuning van (virtuele) teams met informatietechnologie.

Colofon

Productie:
Communicatie Service
Centrum TU/e

Fotografie cover:
Rob Stork, Eindhoven

Ontwerp:
Plaza ontwerpers,
Eindhoven

Druk:
Drukkerij Lecturis,
Eindhoven

ISBN: 90-386-1053-X