

Op weg naar samenwerking : van communicatie en synchronisatie naar teamwork

Citation for published version (APA):

Hammer, D. K. (1993). *Op weg naar samenwerking : van communicatie en synchronisatie naar teamwork*. Technische Universiteit Eindhoven.

Document status and date:

Gepubliceerd: 01/01/1993

Document Version:

Uitgevers PDF, ook bekend als Version of Record

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Op weg naar
samenwerking:
van communicatie
en synchronisatie
naar teamwork

INTREEREDE

Prof.dr.dipl.ing. D.K. Hammer



Technische Universiteit Eindhoven

INTREEREDE

In verkorte vorm uitgesproken op
10 september 1993, ter gelegen-
heid van het aanvaarden van het
ambt van hoogleraar Technische
Toepassingen van de Informatica
aan de Technische Universiteit
Eindhoven

Prof.dr.dipl.ing. D.K. Hammer

Mijnheer de Rector Magnificus,
Dames en heren,

productie- en distributiesysteem voor goederen, in de internationale samenwerking op wetenschappelijk gebied en in internationale hulpacties bij catastrofe of oorlog.

Inleiding

Wij leven in een spannend tijdperk waarin ontzettend veel gebeurt; dat is althans het gevoel dat wij hebben. Vermoedelijk is in andere tijdperken net zo veel gebeurd, alleen waren de mensen zich daarvan niet zo bewust. Dat heeft twee oorzaken:

1. De moderne telecommunicatietechniek maakt het ons mogelijk om wereldwijd van praktisch alle gebeurtenissen op de hoogte te zijn en deze met elkaar uit te wisselen: wij krijgen het wereldnieuws via de zogenaamde broadcast-media, zoals radio en televisie, direct in huis, wij kunnen langzamerhand vanaf ieder punt op aarde met iedereen telefoneren, te land, ter zee en in de lucht, en wij gebruiken continent-overspannende computernetwerken om alle mogelijke gegevens uit te wisselen.
2. Wij zijn veel meer betrokken bij het gebeuren om ons heen omdat wij op een globale schaal met elkaar *samenwerken* en van elkaar afhankelijk zijn. Dit uit zich bijvoorbeeld in de complexe (en vaak ondoorzichtige) samenhangen van onze wereldeconomie, in een wereldwijd

Hier dient zich onmiddellijk de vraag aan: heeft de moderne techniek de wereldeconomie mogelijk gemaakt of is de verruiming van ons bewustzijn ten opzichte van onze omgeving de stimulus geweest om ook de technische hulpmiddelen te scheppen die dit avontuur mogelijk maakten? Mijn stelling is, dat de technische ontwikkeling altijd een gevolg is van de heersende opvattingen. Ik zal dit in het tweede gedeelte van mijn betoog nog toelichten.

Allereerst wil ik u graag meenemen op een ontdekkingsreis door de wereld van de samenwerkende systemen. Ik zal bij mijn eigenlijke vakgebied beginnen, namelijk bij het ontwerpen van *technische computersystemen* en het maken van de bijbehorende software. Daarbij wil ik wel duidelijk stellen dat technische computersystemen geen doel op zich zijn. Hun bestaan wordt gerechtvaardigd door de noodzaak om een bepaald proces efficiënt te ondersteunen. Een productiebesturingssysteem ondersteunt bijvoorbeeld het maken van een produkt. Maar ook televisietoestellen, kopieermachines, telefooncentrales en vliegtuigen worden door complexe computersystemen bestuurd. Het ondersteunde proces is dan het

gebruik van deze machines, d.w.z. het televisiekijken, het kopiëren, het telefoneren of het besturen van een vliegtuig. Omdat al deze computersystemen voor het oog van de gebruiker verborgen zijn, wordt deze klasse van computersystemen vaak ook *embedded systems* genoemd; de bijbehorende software heet dan *embedded software*.

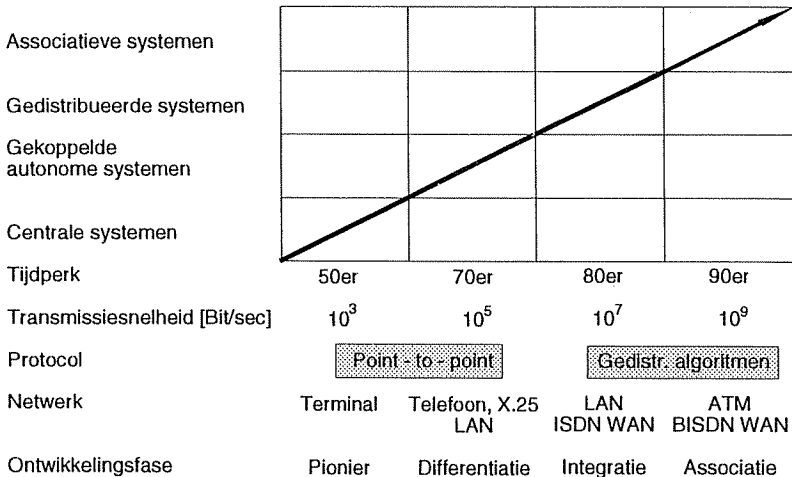
Om een proces effectief te kunnen ondersteunen moet men eerst goed kijken naar de activiteiten, waaruit dat proces bestaat. Dat lijkt triviaal. Niettemin gebeurt het nog steeds dat de constructeurs van een computersysteem, d.w.z. systeemarchitecten en software ontwerpers, zich meer door de technische mogelijkheden

laten leiden dan door de noodzakelijkheden van het proces en de behoeften van de gebruikers. In het tweede gedeelte van mijn rede wil ik daarom ook ingaan op de samenwerking tussen organisaties en informatiesystemen, c.q. het afleiden van een informatiesysteem uit het te ondersteunen proces. Hoewel de hier bedoelde methoden nog in hun kinderschoenen staan zullen zij in de toekomst net zo belangrijk worden als het afleiden van een programma uit zijn specificatie.

Samenwerkende computersystemen

Telecommunicatie is de techniek van

VAN COMMUNICATIE NAAR SAMENWERKING: GEDISTRIBUEERDE SYSTEMEN



Afbeelding 1 De historische ontwikkeling van de samenwerking tussen computersystemen.

het uitwisselen van gegevens op afstand. Laten we eerst eens kijken naar de ontwikkeling van de telecommunicatie en met name naar systemen waar (in tegenstelling tot radio en televisie) de informatie-uitwisseling wederzijds is. Als het om spraak gaat communiceren wij in het algemeen per telefoon, berichten versturen wij per telex of sinds kort ook per electronic mail en computers wisselen gegevens uit via speciale datacommunicatie-netwerken. De ontwikkeling van dit vakgebied maakt deel uit van een veel algemenere ontwikkeling, namelijk het samenwerken om een gemeenschappelijk doel te bereiken. Deze ontwikkeling is in afbeelding 1 in kaart gebracht.

Gecentraliseerde systemen

De historie van de telecommunicatie begint ergens in de jaren '50 met grote gecentraliseerde computersystemen, ook mainframes genoemd, waarmee de gebruikers via terminals konden communiceren. Omdat het om relatief langzaam interactief werk ging was een simpel netwerk met lage transmissiesnelheid en eenvoudige protocollen voldoende. Meestal had zo 'n netwerk de vorm van een ster met de computer als middelpunt.

Gekoppelde autonome systemen

In de loop van de ontwikkeling zijn autonome gecentraliseerde systemen voor alle mogelijke taken ingezet, van de administratie tot en met de produktiebesturing. Op deze

manier ontstond een grote hoeveelheid van zogenoemde *automatiseringseilanden*. Omdat een bedrijf een samenhangend geheel is vroeg deze toestand natuurlijk om koppeling van de verschillende systemen ten behoeve van het uitwisselen van gegevens. Op deze manier ontstond vanaf de jaren '70 de trend naar gekoppelde autonome systemen.

De koppeling van systemen vond buitenshuis meestal plaats via de openbare telefoon WAN (Wide Area Network) en binnenshuis via verschillende typen van LAN's (Local Area Networks). De bandbreedtes waren nog vrij laag en de protocollen beperkten zich op de onderste lagen van het vanaf 1978 geleidelijk geïntroduceerde OSI (Open Systems Interconnection) model. Deze manier van werken had twee grote nadelen:

1. Het maken van zo'n koppeling kostte veel tijd en geld omdat de systemen nog leveranciersspecifiek ofwel gesloten waren; zo'n *interface* is in feite een brug tussen twee verschillende werelden. Op een gegeven ogenblik kon het zelfs zo ver komen dat een grote automobielfabrikant als General Motors meer geld uit moest trekken voor het maken van interfaces dan voor het kopen van computersystemen. En dat kon natuurlijk nooit de bedoeling zijn.
2. De koppeling vond plaats op het niveau van de hardware en de systeem-software, maar niet op

het niveau van de applicaties. Dat laatste is wel de bedoeling omdat het, zoals reeds in de inleiding vermeld, niet om de koppeling van computersystemen maar om de ondersteuning van een gemeenschappelijk proces gaat. Met andere woorden: het probleem van de koppeling van autonome systemen werd te veel vanuit de technische kant en te weinig vanuit de gebruikerskant bekeken.

Gedistribueerde Systemen

Uitgaande van de twee bovengenoemde problemen voltrekt zich de ontwikkeling van gedistribueerde systemen in de loop van de jaren '80 in twee fases, die tot twee typen van gedistribueerde systemen leiden:

Gedistribueerde processing systemen.

- Ook vandaag de dag zijn wij nog steeds bezig met het verbinden van autonome systemen, alleen nu meer tussen bedrijven (bijvoorbeeld door middel van EDI) dan binnen een bedrijf. Dankzij de ontwikkeling van een algemeen bruikbare communicatie-architectuur inclusief de bijbehorende standaardprotocollen, in de vorm van het boven genoemde *OSI-Model*, is het nu mogelijk systemen van heel verschillend type met elkaar te koppelen. Daardoor ontstaan er langzamerhand geavanceerde netwerksystemen die ik gedistribueerde processing-systemen noem. Zij zijn gebaseerd

op een aantal technische ontwikkelingen.

Ten eerste staan er tegenwoordig veel krachtigere LAN's en WAN's ter beschikking die een veel grotere bandbreedte en veel meer services hebben. Bij de laatste hoort bijvoorbeeld het X.25-datanetwerk, het ISDN (Integrated Services Digital Network) netwerk en in de komende jaren ook de MAN (Metropolitan Area Network) netwerken. Maar nog belangrijker is dat langzamerhand ook de hogere lagen van het OSI-Model met standaardprotocollen voor alle mogelijke toepassingen ingevuld zijn. Dit heeft tot gevolg dat applicatie-programma's gebruik kunnen maken van hoog niveau services en zich niet meer hoeven te bekommeren om de details van het communicatienetwerk en het systeem waarmee uitwisseling plaats vindt. Deze stap werd mede mogelijk gemaakt door krachtige computersystemen die de intussen tamelijk complex geworden protocollen in redelijke tijd af kunnen werken.


De belangrijkste kenmerken van gedistribueerde processingsystemen kunt U in het rechter-gedeelte van afbeelding 2 vinden.

Gedistribueerde controlesystemen.

Ondanks deze technische vooruitgang bleef het tweede probleem bestaan. De volgende fase begint pas in de tweede helft van de jaren '80 en is nog steeds in volle

TYPEN VAN GEDISTRIBUEERDE SYSTEMEN

	Gedistr. controle syst.	Gedistr. processing syst.
Taak	Gemeenschappelijk	Niet gerelateerd
Toegankelijkheid	Gesloten	Open
Afhankelijkheid	Hoog	Matig
Systeem samenhang	Gedistribueerde algoritmen	Communicatie
Synchronisatie	Globaal	Lokaal
Focus	Coördinatie Transparantie	Flexibiliteit Openheid



Associatieve systemen

Afbeelding 2 De twee basistypen van gedistribueerde systemen.

ontwikkeling. Het gaat nu om de koppeling van systemen op het niveau van de applicatie semantiek. Dit leidt tot de ontwikkeling van wat ik gedistribueerde controlesystemen noem en waarvan ik de kenmerken in de linkerhelft van afbeelding 2 heb vermeld. Om een bepaald proces goed te kunnen ondersteunen, streeft men nu naar coördinatie van de verschillende deeltaken en naar transparantie (dat wil zeggen naar het verbergen van niet ter zake doende details) en niet alleen maar naar connectiviteit en transmissiesnelheid. In technische termen betekent dit, dat men naar een bepaalde partiële ordening van alle gebeurtenissen in het systeem streeft.

Deze coördinatie bereikt men door

middel van *gedistribueerde algoritmen* die voor de synchronisatie [Hammer '93a] tussen de verschillende partijen zorgen. Deze zijn een veralgemenisering van de eerder genoemde OSI-protocollen die, althans voor de onderste lagen, als gedistribueerde algoritmen met slechts twee communicatiepartners gezien kunnen worden.

Met andere woorden, het gaat nu echt om teamwork tussen meerdere partijen om een gemeenschappelijk doel te bereiken. Weliswaar zijn de partijen in dit geval technische entiteiten, zoals concurrerende processen of objecten, maar het principe is hetzelfde dat wij ook in moderne organisatievormen zoals Lean Enterprise ([Womack et al. '90] en [Hammer '92]) terug kunnen vinden; maar daarover straks meer. De uitdaging op dit gebied ligt in de

ontwikkeling van algoritmen die niet alleen efficiënt zijn op zichzelf, maar ook bestendig zijn tegen fouten en compatibel met andere algoritmen [Le Lann '92]. Het laatste is niet vanzelfsprekend als men bedenkt dat in een modern gedistribueerd systeem een aantal algoritmen voor verschillende doeleinden naast elkaar moet bestaan. Tot nu toe worden deze algoritmen alleen met het oog op een van deze doeleinden ontworpen en niet als coöpererend ensemble. Dit bergt het gevaar in zich dat zij niet echt samenwerken of elkaar zelfs tegen werken. Een typisch voorbeeld voor zo'n suboptimaal ontwerp vormen concurrency-control-algoritmen die uit de wereld van de gedistribueerde databases afkomstig zijn. Op grond van hun lange en niet voorspelbare executietijden zijn zij ongeschikt voor een gedistribueerd real-time systeem. Bovendien zijn zij slechts bestand tegen een zeer beperkte klasse van fouten. Een ander voorbeeld zijn OSI-protocollen. Deze protocollen zijn veelal niet-deterministisch en hebben onvoorspelbare executietijden.

Associatieve systemen

Aan onderzoek liggen in essentie twee motieven ten grondslag: het oplossen van actuele problemen en het anticiperen op toekomstige ontwikkelingen. De vraag is wat de volgende stap in de ontwikkeling van gedistribueerde systemen zou kunnen zijn. Hoewel het voorspellen

van de toekomstige ontwikkeling altijd een hachelijke zaak is, wil ik met betrekking tot samenwerkende systemen een poging doen. Voor het ogenblik ga ik uit van mijn in het begin genoemde stelling dat de technische ontwikkeling een gevolg is van de algemene menselijke en sociale ontwikkeling.

Toegepast op het bedrijfsleven zien we, na een fase van consolidatie en professionalisering, steeds meer netwerken van wisselende associaties ontstaan. De individuele units zijn relatief zelfstandig, onafhankelijk en hebben een homogene innerlijke structuur; de samenwerkingsverbanden wisselen naar behoefte of per project. Dit verschijnsel, *networking* genoemd, treft men ook aan tussen professionals op alle mogelijke gebieden. Hoe zou een systeemstructuur er nu uit moeten zien die deze manier van werken goed kan ondersteunen?

Dit zijn mijns inziens associatieve of federatieve systemen [Litwin et al. '90] die de voordelen van beide typen van gedistribueerde systemen in zich verenigen. Het onderste gedeelte van afbeelding 2 laat deze ontwikkeling zien:

- Binnen een homogene technische of organisatorische eenheid biedt het werken met een gedistribueerd controlesysteem grote voordelen omdat zo'n eenheid aan een gemeenschappelijke taak werkt;

hier gaat het om een hechte koppeling ten behoeve van transparantie, hoge prestatie en hoge betrouwbaarheid.

- Tussen de verschillende eenheden kan met veel minder volstaan worden omdat het alleen om het uitwisselen van gegevens gaat. Hier is met name flexibiliteit en openheid van belang; tevens spelen ook hoge connectiviteit en voldoende bandbreedte een belangrijke rol.

Vandaag tekent zich deze ontwikkeling al duidelijk af. Bijvoorbeeld bestaat een moderne *digitale telefooncentrale* uit meerdere hardware- en software-modules die, al naar gelang de benodigde schakelcapaciteit, uitgebreid kunnen worden. Deze modules moeten nauw samenwerken en kunnen het best als gedistribueerd controlesysteem ontworpen worden. Voor het schrijven van correcte programma's met een hoge produktiviteit is het een groot voordeel als de structuur van de hardware en de details van de distributie van de verschillende resources door een gedistribueerd besturingssysteem verborgen wordt. Aan de andere kant maakt zo'n telefooncentrale deel uit van een veel groter netwerk. Het zou veel te onhandig en te duur zijn om de faciliteiten van het gedistribueerde controlesysteem ook over het hele netwerk aan te bieden. Dit zou ook onverstandig zijn omdat verschillende gedeeltes van zo'n netwerk

misschien door verschillende organisaties beheerd worden. Het is daarom veel logischer om voor de verbinding tussen de verschillende telefooncentrales een gedistribueerd processingsysteem te gebruiken.

Een ander voorbeeld is een *produktiebesturingssysteem*. Ook zo'n systeem bestaat uit een aantal produktiestations die deel uitmaken van een flexibele productiecel ofwel produktiestraat. Voor de besturing van een productiecel biedt het gebruik van een gedistribueerd operatingsysteem grote voordelen [Hammer '91]. Voor de besturing op fabrieks- of bedrijfsniveau kan daarentegen veel beter met een gedistribueerd processingsysteem gewerkt worden.

Voor het bijbehorende *logistieke systeem* doet zich een soortgelijke situatie voor. Lokaal (bijvoorbeeld binnen een shop of productie-eenheid) zijn de taken (scheduling van de produktie en routing van de produkten) zo nauw aan elkaar gerelateerd dat het best met een gedistribueerd controlesysteem gewerkt kan worden. Tussen verschillende produktie-eenheden, fabrieken of bedrijven kan volstaan worden met een veel lossere koppeling in de vorm van een gedistribueerd processingsysteem, dat bijvoorbeeld door middel van EDI geïmplementeerd is.

Het is interessant om ook nog vanuit een andere invalshoek naar de

VAN COMMUNICATIE NAAR SAMENWERKING: APPLICATIES

DATAPROCESSING

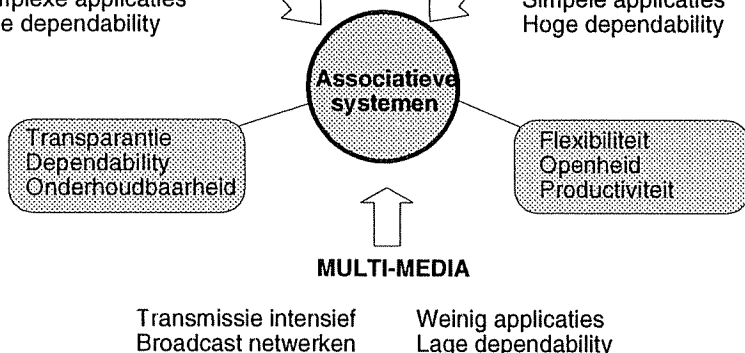
Gedistr. controle systemen

Data/besturings intensief
Simpel netwerken
Complexe applicaties
Lage dependability

DATACOMMUNICATIE

Gedistr. processing systemen

Transmissie intensief
Complexe netwerken
Simpel applicaties
Hoge dependability



Abbeelding 3 De applicatie-componenten van toekomstige gedistribueerde systemen.

toekomst te kijken, namelijk vanuit de toepassingsgebieden van gedistribueerde systemen. Globaal kan men drie grote gebieden onderscheiden: dataprocessing (alle soorten van administratieve en technische toepassingen), datacommunicatie (alles wat met de overdracht van gegevens te maken heeft) en multi-media toepassingen (televisie, interactieve CD, etc.). Dat ik dit laatste gebied hierbij noem zal misschien verbazing wekken, maar ook het televisienetwerk kan als gedistribueerd systeem opgevat worden, zeker als men er relaystations en satellieten bij betreft.

Bovendien bestaat er een groeiende behoefte aan multi-media toepassingen, waaraan, dankzij de groeiende bandbreedte en verwerkingscapaciteit van moderne computersystemen en netwerken, ook voldaan kan worden.

De meest belangrijke kenmerken van deze drie toepassingsgebieden zijn in afbeelding 3 geschetst.

Opvallend is het complementaire karakter van de verschillende gebieden. De integratie van deze drie gebieden zou dus een grote synergie opleveren. Dit is tevens de drijfveer voor de ontwikkeling van

toekomstige associatieve systemen. De noodzaak tot integratie van multi-mediasystemen zal duidelijk zijn. Omdat dit gebied niet tot mijn direct werkterrein behoort zal ik daar niet nader op ingaan en alleen een aantal interessante aspecten van de twee andere gebieden behandelen. Dataprocessingsystemen zijn tot nu toe gekenmerkt door complexe applicaties die door middel van relatief eenvoudige netwerken geïmplementeerd worden. Daarbij gaan moderne toepassingen heel duidelijk in de richting van gedistribueerde controlesystemen. Aan de andere kant zijn, uitzonderingen daargelaten, de dependabilityeisen nog relatief gering. Bij datacommunicatiesystemen is het precies andersom. Hier zijn in de afgelopen twintig jaar uitermate complexe netwerken ontstaan die aan hoge dependabilityeisen moeten voldoen. Helaas is er aan toepassing veel minder aandacht besteed. Pas in de afgelopen jaren kwamen heel langzaam bruikbare telematica-toepassingen op de markt en worden APIs (Application Programming Interfaces) voor verschillende toepassingen gedefinieerd, o.a. voor de koppeling met PC's en andere dataprocessingsystemen. Omdat de nadruk nog steeds op communicatie ligt kunnen de zo ontstane systemen meestal als gedistribueerde processingsystemen ingeschaald worden. Alleen ontstaan de laatste tijd in de applicatielaag van het OSI-Model steeds meer protocollen die

de coördinatie van meerdere partijen ondersteunen. Een goed voorbeeld voor deze trend naar gedistribueerde controlesystemen is het ODP (Open Systems Processing) model [ODP '92].

De term dependability die ik zojuist gebruikte, vraagt om enige uitleg. Onder *dependability* verstaat men de verzameling van alle niet-functionele eisen zoals prestatie (real-time gedrag), betrouwbaarheid (weinig fouten), beschikbaarheid (grote inzetbaarheid), zekerheid (het niet optreden van catastrofale fouten) en veiligheid (het bestand zijn tegen hackers en andere soorten van frauduleuze manipulatie) [Laprie '89]. Deze eigenschappen worden des te belangrijker naarmate er meer privépersonen en ondernemingen bij de uitvoering van hun activiteiten afhankelijk zijn van het correct werken van hun computersystemen. Vroeger was het vaak zo, dat dependability-aspecten nauwelijks werden gespecificeerd en eigenlijk pas tijdens het ontwerp en de implementatie van een systeem aan de orde kwamen. Als er echter aan een informatiesysteem hoge dependabilityeisen gesteld worden is deze werkwijze niet meer voldoende. In de toekomst zal het steeds belangrijker worden om de dependabilityaspecten uit het te ondersteunen proces af te leiden, formeel te specificeren, te verfijnen en te verifiëren. Voor mij is het dan een uitdaging om in de zo ontstane multi-dimensionele

ontwerpruimte net zo systematisch te werk te gaan als wij dat voor de functionaliteit van een systeem gewend zijn.

Helaas werken verschillende groepen aan de verdere ontwikkeling van deze drie toepassingsgebieden van gedistribueerde systemen. Het feit dat deze groepen verschillende doelstellingen hebben en verschillende talen spreken, heeft de integratie tot nu toe behoorlijk vertraagd. De uitdaging voor een universiteit zie ik dan ook in de bevordering van deze integratie door gemeenschappelijke kenmerken van de verschillende applicaties uit te werken en deze visie duidelijk uit te dragen. Een belangrijk neveneffect van deze activiteit is zeker de reductie van de complexiteit en de kosten van toekomstige gedistribueerde systemen. Het eerste effect zal de betrouwbaarheid en zekerheid van zulke systemen ten goede komen; het tweede effect zal de praktische toepassing van moderne technieken bevorderen.

Gedistribueerde operating systemen

Na dit overzicht over de wereld van de samenwerkende systemen wil ik U graag mee nemen voor een uitstapje naar mijn eigen onderzoeksgebied, gedistribueerde operatingsystemen. U heeft zich misschien al afgevraagd waarom iemand, wiens leeropdracht "technische toepassingen van de informa-

tica" is, nu zo aan samenwerkende systemen verknocht is. Dat heeft simpelweg te maken met het feit dat ook de meeste toepassingen gedistribueerd zijn. Een tweede reden is dat gedistribueerde systemen inherent parallel werken en vele redundante resources kunnen bevatten, zoals processoren, netwerkverbindingen en perifere eenheden. Het eerste feit uit zich in de hoge rekencapaciteit van gedistribueerde systemen en de mogelijkheid om separation of concerns toe te passen door functioneel gescheiden taken op verschillende processoren te laten lopen. Het tweede feit heeft tot gevolg dat gedistribueerde systemen inherent foutentolerant zijn, tenminste als zij goed ontworpen worden. De vraag is nu: hoe kunnen gedistribueerde controlesystemen op een zodanige manier geconstrueerd worden dat zij efficiënt, dependable en gemakkelijk te programmeren zijn?

Mijn antwoord is simpel: door ervoor te zorgen dat zowel de software-ontwikkelaars als ook de door hen gemaakte applicatieprogramma's goed samen kunnen werken. In de praktijk komt dat op het volgende neer:

1. *Abstraheren:*

Het bouwen van goede gedistribueerde operatingsystemen die de vele vervelende details van het bestuurd proces en het onderliggende netwerk abstraheren, de ter beschikking staande resources

efficiënt administreren en het implementeren van dependable systems ondersteunen. Het ontwerpen van geschikte hoge niveautalen die het de applicatie-programmeur toestaan om naast de functionaliteit van het systeem ook de dependability-eisen en de distributie-eisen op een natuurlijke en eenvoudige manier te specificeren.

Omdat wij nog aan het begin van dit onderzoek staan wil ik hier niet verder op ingaan. Voor een goede oplossing op dit gebied is de inbreng van verschillende informatica-specialismen nodig is, en ik kan mij erop verheugen om dit thema in de toekomst samen met mijn collega's verder uit te diepen.

2. *Samenwerken en coördineren.*

Dit behelst het ontwerpen van een verzameling van gedistribueerde algoritmen die niet alleen op zichzelf efficiënt, maar ook nog met elkaar compatibel zijn. Met betrekking tot datacommunicatie moeten deze algoritmen veel eenvoudiger en efficiënter zijn dan hun OSI-tegenhangers. Dit betekent tevens dat de zogenoemde protocolstack van zeven lagen naar drie lagen gereduceerd kan worden.

3. *Proces-georiënteerd werken.*

Daarmee bedoel ik het ontwikkelen van software-engineering methoden die het mogelijk maken om de specificatie van een systeem op een natuurlijke manier uit het te ondersteunen proces af te leiden en om zo'n specificatie

op een systematische manier naar een implementatie te vertalen.

Op dit onderwerp zal ik in het laatste gedeelte van mijn rede nog terugkomen.

Terug naar gedistribueerde operatingsystemen en naar het DEDOS (Dependable Distributed Operating System) project dat het voertuig voor mijn onderzoek is. Misschien vraagt u zich nu af wat een gedistribueerd operatingsysteem, dat toch een tamelijk fundamenteel stuk software is, nu in hemelsnaam met technische toepassingen van de informatica te maken heeft? Ook hier is het antwoord simpel: er zijn ontelbare technische toepassingen van de informatica en dagelijks komen er nieuwe bij. Wil men niet versnipperen, en dat zou voor serieus onderzoek funest zijn, dan moet men naar het kleinste gemeenschappelijke veelvoud van alle toepassingen zoeken. Welnu, dit zijn volgens mij gedistribueerde operatingsystemen die als gemeenschappelijk hoge-niveau-executieplatform voor de meest uiteenlopende applicaties kunnen fungeren, zoals telefooncentrales, kopieermachines, avionics systemen, productiebesturingssystemen, logistieke besturingssystemen en besturing van fysische experimenten. Een leuke bijkomstigheid is, dat ook op het gebied van de datacommunicatie naar soortgelijke oplossingen gezocht wordt. Alleen praat men daar over IN's (Intelligent Networks).

Natuurlijk zijn er verschillen, zoals het feit dat het concept van intelligente netwerken geen duidelijk verschil maakt tussen operating-systeem en applicatie en ook een zogenaamd SCE (Service Creation Environment) bevat, dat wil zeggen een geavanceerde softwareontwikkelomgeving voor telecommunicatietoepassingen. Niettemin is het uitgangspunt hetzelfde, namelijk om de programmering van de applicatiesoftware eenvoudiger en efficiënter te maken.

Van zo'n stuk systeemsoftware wordt vooral efficiëntie en *transparantie* [Tanenbaum '92] verwacht. Het laatste betekent dat de gebruiker, dat wil zeggen de applicatieprogrammeur, zich niet met alle mogelijke vervelende details moet bezighouden. Ik geef hier alleen enkele voorbeelden:

- Applicatieprogramma's moeten met elkaar kunnen praten in termen van logische namen in plaats van netwerkadressen. Getransponeerd naar het telefoonnetwerk zou dat betekenen dat u de naam opgeeft van degene die u wilt bellen en niet zijn telefoonnummer.
- De tijdseisen die in het programma gespecificeerd zijn moeten door het operatingsysteem gewaarborgd worden, zonder dat de applicatie dit met behulp van alle mogelijke timeouts zelf moet bewaken.
- De effecten van fouten moeten liefst door het operatingsysteem

opgelost worden, zonder dat de applicatie daar iets van merkt. Dit is bijvoorbeeld mogelijk door het systeem dynamisch te reconfigureren, programma's van gefaalde processoren naar andere machines te verhuizen en boodschappen over andere routes te versturen.

Andere belangrijke voordelen van het gebruik van een gedistribueerd operatingsysteem zijn een hogere produktiviteit voor het maken van programmatuur en een hogere flexibiliteit ten opzichte van uitbreidingen en veranderingen. Het laatste is simpelweg een gevolg van de bovengenoemde transparantie. De hogere programmeerproduktiviteit is, gezien het toenemend aandeel van de softwarekosten aan de totale produktkosten, een hot item. Zij is een direct gevolg van het feit dat de applicatieprogrammeur zich niet meer bezig hoeft te houden met een aantal vervelende gevolgen van de distributie en daardoor ook in staat gesteld wordt om correctere programma's te schrijven. Juist de synchronisatie van concurrerende resource-behoeften, de afhandeling van fouten en het bewaken van de consistentie van gerepliceerde resources zijn ingewikkelde taken, die tot veel programmeerfouten aanleiding geven.

Voor ik naar het volgende hoofdstuk ga wil ik nog een aantal aspecten van het DEDOS-project kort toelichten. Want gedistribueerde

operatingsystemen zijn er meer en rijst terecht de vraag naar de toegevoegde waarde van dit project. Deze schuilt in de eerste twee letters die voor dependability staan, waarbij wij ons op het ogenblik met name op de combinatie van real-time gedrag en betrouwbaarheid richten.

Er wordt een onderscheid gemaakt tussen hard real-time taken, die hun deadlines onder alle omstandigheden moeten halen, en soft real-time taken voor welke het te laat uitvoeren van een taak geen catastrofale gevolgen heeft. De tijdseigenschappen van harde real-time taken worden vooraf (off-line) door middel van deterministische heuristische schedulingmethoden geverifieerd [Verhoosel et al. '91]. Het feit dat men in dit geval altijd van het slechtst mogelijke geval moet uitgaan heeft tot gevolg dat de systeemresources heel inefficiënt gebruikt worden. Het resultaat is een onevenredig duur systeem dat alleen voor kritieke toepassingen met hoge zekerheidseisen zinvol ingezet kan worden. Om toch tot een efficiënt systeem te kunnen komen, worden de periodes waarin geen hard real-time taken lopen voor soft real-time doeleinden gebruikt. Daarvoor kunnen conventionele dynamische (on-line) schedulingmethoden toegepast worden die een hoge resource utilisatie, en daarmee een efficiënt en goedkoop systeem, waarborgen.

Deze eigenschappen moeten natuurlijk ook bij het optreden van

fouten bewaard blijven. Om dit te bereiken wordt de fouten-tolerantie voor hard en soft real-time taken op twee verschillende manieren verwezenlijkt, namelijk door programma-replicatie en door dynamische reconfiguratie, gevolgd door error-recovery. Deze tweedeling doortrekt als het ware het hele systeem en wordt daarom ook het *dual dependability paradigm* genoemd.

Samenwerking in organisaties

Organisatievormen en systeemarchitecturen

Dames en heren,
De vier ontwikkelingsfasen van samenwerkende systemen die ik hierboven geschetst heb zijn niet het produkt van een min of meer toevallige technologische ontwikkeling. Zij komen overeen met de vier ontwikkelingsfasen van een organisatie, zoals die bijvoorbeeld in [Glasl et al. '93] beschreven zijn. Deze samenhang is in afbeelding 4 samengevat.

Omdat dit niet de plaats is om op de details van de verschillende ontwikkelingsfasen van een organisatie in te gaan, beperk ik mij tot drie aspecten: wie heeft het voor het zeggen, hoe verloopt de communicatie en hoe ziet het ondersteunende informatiesysteem er idealiter uit?

- De *pioniersfase* van een organisatie is gekenmerkt door een domi-

VAN COMMUNICATIE NAAR SAMENWERKING: ORGANISATIES

Ontwikkelings-fase	Kenmerk	Organisatie-vorm	Communicatie-vorm	Informatie-systeemtype
Pionier	Centralisatie Universaliteit	Patriarchaat Leiderschap	One voice	Centrale systemen
Differentiatie	Specialisatie Ordering	Hierarchie Delegatie	Hierarchisch	Gekoppelde autonome systemen (vaak hierarchisch)
Integratie	Samenhang Taakverdeling	Democratie Econom. coöper.	Gedistribueerd (Gestructureerd)	Gedistribueerde systemen
Associatie	Situationeel Teamwerk	Vrije associatie Ideële coöper.	Vrij (Na behoefte)	Associatieve systemen

versus de ontwikkeling van de samenwerking tussen computersystemen.

nerende rol van de oprichter, de pionier dus. Hij is de leider die over alles gaat en die ook het centrum is van alle activiteiten en de bijbehorende communicatie. Het daarbij passende informatie-systeemtype is een gecentraliseerd systeem, dat meestal zijn uitgangspunt in de administratie heeft en vanuit een klein begin (bijvoorbeeld een PC) met de onderneming meegroeit.

Als de organisatie groeit is deze aanpak op den duur natuurlijk niet meer vol te houden. De leider is nu niet meer in staat om alles te overzien en moet taken afstaan. De verschillende onderdelen van de organisatie moeten zich vervolgens specialiseren om professioneel te kunnen werken.

- In de *differentiatiefase* wordt de organisatie volgens technocratisch-rationele principes gestructureerd en gecoördineerd. Er is een duidelijke top die wel taken naar beneden delegeert, maar toch probeert om alles in de gaten te houden en te controleren. De verschillende specialismen, zoals ontwikkeling, productie en administratie, bouwen nu hun eigen computersystemen op. Deze zijn voor hun specifieke behoeften ingericht en meestal niet compatibel. Door verbinding van deze automatiseringseilanden ontstaat uiteindelijk een netwerk van gekoppelde autonome systemen, meestal met een duidelijke top die in de administratie verankerd is. Mettertijd wordt zo'n organisatie zo

star en bureaucratisch dat ze alleen nog met zichzelf bezig is, de gemeenschappelijke visie verloren gaat en zij niet meer flexibel op de behoeften van de markt kan inspelen.

- In de *integratiefase* bezint de onderneming zich op haar eigenlijke taken en ontwikkelt weer een gemeenschappelijke visie en strategie. Dit geeft tevens de mogelijkheid om de organisatie in goed overzienbare cellen of units te splitsen, die relatief onafhankelijk hun eigen interne of externe klanten kunnen bedienen. Ook de communicatiepatronen en beslissingsprocedures worden duidelijk vrijer en democratischer. Deze fase kan het best door een gedistribueerd informatiesysteem ondersteund worden. Volgens de momenteel gebruikelijke technologie zal dit in de meeste gevallen een gedistribueerd processing-systeem zijn. Maar in technologie-intensieve gebieden, zoals ontwikkeling en produktie, is al een duidelijke trend naar gedistribueerde controlesystemen te zien. Op een gegeven ogenblik ontstaat voor de onderneming de noodzaak om zich meer aan de buitenwereld te oriënteren en de overgang naar de associatiefase te maken. Dit heeft twee oorzaken. Aan de ene kant moet men steeds directer met de klant samenwerken om snel en adequaat op zijn behoeften in te kunnen spelen. Aan de andere kant moet intensiever met

toeleveranciers samengewerkt worden omdat het economisch niet meer haalbaar is om alle specialismen in huis te hebben. Op deze manier ontstaan ketens van bedrijven die zich ieder afzonderlijk op een bepaald stuk van het waardescheppingsproces richten [Porter '85].

- De *associatiefase* van een onderneming kenmerkt zich naar buiten door oriëntatie op het primaire proces, dat bij de toeleveranciers begint en bij de klanten eindigt. De belangrijkste kenmerken binnen zo'n organisatie zijn: de vaardigheid om met elkaar in wisselende teams samen te werken, samen te leren en samen de gemeenschappelijke idealen, visies en strategieën voortdurend aan de omgeving aan te passen. Dit uit zich onder andere in de continue inspanning om de kwaliteit van de bedrijfsprocessen te verbeteren, bijvoorbeeld met behulp van TQM (Total Quality Management) [TQM '88]. Het totale concept is ook onder de naam *Lean Enterprise* (slanke onderneming) of *Lean Production* [Womack et al. '90] bekend.
- Het zal intussen duidelijk zijn dat ik van mening ben, dat een organisatie in de associatiefase het best door een associatief informatiesysteem ondersteund kan worden. Binnen een processtap of unit ligt het voor de hand om van een gedistribueerd controlesysteem gebruik te maken. De wisselende

contacten tussen de verschillende units en de contacten naar buiten kunnen op een flexibele manier via een gedistribueerd processing-systeem verzorgd worden.

Ik heb met dit uitstapje in de wereld van de organisatieontwikkeling willen aantonen dat organisatievormen en informatiesysteem-architecturen nauw met elkaar verbonden zijn. Bij iedere organisatievorm hoort als het ware een ideale systeemstructuur. Hier dient zich natuurlijk weer het in de inleiding genoemde kip-en-ei-probleem aan, namelijk: wat is waarvoor de drijfveer, de organisatievorm voor de technische structuur of andersom? De historische ontwikkeling laat duidelijk zien dat bij de organisatiestructuren de technische oplossingen altijd vooruit lopen. Ik zal dit gegeven hier niet nader toelichten, omdat het op het terrein van mijn collega's van de geschiedenis van de techniek ligt. Het is wel zo, dat deze samenhang bij nader inzien niet zo verwonderlijk is. Immers leven de ingenieurs die de informatiesystemen ontwerpen zelf in een organisatiecultuur die hun manier van denken sterk beïnvloedt. Dit denken vertaalt zich via de bouwplannen uiteindelijk naar de fysieke artefacten waarvan een bedrijf gebruik maakt. Hoe meer toekomstgericht de ideeën, des te meer toekomstgericht ook de technische artefacten die daaruit voortkomen. Ik ben er dan ook van overtuigd, dat het belangrijkste wat wij als universiteit een student behal-

ve gedegen vakopleiding mee kunnen geven, een ruime en onconventionele manier van denken is.

Uniforme ontwerp-paradigmen

Ik wil met u nog wat langer stil blijven staan bij de analogie tussen organisaties en computersystemen. Niet alleen omdat dit mijn hobby is, maar ook omdat men daaruit een heleboel inzichten op kan doen die mijns inziens in de toekomst een steeds belangrijker rol zullen gaan spelen.

Een belangrijk hedendaags vraagstuk is het herontwerpen van organisaties die de integratie- of associatiefase ingaan. Onderdeel van deze activiteit is ook het herontwerp van informatiesystemen die de vernieuwde bedrijfsprocessen efficiënt kunnen ondersteunen. Beide activiteiten vat men ook onder het begrip *Business Proces Reengineering* samen, dat de laatste tijd toeneemende belangstelling geniet.

Zo'n herontwerp van een organisatie en haar informatiesystemen vraagt om samenwerking tussen verschillende vakdisciplines: organisatieontwikkelaars, human resource specialisten, opleidingspecialisten, informatie-analisten, bedrijfskundigen, machinebouwers, systeemarchitecten, software-ingenieurs en nog vele anderen. De trend naar specialisatie zal zich zeker voortzetten omdat dit een voorwaarde voor professionaliteit is. Daarom zal het er in de toekomst steeds meer op aankomen dat de verschillende

specialismen ten behoeve van een bepaalde taak of project doelgericht samenwerken. De basis voor een effectieve samenwerking tussen zulke verschillende gebieden zijn gemeenschappelijke visies, strategieën en ontwerp-paradigmen.

Met name de uniforme ontwerp-paradigmen zijn belangrijk voor de consistentie en correctheid van het (her)ontwerp. Anders bestaat het gevaar dat iedere discipline haar eigen methoden en gereedschappen gebruikt. De vertaling van een deelontwerp naar het volgende deeltraject wordt daardoor onnodig moeilijk gemaakt en de kans op fouten neemt toe. Vandaag de dag is dit een serieus probleem, vaak zelfs binnen één discipline. Een voorbeeld uit mijn eigen vakgebied is software engineering, waar de analyse-, specificatie-, ontwerp- en programmeer-methoden vaak niet goed op elkaar aansluiten.

Zowel bij het ontwerp van een organisatie als ook bij het ontwerp van een computersysteem moeten vooral twee aspecten gemodelleerd worden. Ten eerste de samenstelling van zo'n entiteit uit zijn onderdelen of modules. Omdat op deze manier als het ware het systeem in rust beschreven wordt spreekt men ook van de statische structuur. Ten tweede het gedrag ofwel de activiteiten van zo'n entiteit. Omdat het hier om een beschrijving van het systeem in beweging gaat spreekt men ook van de dynamische structuur.

Om tot universele ontwerp-

paradigmen te komen ga ik uit van de volgende waarnemingen:

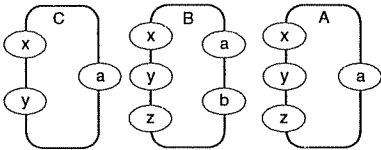
- Moderne organisatievormen (zoals Lean Enterprise) en systeemarchitecturen (zoals associatieve systemen) zijn op *samenwerking* gebaseerd: in organisaties praat men over teamwork, ingenieurs bouwen gedistribueerde systemen met behulp van gedistribueerde algoritmen.
- De statische structuren waar deze samenwerking zich in afspeelt bestaat uit relatief kleine zelf-verantwoordelijke *eenheden*, die rond een bepaalde taak georganiseerd zijn. Deze eenheden kenmerken zich door een hoge interne samenhang en goed gedefinieerde interfaces naar de buitenwereld: in een organisatorische context heet zo'n eenheid unit, bij technische systemen spreekt men over objecten.
- De dynamische structuren waar deze samenwerking zich aan oriënteert zijn de processen. Zo'n proces bestaat uit een opeenvolging van goed gedefinieerde *activiteiten* die via evenveel goed gedefinieerde interfaces met elkaar verbonden zijn: in organisatietermen zijn dit de processtappen, in technische termen zijn dit de services (vaak ook operaties of methoden genoemd), die de objecten aan hun omgeving ter beschikking stellen. Een bepaalde keten van activiteiten wordt vaak ook transactie of executie genoemd.

PROCES-GEORIENTEERD ONTWERPEN

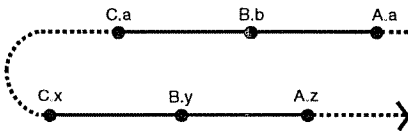
Proces



Objecten: Resourcebeheer



Activiteiten: Gedrag
Informalbestromen
Executies/Transacties



Afbeelding 5 Proces-georiënteerd ontwerpen

Mijn stelling is dat *object-georiënteerde* methoden [Meyer '88] uitermate geschikt zijn om statische structuren te modelleren, terwijl een *activiteiten-georiënteerde* benadering nodig is om de dynamische structuur van een systeem in kaart te brengen [Hammer '93b]. Vanuit deze twee universele paradigma's kan het hele ontwerptraject, van het (her)ontwerp van de organisatieprocessen tot en met de implementatie van het bijbehorende informatiesysteem, benaderd worden.

Proces-georiënteerd ontwerpen

Onder proces-georiënteerd ontwerpen versta ik een probleembena-

dering die zich aan het te ondersteunen proces oriënteert. Dit proces kan zowel een ontwikkelproces, een productieproces als een gebruikproces zijn. Een belangrijk kenmerk van deze opzet is dat aan het ontwerp van de dynamische eigenschappen net zo veel aandacht wordt besteed als aan het ontwerp van de structuur. Verder moeten deze twee aspecten goed geïntegreerd zijn, dat wil zeggen er moet een gemeenschappelijke taal met een goed gedefinieerde syntax en semantiek bestaan. In de verdere uitbouw van deze benadering en de integratie van de verschillende deeltrajecten met behulp van de twee bovengenoemde uniforme paradigma's zie ik een belangrijke taak voor de toekomst.

De basisopzet van proces-georiënteerd ontwerpen met behulp van objecten en activiteiten is in afbeelding 5 te zien. Een proces wordt in beweging gebracht door de vraag van een interne of externe klant. De keten van activiteiten eindigt als aan de behoeften van die klant is voldaan. Voor het uitvoeren van de verschillende processtappen zijn organisatorische eenheden verantwoordelijk, die door objecten gemodelleerd worden. Hetzelfde geldt voor de informatiesystemen die deze eenheden bij de vervulling van hun taak ondersteunen. De activiteiten van het organisatorische of technische systeem kunnen dan in termen van service-aanroepen van deze objecten gemodelleerd worden.

Bij een programma spreekt men in plaats van activiteiten ook vaak van call-grafen.

Afbeelding 6 laat de verschillende aspecten van proces-georiënteerd ontwerpen meer in detail zien. Het leuke is, dat bij deze benadering ook de verschillende dependability-aspecten op een natuurlijke manier gespecificeerd kunnen worden.

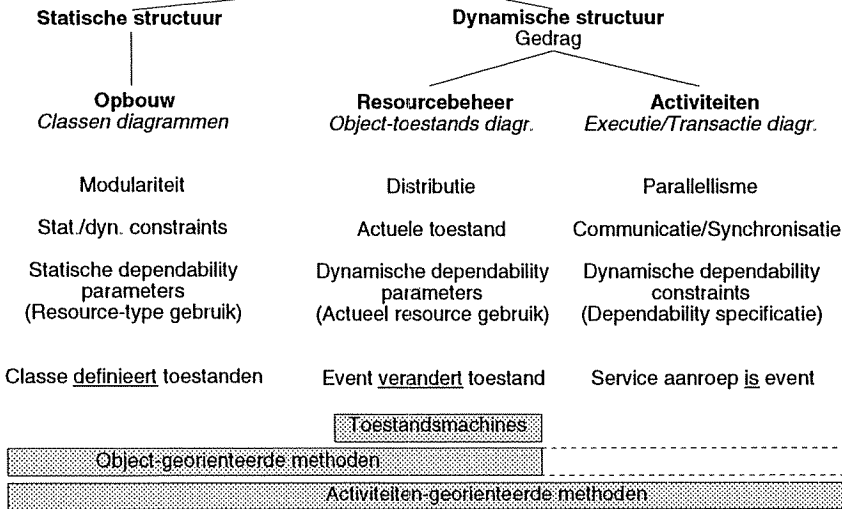
- De statische structuur wordt volgens het object-georiënteerd paradigma met behulp van klassen-diagrammen ontworpen. Daarmee worden de verschillende aspecten van de modulariteit [Meyer '88] bepaald. Door middel van klassen-invarianten, respectievelijk pre- en postcondities kunnen verder de statische respectievelijk dynamische constraints gedefinieerd worden. Tevens worden ook de *statische dependability parameters* vastgelegd, dat wil zeggen de resource-typen die door de objecten van een bepaalde klasse gebruikt kunnen worden. Een typisch voorbeeld van een statische dependability-constraint is de specificatie van die fout-typen waarmee rekening gehouden wordt door middel van een zogenaamde fout-hypothese.
- Uitgaande van een object-georiënteerde benadering heeft het systeemgedrag twee componenten:
 - Aan de ene kant worden objecten van een bepaald type of klasse dynamisch geïnstantieerd. Een

object bevindt zich altijd in een bepaalde toestand, waarbij de toestandsruimte (dat wil zeggen de verzameling van alle mogelijke toestanden) door de bijbehorende klasse gedefinieerd is.

Objecten worden voor een bepaalde executie-omgeving geïnstantieerd, bijvoorbeeld voor een bepaalde processor in een gedistribueerd systeem en de op hem draaiende systeemsoftware (operatingsysteem, databasesysteem, etc.). Zo'n object beheert bepaalde resources, zoals data-structuren, sensoren en actuatoren. Het actuele gebruik van deze resources is een belangrijk ontwerp-criterium en wordt een *dynamische dependability parameter* genoemd.

- Aan de andere kant moet ook de opeenvolging van activiteiten door middel van executie- of transactie-diagrammen net zo systematisch ontworpen worden als de statische structuur. In een gedistribueerd systeem kunnen activiteiten zich van de ene naar de andere processor voortplanten en kunnen verschillende activiteiten parallel aflopen. Daardoor ontstaat ook de noodzaak van communicatie en synchronisatie. Al deze verschijnselen kunnen met behulp van executie- en transactiediagrammen overzichtelijk weergegeven en ontworpen worden. Uiteindelijk kan met behulp van executies ook geverifieerd worden in hoeverre het

PROCES-GEORIENTEERD ONTWERPEN



Afbeelding 6 De verschillende aspecten van proces-georiënteerde ontwerpen.

ontwerp aan de dependability-eisen, ofwel *dynamische dependability constraints*, voldoet. Dit wordt verklaard door het feit, dat dependability een dynamische eigenschap is en executies die entiteiten zijn, die resources gebruiken en tegen dynamisch optredende fouten aanlopen. In het DEDOS-systeem wordt bijvoorbeeld al tijdens het ontwerpen met behulp van executie-grafen gecheckt in hoeverre hard real-time taken aan hun tijdseisen voldoen [Verhoosel '93].

Huidige object-georiënteerde methoden zijn vooral geschikt voor het ontwerpen van de systeemstructuur. Gedeeltelijk staan zij ook het model-

leren van objecten en hun toestandsdiagrammen toe. Om een proces-georiënteerde ontwerp-methodologie te kunnen ondersteunen zal het in de toekomst nodig zijn om object-georiënteerde methoden uit te breiden tot activiteiten-georiënteerde methoden, zoals ik dit in het onderste gedeelte van afbeelding 6 aangeduid heb.

Voor alle duidelijkheid wil ik er nog op wijzen dat ik hier geenszins voor een puur technocratische aanpak pleit, waarin een levende organisatie in de vorm van een al dan niet deterministische automaat gemodelleerd wordt. In tegendeel, het is uitermate belangrijk, dat voor het modelleren en (her)ontwerpen van de bedrijfsprocessen alle organisa-

tie-aspecten, zoals bijvoorbeeld beschreven in [Glasl et al. '93], meegenomen worden.

Informatie-Management

Dames en heren,

In het voorafgaande heb ik al een aantal keren op ontwikkelingen van het vakgebied gewezen, die of reeds in gang zijn of mijn inziens in de toekomst belangrijk zullen worden. Tot besluit wil ik mijn visie op de ontwikkeling van de informatica nog een keer samen vatten.

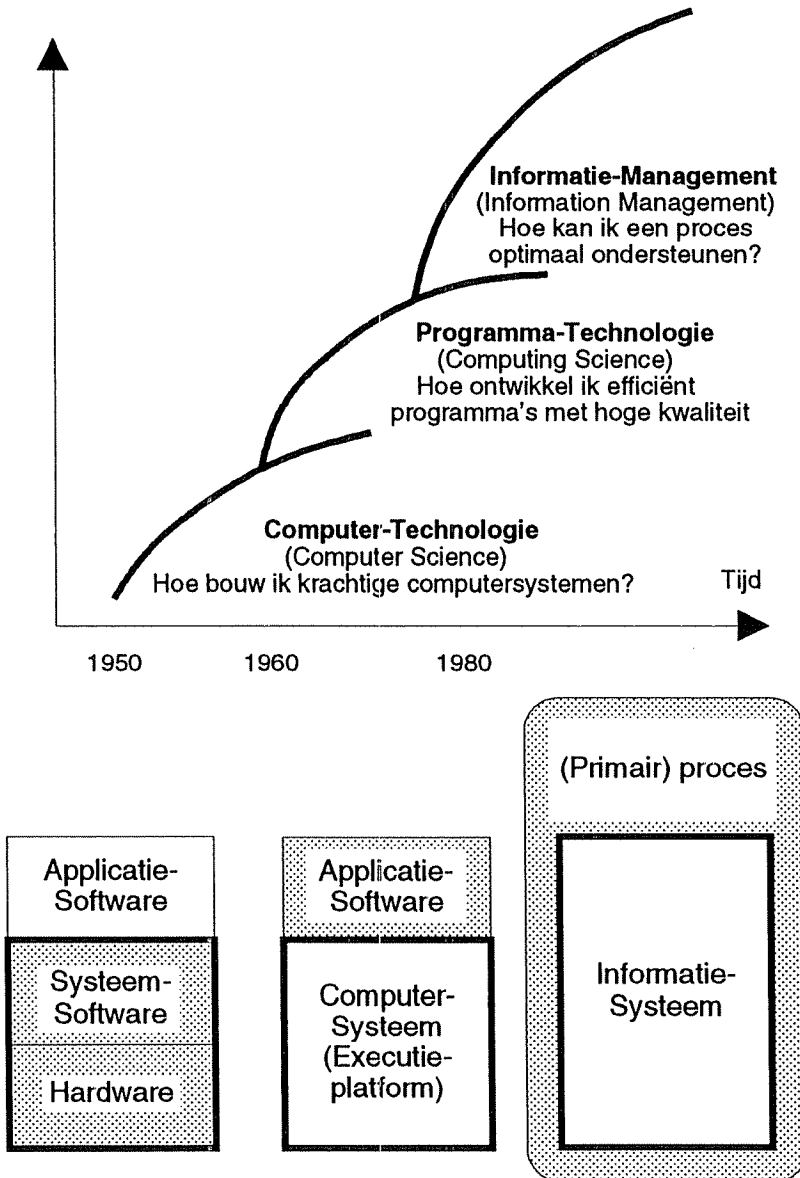
Daarvoor wil ik met u naar de historische ontwikkeling kijken zoals die in afbeelding 7 te zien is. Voor de juiste interpretatie van deze afbeelding is het belangrijk om te beseffen dat de verschillende ontwikkelingsstadia van de informatica niet alleen een logisch vervolg op elkaar zijn, maar dat de ontwikkeling van een eerder ontgonnen deelgebied ongeremd doorgaat als een nieuw gebied ontgonnen wordt. Sterker nog, de verdere ontwikkeling van een ouder deelgebied is voorwaarde voor de groei van een nieuw deelgebied. De wortels van de informatica zijn achtereenvolgens te vinden in de natuurkunde (actieve halfgeleider), de analoge elektronica (transistoren) en de digitale elektronica (logische schakelingen).

- De eigenlijke ontwikkeling van de informatica begon pas in de 50er jaren. Het eerste doel was de ontwikkeling van de hardware en

met name het bouwen van krachtige computersystemen. In een latere fase kwam daar nog het bouwen van dicht bij de hardware liggende systeem-software, zo als operatingsystemen en databases, bij. Dit ontwikkelingsstadium laat zich het beste omschrijven als *Computer-Technologie* ofwel *Computer Science*.

- Dit was de basis waarop zich vanaf de 60er jaren de *Programma-Technologie* ofwel *Computing Science* kon ontwikkelen. Nadat de eerste complete computersystemen ofwel executieplatforms beschikbaar waren, verschoof de aandacht naar het maken van de applicatiesoftware. Natuurlijk werden ook de eerste computers geprogrammeerd, maar de programma's waren oorspronkelijk klein en werden door hooggespecialiseerde wetenschappers en technici ontwikkeld. Pas met het snel omvangrijker worden van de programma's en de net zo snelle toename van het aantal software-specialisten ontstond de behoefte aan een systematische aanpak. Het ging nu om de vraag hoe efficiënte en kwalitatief hoogwaardige (dat wil zeggen correcte) programma's met een hoge productiviteit ontwikkeld konden worden. Binnen onze faculteit is op dit gebied baanbrekend werk verricht, met name door programma's systematisch uit hun specificaties af te leiden; een impuls die vandaag de dag nog steeds vruchtbaar is.

DE ONTWIKKELING VAN DE INFORMATICA



Afbeelding 7 De ontwikkeling van de informatica.

De vraag is nu wat de volgende stap zal zijn.

- Naar aanleiding van mijn betoog kunt u de richting waarin ik denk vermoedelijk al raden. Het gaat nu niet meer alleen om het construeren van correcte informatiesystemen, maar ook om het bouwen van systemen die een gegeven proces optimaal kunnen ondersteunen. Met andere woorden, het ontwerpen van de systeemspecificaties in een bedrijfsmatige context zal steeds belangrijker worden. Deze trend komt ook in een rapport van het NGI (Nederlandse Genootschap voor Informatica) naar voren, dat de behoefte aan een nieuwe type van informatica-ingenieurs formuleert [NGI '89]. Ik noem dit nieuwe gebied van de informatica *Informatie-Management* ofwel Information Management. Voor de verdere ontwikkeling van dit gebied zijn drie dingen nodig:
 - Een verfijning van de gebruikelijke requirements-engineering methoden. De gebruikelijke requirements-engineeringmethoden kennen twee grote bezwaren. Ten eerste richten zij zich teveel op het modelleren van de statische structuur en zijn te weinig proces-georiënteerd. Ten tweede laten zij zich teveel door de techniek inspireren en hebben te weinig aandacht voor organisatorische en bedrijfskundige aspecten. Zoals ik al meerdere keren toegevoegd heb, zou een verbeterde me-

thode het afleiden van een informatiesysteem uit het te ondersteunen proces, moeten ondersteunen. Zo'n aanpak zou bijvoorbeeld de volgende vijf stappen moeten bevatten [Hammer '93c]:

1. *Procesevaluatie*, om de bestaande situatie in kaart te brengen en problemen te identificeren.
2. *Strategie-ontwikkeling*, om de interne en externe doelstellingen en strategieën helder te krijgen.
3. *Procesontwikkeling*, om het proces en de organisatie volgens de vernieuwde doelstellingen te herontwerpen.
4. *Informatie-ontwikkeling*, om de informatiestromen en informatietypen te identificeren die voor de ondersteuning van het proces nodig zijn.
5. *Systeemontwikkeling*, om het informatiesysteem te specificeren, te ontwerpen en te implementeren dat deze informatie efficiënt en betrouwbaar kan verwerken.

Volgens het basisprincipe van een Lean Enterprise zou bij de laatste drie stappen steeds naar minimale oplossingen gezocht moeten worden. Dit lukt echter alleen als de eerste twee stappen goed uitgevoerd zijn en met name als de doelstellingen duidelijk zijn.

- Multi-disciplinair samenwerken. Tegenwoordig is het niet meer voldoende dat een informatie-analist of informaticus de gebruiker ondervraagt en na een tijdje met een oplossing komt die gebaseerd

is op zijn noodzakelijkerwijze beperkte begrip van het bedrijfsprobleem. Ook oplossingen die alleen door de ter beschikking staande technologie ingegeven zijn voldoen niet.

Mijn stelling is, dat een goede oplossing alleen in een multi-disciplinair team gevonden kan worden. In een administratieve omgeving zal zo'n projectteam bijvoorbeeld bestaan uit gebruikers die tevens de processpecialisten zijn, uit organisatie-specialisten, informatie-analisten en informatica-ingenieurs. Voor het bouwen van embedded software zullen applicatie/processpecialisten, bedrijfskundigen, machinebouwers, hardware-ingenieurs en software-ingenieurs nauw samen moeten werken om tot een optimaal ontwerp te komen. Het is belangrijk om zich te realiseren, dat multidisciplinair samenwerken niet alleen een zaak van goede wil is, maar openheid en

begrip eist voor elkaars manier van denken en werken. Tussen bijvoorbeeld informatici, elektrotechnici en machinebouwers valt dat nog wel mee, omdat alle drie disciplines bij de technische wetenschappen horen en een gemeenschappelijke achtergrond hebben. De samenwerking met vertegenwoordigers van organisatie- en humanwetenschappen is al een stuk moeilijker, omdat hun wereldbeeld er vaak heel anders uitziet. Om dit te verduidelijken heb ik de meest belangrijke verschillen in afbeelding 8 in het kort samengevat. Als technici hebben wij vaak de neiging om alles over één kam te scheren en bijvoorbeeld de synthetisch-intuïtieve werkwijze van de humanwetenschappen als achterhaald te beschouwen. Ik ben echter van mening dat het juist andersom is en dat wij een heleboel van andere disciplines kunnen leren. In feite gaat het altijd om een evenwichtige benadering, waarin

MULTI-DISCIPLINAIR SAMENWERKEN

eist begrip voor elkaars manier van denken en werken

	Technische Wetenschappen	Humaan/Organisatie Wetenschappen
Denkwijze	Analytisch-rationaal	Synthetisch-intuïtief
Werkwijze	Formeel	Informeel
Focus	Efficiëntie Correctheid	Effectiviteit Zinvolheid

Afbeelding 8 Belangrijke aspecten van multidisciplinair samenwerken.

- alle aspecten tot hun recht komen.
- Het toepassen van de reeds eerder genoemde uniforme ontwerp-paradigmen.

Zoals ik reeds heb toegelicht zullen organisaties zich in de toekomst steeds meer op het primaire proces oriënteren en geleidelijk de stap naar een Lean Enterprise met Lean Production maken. De vraag is nu wat voor een soort van Informatie-Management zo'n Lean Enterprise nodig heeft. Volgens het al eerder gebruikte analogie-argument zullen de paradigmata voor de informatieverzorging dezelfde moeten zijn als voor het slank maken van bedrijfsprocessen en organisaties. Ik praat daarom over *Lean Information Management* met als essentiële kenmerken: proces-oriëntatie, object-oriëntatie, klantgerichtheid, teamwork en een verrijnde requirements-engineering fase. Omdat ik hier niet de tijd heb om op details in te gaan verwijs ik naar [Hammer '92] en [Hammer '93c].

Zoals ik al eerder toegelicht heb is de proces-oriëntatie niet alleen voor productie- of gebruiksprocessen belangrijk, maar ook voor ontwerp-processen. Met betrekking tot het maken van software zal men dan ook over *Lean Software Engineering* kunnen spreken [Hammer '93c]. De aan Amerikaanse Carnegie-Mellon universiteit ontwikkelde en door het SEI (Software Engineering Institute) verder uitgebouwde *Software Maturity Model* is een eerste stap in

deze richting ([Humphry '89] en [Humphry et al. '91]). Ook in dit model wordt duidelijk aangegeven, dat het bereiken van de hoogste kwaliteitsniveaus met name een probleem van organisatie-ontwikkeling is. Deze niveaus wachten dan ook nog op hun verdere uitwerking.

De volgende vraag is natuurlijk: wat betekent dat voor ons als technische universiteit? Op de eerste plaats de noodzaak tot bundeling van al bestaand onderzoek, met name tussen de faculteiten Technische Bedrijfskunde en Wiskunde en Informatica. Dit gebeurt bijvoorbeeld in de onderzoeksschool Productie en Logistiek en Kwaliteitszorg waarin ook onze faculteit een essentiële bijdrage levert. Verder moet er natuurlijk ook een opleiding voor het nieuwe gebied van de informatica, dat ik Informatie-Management genoemd heb, opgezet worden. Ik ben blij dat onze faculteit, na wat aarzelingen, samen met de faculteit Technische Bedrijfskunde het initiatief voor een nieuwe studierichting *Informatiesystemen* genomen heeft. Dat dit initiatief om voor mij niet erg heldere redenen in de universiteitsraad gesneuveld is vind ik heel spijtig, want de markt zal niet op de TUE blijven wachten.

Niettemin heb ik goede hoop dat bij alle betrokkenen het inzicht door zal breken dat de industrie naast goede systeembouwers ook informatica-ingenieurs nodig heeft, die meer applicatiegericht zijn en in samenwerking met de toekomstige gebrui-

kers en andere specialisten een
zowel technisch als ook bedrijfsmatig
bevredigende oplossing kunnen
ontwerpen.

Conclusies

Ik vat de belangrijkste conclusies van mijn betoog nog een keer puntsgevijs samen:

1. Efficiënte ondersteuning van een proces door een informatiesysteem eist samenwerking van de verschillende deeltaken. Deze samenwerking kan het best in de vorm van een gedistribueerd systeem gerealiseerd worden. De basis van zo'n gedistribueerd systeem wordt gevormd door gedistribueerde algoritmen.
2. In de toekomst zullen steeds meer associatieve systemen ontstaan omdat zij de voordelen van beide typen van gedistribueerde systemen in zich verenigen: lokaal gaat het om transparantie en strakke coördinatie, interlokaal gaat het om flexibiliteit en openheid.
3. Een gedistribueerd operating-systeem is een belangrijk hulpmiddel voor het realiseren van gedistribueerde systemen. Het abstraheren van de details van de onderlinge hardware en de gedistribueerde natuur van het systeem maakt het programmeren van de applicatie eenvoudiger en productiever.
4. Object-oriëntatie en proces-oriëntatie zijn twee universele paradigma's om de structuur en het gedrag van organisatorische of technische entiteiten te modelleren. Zij ondersteunen het hele (her)-ontwerptraject van de herstructurering van de processen tot en met de implementatie van het ondersteunende informatiesysteem. Het afleiden van specificaties uit het te ondersteunen proces is daarbij net zo belangrijk als het afleiden van een programma uit zijn specificatie.
5. De volgende stap in de ontwikkeling van de vakdiscipline informatica is wat ik Informatie-Management noem. De uitdaging is nu niet meer alleen het bouwen van efficiënte en correcte informatiesystemen, maar het effectief ondersteunen van de bedrijfsprocessen.
6. Lean Enterprise is een belangrijk paradigma voor de volgende stap van de industriële ontwikkeling, dat wil zeggen de stap naar associatieve ondernemingen. Een Lean Enterprise moet door Lean Information Management ondersteund worden. Met betrekking tot het ontwikkelen van software zal Lean Software Engineering in de toekomst steeds belangrijker worden.
7. Na een lange periode van specialisatie zal het in de toekomst steeds meer om integratie gaan, dat wil

zeggen om een evenwichtige benadering vanuit verschillende invalshoeken. Deze variëteit van benaderingswijzen is binnen een vakdiscipline net zo belangrijk als tussen vakdisciplines.

Binnen de informatica gaat het bijvoorbeeld om het evenwicht tussen functionele en dependability eigenschappen, alsmede om een integratie tussen structuur-georiënteerd en proces-georiënteerd ontwerpen. Buiten de informatica gaat het om een multidisciplinaire benadering en met name om de samenwerking tussen organisatiegerichte disciplines en technische disciplines om tot een optimale oplossing voor de klant te komen.

Tenslotte

Dames en heren,

Aan het einde van mijn betoog gekomen, rest mij de aangename taak om mijn dank uit te spreken aan al de mensen die mijn weg binnen en buiten de universiteit begeleid hebben.

Allereerst gaat mijn dank uit naar de leden van het College van Bestuur en het bestuur van de faculteit Wiskunde en Informatica van deze universiteit voor het vertrouwen dat zij in mij hebben gesteld.

Aan de vakgroep Informatica ben ik dank verschuldigd voor de vrijheid die ik had om mijn vakgebied zelf in te vullen. Alle medewerkers van de faculteit en met name mijn collega's wil ik danken voor de goede samenwerking in een sfeer van wederzijds vertrouwen en waardering. Ik hoop dat wij er met z'n allen in slagen om dit klimaat ook in de toekomst te handhaven. Ik wil ook in de komende jaren mijn best doen om samen met mijn collega's actief mee te werken aan de internationale reputatie van deze universiteit en de verbetering van het onderwijs op mijn vakgebied.

Mijn bijzondere dank gaat naar de medewerkers van mijn sectie die met mij samen in de afgelopen jaren het vakgebied hebben opgebouwd.

Daarbij denk ik niet alleen aan de mensen die met mijn eigen specialisme, gedistribueerde systemen en embedded software, bezig zijn, maar ook aan de groep computergraphics die als het ware het tweede standbeen van de sectie Technische Toepassingen vertegenwoordigt.

Ook mijn secretaresse wil ik bij deze dankbetuiging niet vergeten.

Beste Marja, zonder jouw inzet zou de sectie Technische Toepassingen niet staan waar zij vandaag de dag staat. Als co-manager was je voor mij een grote steun; samen met de rest van de sectie ben ik blij met de goede sociale sfeer die je hebt helpen opbouwen.

Beste TTERS, ik kijk uit naar een voortzetting van de samenwerking met jullie en hoop dat die net zo vruchtbaar en plezierig mag zijn als in de afgelopen jaren.

Omdat ik samenwerking tussen de verschillende vakdisciplines zo belangrijk voor de toekomst vind, ben ik bijzonder blij met de vruchtbare samenwerkingsverbanden die ik buiten de faculteit op kon bouwen. Naast veel inspirerende contacten zijn daaruit ook een aantal concrete samenwerkingsprojecten ontstaan. Als gepromoveerd natuurkundige was het voor mij een plezier om samen met de faculteit Natuurkunde de interfacultaire werkgroep Procesbe-

sturing op te kunnen richten. De samenwerking met de faculteit Technische Bedrijfskunde, met name met de vakgroep I&T (Information & Technology), komt voort uit mijn stellige overtuiging dat een van de belangrijke uitdagingen van de informatica de effectieve ondersteuning van bedrijfsprocessen is. Hooggeleerde Bemelmans en Wortmann, ik verheug mij erop met jullie samen dit gebied verder te kunnen bewerken.

Ook de intensieve samenwerking met de industrie wil ik op deze plaats dankbaar vermelden. Zij vormt niet alleen een inspiratiebron voor mijn werk maar geeft ook de mogelijkheid de resultaten aan de praktijk te toetsen. Van de vele werkkringen waarmee ik mij verbonden voel kan ik er op deze plaats alleen enkele noemen. Om te beginnen, mijn eerste werkgever in Nederland, Philips Telecommunication Systems te Hilversum. Voor de ervaringen die ik in dit laboratorium op het gebied van de telecommunicatie en de software-engineering voor embedded systems op mocht doen ben ik nog steeds dankbaar. Ik ben blij dat ik de samenwerking op dit gebied ook met het Natuurkundig Laboratorium van Philips in Eindhoven respectievelijk Aken kan voortzetten. Ook met OCÉ Nederland B.V. te Venlo heb ik een samenwerking van vele jaren met betrekking tot software-engineering voor embedded systems.

Kopieermachines hebben, net als telefooncentrales, een aantal eigen-

schappen, die prachtig als voorbeeld kunnen dienen voor de problemen die men op dit gebied tegenkomt. Op het gebied van de constructie van betrouwbare systemen ben ik gelukkig met het samenwerkingsverband met Fokker Aircraft B.V. in het kader van een Europees project. Tenslotte wil ik hier het jarenlange contact met het TNO-IPL (Instituut voor Productie en Logistiek) vermelden, dat ik op het gebied van de productiebesturing op kon bouwen.

Waarde studenten, ik zie het als een voorrecht om jullie een stuk op jullie weg te mogen begeleiden. Ik heb daar in het verleden veel motivatie en plezier uit geput. Dit geldt zowel voor de vierjarige ingenieursopleiding als ook voor de voortgezette cursussen, zoals de tweejarige ontwerpersopleiding en de vierjarige wetenschappelijke opleiding. Jullie inzet en succes zijn net zo belangrijk als de mijne. Ik zal mij ervoor in blijven zetten om in samenwerking met jullie voor een gedegen en leuk stuk onderwijs te zorgen.

Beste Minne en Loren, ook wij hebben in de loop van de afgelopen 10 jaar een hechte samenwerkingsband ontwikkeld. Zonder jullie tolerantie en medewerking zou het voor mij niet mogelijk geweest zijn om vandaag hier te staan. Ik hoop dat ik mijn liefde en aandacht voor jullie ook in de toekomst met mijn obsessie voor mijn vakgebied kan delen. Ik heb gezegd.

Referenties

[Glasl et al. '93]

F. Glasl und B.C.J. Lievegoed, Dynamische Unternehmensentwicklung: Wie Pionierbetriebe und Bürokratien zu Schlanken Unternehmen werden, Paul Haupt Verlag, Bern, 1993; Freies Geistesleben, Stuttgart, 1993.

[Hammer '91]

D.K. Hammer, The Use of a Distributed Operating System (DOS) for the Construction of Automatic Production Environments, Proc. IFIP WG5.7 Conf. on One-of-a-Kind Production: New Approaches, B.E. Hirsch and K.D. Thoben (Ed't's), North Holland, 1992.

[Hammer '92]

D.K. Hammer, Lean Information Management: The Integrating Power of Information, Proc. IFIP WG5.7 Conf. on Integration in Production Management Systems, H.J. Pels and J.C. Wortmann (Ed't's), North Holland, 1992.

[Hammer '93a]

D.K. Hammer, Synchronization Techniques, Illustrated by the Concepts of the Dependable Distributed Operating System DEDOS, Nato Advanced Study Institute (ASI), Int. Conference on Real-Time Computing, Oct. 1992, St. Maarten, Dutch Antilles.

[Hammer '93b]

D.K. Hammer, Object-Oriented Modelling of Real-Time Systems: What are the Important Issues for the Future?, Workshop on Parallel and Distributed Systems at the 7th Int. Parallel Processing Symposium, Newport Beach, April 1993.

Hammer '93c]

D.K. Hammer, The Development of Large and Complex Software Systems: A Technical Issue?, Workshop on the Development of Large and Complex Software Applications at the Energy-Sources Technology Conference & Exhibition, New Orleans, To be published January 1994.

[Humphry '89]

W.S. Humphry, Managing the Software Process, Addison-Wesley, 1989.

[Humphry et al. '91]

W.S. Humphry, T.R. Snyder and R.R. Willis, Software Process Improvement at Hughes Aircraft, IEEE Computer, July 1991.

[Le Lann '92]

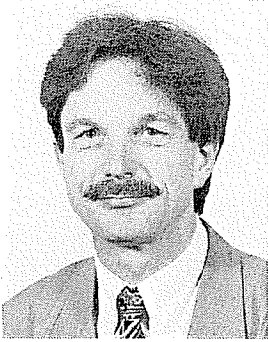
G. Le Lann, Designing Real-Time Dependable Distributed Systems, Computer Communications, Vol. 15, No. 4, May 1992.

- [Laprie '89]
J.C. Laprie, Dependability: A unifying Concept for reliable Computing and Fault Tolerance, in Dependability of Resilient Computers (e.d. T. Anderson), Blackwell Scientific Publications, Oxford, 1989.
- [Litwin et al. '90]
W. Litwin, L. Mark and N. Roussopoulos, Interoperability of Multiple Autonomous Databases, ACM Computing Surveys, Vol. 22, No. 3, 1990.
- [Meyer '88]
B. Meyer, Object-Oriented Software Construction, Prentice-Hall, 1988.
- [NGI '89]
Functies in de technische informatica, Rapport van de Werkgroep Functies Technische Informatica van de Commissie Beroepontwikkeling van de Nederlandse Genootschap voor Informatica, 1989.
- [ODP '92]
ISO / IEC JTC1 / SC21 / WG7 Secretary, Working Documents and Draft Proposals, ANSI, New York, USA.
- [Porter '85]
M.E. Porter, Competitive Advantage, Free Press, 1985.
- [Tanenbaum '92]
A.S. Tanenbaum, Modern Operating Systems, Prentice-Hall, 1992.
- [Verhoosel et al. '91]
J.P.C. Verhoosel, E.J. Luit, D.K. Hammer and E. Jansen, A Static Scheduling Algorithm for Distributed Hard Real-Time Systems, The Journal of Real-Time Systems, 3, 1991.
- [Verhoosel '93]
J.P.C. Verhoosel, A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, to be published as EUT Computing Science Note.
- [TQM '88]
D. Garvin, Managing Quality, Free-Press, 1988.
- [Womack et al. '90]
J.P. Womack, D.T. Jones and D. Roos, The Machine that Changed the World: The Story of Lean Production, Macmillan, 1990.

Vormgeving en druk:
Reproductie en Fotografie van de CTD
Technische Universiteit Eindhoven

Informatie:
Academische en Protocolaire Zaken
Telefoon (040-47)2250/4676

ISBN nr. 903860422X



Dieter Klaus Hammer werd in 1940 geboren te Wenen, Oostenrijk.

Na het behalen van het HBS-diploma werktuigmachinebouw in 1959 studeerde hij technische natuurkunde aan de Technische Universiteit Wenen, waar hij na voltooiing van zijn studie werkzaam was als wetenschappelijke medewerker, 1971 promoveerde en 1978 habiliteerde voor het gebied Natuurkundige Meettechniek. Vanaf 1976 tot 1980 werkte hij als systeemarchitect, software-ingenieur en projectleider bij de afdeling systeemontwikkeling van Philips te Wenen. In deze functie was hij o.a. verantwoordelijk voor de software van de Philips P5000 tekstverwerkingssystemen en de Philips P2000 personal computer. Vanaf 1980 werkte hij bij Philips Telecommunicatie Systems te Hilversum, waar hij eerst projectleider was voor een gedistribueerd operating-

systeem voor een experimentele digitale telefooncentrale en later voor een electronic mail-systeem volgens het CCITT X.400 standaard.

Vanaf 1987 is hij werkzaam als hoogleraar Technische Toepassingen van de Informatica aan de Technische Universiteit Eindhoven.