Intreerede
prof.dr.ir. Twan Basten
15 april 2011

/ Faculteit Electrical Engineering

TU/e Technische Universiteit **Eindhoven** University of Technology

De computer verdwijnt

Where innovation starts

Intreerede prof.dr.ir. Twan Basten

De computer verdwijnt

Uitgesproken op 15 april 2011
aan de Technische Universiteit Eindhoven

De computer van de 21ste eeuw

"De meest ingrijpende technologieën zijn technologieën die verdwijnen. Ze integreren in ons dagelijkse leven totdat ze er niet meer van te onderscheiden zijn."

Dit zijn niet mijn woorden, maar die van wijlen Mark Weiser, in zijn artikel "*The computer for the 21st century*", dat precies twintig jaar geleden verscheen:

"*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*." [22]

De computer verdwijnt snel uit ons dagelijks leven. De eerste computers werden zo'n 70 jaar geleden ontwikkeld. Met de komst van de *personal computer*, de pc, werd de computer gemeengoed. Vandaag de dag zijn er volgens schattingen van Gartner ruim een miljard pc's in gebruik [6]. Hoezo verdwijnt de computer dan, zult u zich afvragen. Slechts een paar procent van de nieuw opgeleverde computers betreft pc's. Verreweg de meeste nieuwe computers zijn ingebedde systemen. Een ingebed systeem (Engels: *embedded system*) is een computersysteem dat is geïntegreerd in gebruiksartikelen of apparaten, met de bedoeling deze een vorm van intelligent gedrag te bezorgen [25]. U kent ze allemaal, van mobiele telefoon tot wasmachine en auto. Een moderne auto bevat al snel enkele tientallen ingebedde systemen. En daarmee raken we de kern van de zaak. De computer is steeds vaker onderdeel van andere apparaten en we zijn ons niet langer bewust van zijn aanwezigheid. *De computer verdwijnt*.

Hoe belangrijk zijn ingebedde systemen in de hedendaagse maatschappij? Daar kunnen we kort over zijn: heel belangrijk. En het belang zal alleen maar toenemen. Een bekend, maar treffend gedachtenexperiment maakt dit direct duidelijk. Stelt u zich voor wat er allemaal niet meer mogelijk zou zijn zonder ingebedde systemen. Communiceren via vaste of mobiele telefoon of via internet zou bijvoorbeeld niet meer kunnen. Auto's en treinen zouden niet meer rijden, vliegtuigen niet meer vliegen. De energievoorziening zou onderbroken worden. Medische zorg zou grotendeels terugvallen naar het niveau van midden vorige eeuw.

Cijfers versterken dit beeld. De wereldmarkt voor ingebedde systemen heeft inmiddels naar schatting een volume van ruim US$ 100 miljard bereikt [3]. Schattingen van de jaarlijkse groei van deze markt lopen uiteen van 4% tot 14% [3,5]. De jaarlijkse groei van het bruto nationaal product van de EU-landen in het decennium voor de financiële crisis bedraagt slechts 2.5% [4]. De markt voor ingebedde systemen draagt daar dus buitenproportioneel aan bij. Zo'n 30% van de waarde van een auto komt vandaag de dag voort uit ingebedde systemen [1]. In 2015 zullen er zo'n 40 miljard ingebedde systemen in gebruik zijn [5]. Dat komt neer op zo'n vijf systemen per persoon, waarbij we rekening houden met de groei van de wereldbevolking tot zo'n 7.5 miljard mensen.

We kunnen concluderen dat ingebedde systemen maatschappelijk en economisch steeds belangrijker worden. Uiteindelijk zal de computer verdwijnen en volledig geïntegreerd worden in de apparaten en gebruiksvoorwerpen in onze dagelijkse omgeving. Stemt dit beeld u vrolijk? Ik vermoed dat velen van u gemengde gevoelens zullen hebben bij dit toekomstbeeld. Computers vandaag de dag schieten te kort op enkele essentiële aspecten: gebruiksgemak, veiligheid en betrouwbaarheid. Zonder de eerste twee aspecten tekort te willen doen, zal ik het in deze rede vooral hebben over het derde aspect, betrouwbaarheid.

Te vertrouwen systemen

Wat is betrouwbaarheid in de tot zover geschetste context? In het vakgebied worden de Engelse termen *reliability* en *dependability* gebruikt. Een *reliable system*, een betrouwbaar systeem, is een systeem dat correct functioneert. Het doet wat het verwacht wordt te doen. Een *dependable system* is letterlijk zoiets als 'een systeem waarvan men afhankelijk zou durven zijn', ofwel een *te vertrouwen systeem*. *Dependability* [2] omvat daarmee meer dan alleen het correct functioneren van het systeem. Het omvat ook aspecten zoals beschikbaarheid van het systeem, onderhoudbaarheid, en de zekerheid dat de nadelige gevolgen beperkt blijven, mocht er onverhoopt iets stuk gaan.

Laten we de auto als voorbeeld nemen. Een auto functioneert doorgaans zoals verwacht zonder problemen. Betrouwbaarheid en beschikbaarheid zijn in orde. Het benodigde onderhoud is goed te overzien en mocht er iets stuk gaan, dan is reparatie doorgaans eenvoudig tegen beperkte kosten. Zelfs bij ongelukken is de schade tegenwoordig vaak beperkt, dankzij aspecten als kooiconstructies en airbags. Een auto is een systeem waarop we kunnen vertrouwen en ook durven te vertrouwen.

Vliegtuigen en medische apparatuur zijn andere voorbeelden van systemen waarin we doorgaans vertrouwen hebben. En om misverstanden te voorkomen, mijns inziens is dit vertrouwen ook terecht. Pc's, en recent de ov-chipkaart, zullen velen niet vertrouwen. Daarvoor werken ze eenvoudigweg niet goed genoeg. Een pc is bovendien veel te kwetsbaar voor bijvoorbeeld virussen, en de ov-chipkaart is te fraudegevoelig en heeft te veel administratieve problemen. Ook beveiligingssystemen voor tunnels of spoor zijn bekende probleemgevallen als we het hebben over betrouwbaarheid.

Het is interessant om ook even stil te staan bij veranderingen in de ervaren betrouwbaarheid van systemen. De betrouwbaarheid van pc's is de afgelopen 30 jaar sterk verbeterd. Desalniettemin blijft het beeld bestaan dat pc's onbetrouwbaar zijn. Voor auto's lijkt de laatste jaren juist een verandering in tegenovergestelde richting op te treden. We zien steeds vaker dat grote aantallen auto's door de fabrikanten teruggeroepen worden, om een grote verscheidenheid aan

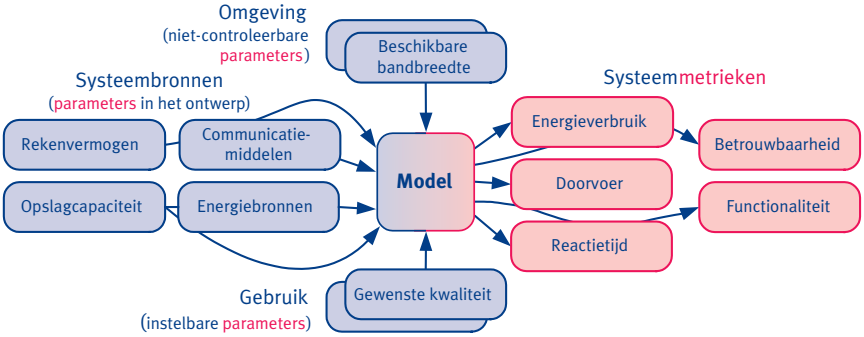
redenen. De consequenties lijken vooralsnog beperkt. De fabrikant met de meeste terugroepacties zag de verkopen in 2010 desondanks sterk toenemen [21]. De fabrikanten lopen echter grote risico's. Als het beeld over betrouwbaarheid eenmaal omslaat, dan zal het niet eenvoudig zijn om het tij weer te keren.

In alle genoemde voorbeelden spelen computers en ingebedde systemen een belangrijke rol. We kunnen twee belangrijke conclusies trekken. Ten eerste, beeldvorming speelt een cruciale rol in de ervaren betrouwbaarheid van systemen. Ten tweede, het is blijkbaar moeilijk om betrouwbare systemen te realiseren. Op het eerste punt zal ik later nog terugkomen. Ik wil nu eerst ingaan op het tweede punt. Waarom is het zo moeilijk om betrouwbare systemen te maken? Hierover valt veel te zeggen. Het komt er echter uiteindelijk eenvoudigweg op neer dat we de systemen die we bouwen onvoldoende goed begrijpen.

Computermodellen

"*Knowing is not understanding*." Charles Kettering

Weten is niet hetzelfde als begrijpen. We weten veel van computersystemen, maar we begrijpen ze niet. We zullen systemen pas vertrouwen als we ze begrijpen. En begrip begint bij modelvorming.



Figuur 1

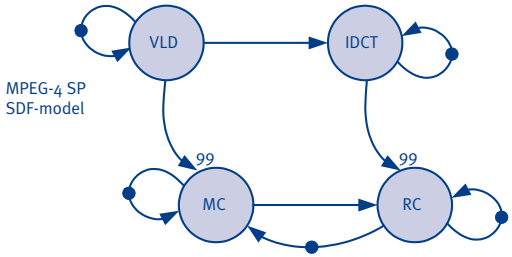
Een computermodel relateert systeemmetrieken aan parameters van het systeem.

Wat is een model? En wat kunnen we ermee?

Een model is een vereenvoudigde voorstelling, beschrijving of nabootsing van een deel van de werkelijkheid [26]. Als we het over computersystemen hebben, dan kunnen we dit nader preciseren. Doorgaans hebben we het dan over goed gedefinieerde, vaak wiskundige, beschrijvingen van computers en de wijze waarop deze bewerkingen uitvoeren. Zo'n computermodel beschrijft hoe eigenschappen van het systeem, vaak metrieken genoemd, afhangen van de variabelen in het systeem, doorgaans parameters genoemd (Figuur 1). Een model kan bijvoorbeeld beschrijven hoe het energieverbruik van een intelligente telefoon voor het vertonen van een video afhangt van de gewenste beeldkwaliteit en het benodigde rekenvermogen.

Waarom maken we modellen? Hier zijn vele antwoorden mogelijk. Om systemen beter te begrijpen, om de functionaliteit van het systeem te controleren voordat het systeem gebouwd wordt, om de prestaties van het systeem door te rekenen, of om een realisatie automatisch te genereren. Grofweg samengevat maken we modellen voor *analyse*doeleinden, het afleiden van eigenschappen van een systeem, en voor *synthese*doeleinden, het constructief genereren van een systeem.

Laten we ter illustratie een *dataflow*-model van een MPEG-4 SP decoder bekijken (Figuur 2). Het betreft een zogenoemd *Synchronous DataFlow* (SDF, [13]) model. MPEG-4 [14] is een standaard om audio en video te coderen, bijvoorbeeld voor gebruik in mobiele telefoons. Een SDF-model is doorgaans relatief eenvoudig. Het beschrijft de uit te voeren taken, middels actoren, de grote blauwe cirkels in de figuur, en de afhankelijkheden daartussen, middels kanalen, de pijlen in de figuur. Het beschrijft verder de geproduceerde en geconsumeerde eenheden data per actorexecutie, aangegeven middels de cijfers bij de uiteinden van de kanalen, waarbij het cijfer 1 doorgaans weggelaten wordt. Een executie van de MC-actor bijvoorbeeld, consumeert 99 eenheden data uit het kanaal tussen VLD en MC. Een VLD-executie produceert 1 eenheid data in dit kanaal. De op enig moment in kanalen aanwezige eenheden data worden middels de kleine blauwe cirkels in de figuur weergegeven. Een SDF-model in pure vorm abstraheert van vele aspecten, bijvoorbeeld van de exacte functionaliteit van de actoren, van de waardes van data, van de tijd nodig voor de executie van een actor, en van het energieverbruik.



executietijd (µs)	
VLD	33
IDCT	14
MC	325
RC	292

analyseresultaat	maximaal te garanderen doorvoer is 306 beelden per seconde

Figuur 2

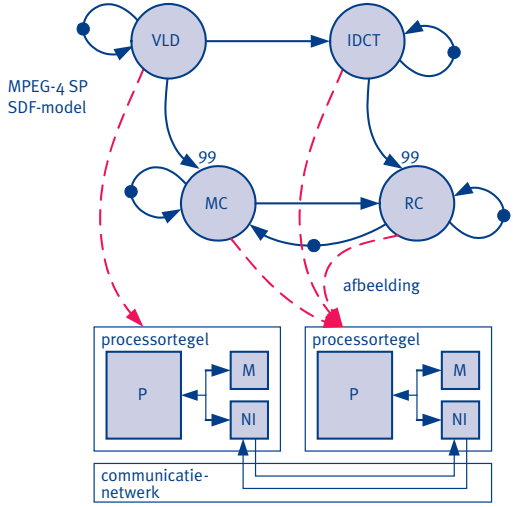
Een SDF-model van een MPEG-4 SP decoder. De gegeven actorexecutietijden zijn ter illustratie. Het zijn geschatte bovengrenzen voor de werkelijke executietijden op een typische moderne ingebedde processor. Uitgaande van deze bovengrenzen is maximaal een doorvoergarantie van 306 beelden per seconde mogelijk.

Als in aanvulling op het SDF-model de executietijden van actoren bekend zijn, of bovengrenzen daarvoor, dan is het mogelijk om de doorvoer te berekenen [8]. De doorvoermetriek is gedefinieerd als de hoeveelheid verwerkte data per tijdseenheid. Het model wordt dan gebruikt voor *analyse*. Op basis van de voor het MPEG-4-voorbeeld geschatte bovengrenzen voor de actorexecutietijden kan een doorvoer van 306 beelden per seconde gegarandeerd worden. De analyse neemt slechts een fractie van een seconde rekentijd op een normale laptop in beslag. De te halen doorvoer is ruimschoots voldoende voor gebruik in een mobiele telefoon, waar een doorvoer van bijvoorbeeld 15 beelden per seconde al acceptabel kan zijn. Als we de decoder daadwerkelijk gaan implementeren, dan zal de maximaal mogelijk te garanderen doorvoer echter doorgaans sterk afnemen, bijvoorbeeld omdat processors tussen actoren gedeeld moeten worden en omdat uitwisseling van data tussen actoren op verschillende processors tijd kost. Het resultaat van de doorvoeranalyse op het SDF-model laat echter zien dat er voldoende ruimte is om het realisatietraject in te gaan.

Ter illustratie van een *synthese*traject bekijken we een model van de MPEG-4 SP decoder zoals die afgebeeld is op een multiprocessorplatform met twee processors met lokale geheugens en verbonden middels een communicatienetwerk (Figuur 3). Ook het platform en de afbeelding van de decoder op dat platform worden gerepresenteerd middels modellen. Het platformmodel legt bijvoorbeeld de aantallen en typen van processors vast, de organisatie en grootte van geheugens en de snelheid van het communicatienetwerk. In dit voorbeeld gaan we uit van twee processors van hetzelfde type, met ieder een eigen geheugen. We gaan bovendien uit van een communicatienetwerk dat bandbreedtegaranties kan geven. Dat wil zeggen dat de tijd die nodig is om een bepaalde hoeveelheid data te communiceren begrensd is en dat deze tijd vooraf berekend kan worden. Het model van de afbeelding plaatst de VLD-actor op één van de processors, terwijl de andere drie actoren de andere processor delen. De executietijden van de actoren worden uit deze afbeelding afgeleid en dienen een bovengrens te zijn voor de werkelijke executietijden op de betreffende processors.

Gegeven een applicatiemodel en een platformmodel kan een synthesetraject voor een gewenste doorvoergarantie een afbeelding van de applicatie berekenen met de bijbehorende benodigde systeembronnen. De in de figuur geïllustreerde afbeelding is bijvoorbeeld het resultaat van een bestaand synthesetraject [18].

De afbeelding geeft een doorvoergarantie van 15 beelden per seconde, wat voldoende is voor typische toepassing in mobiele telefoons. Om deze garantie te kunnen geven, dient 88% van de twee processors en 33% van de beschikbare netwerkbandbreedte gereserveerd te worden; de gezamenlijke capaciteit van de beide geheugens moet tenminste 254 KB bedragen.



executietijd (µs)	
VLD	33
IDCT	14
MC	325
RC	292

syntheseresultaat	
doorvoer-garantie	15 (beelden per seconde)
processor-capaciteit	88 %
geheugen-capaciteit	254 KB
bandbreedte	33 %

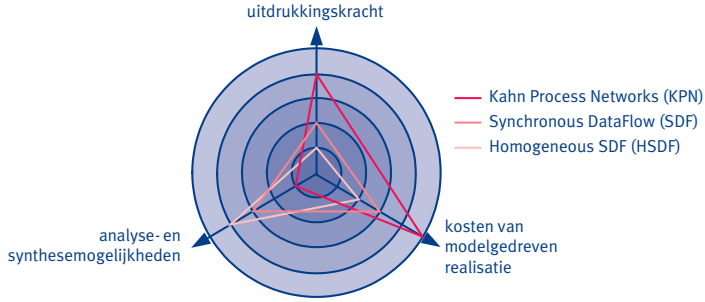
Figuur 3 (P: processor; M: geheugen; NI: NetwerkInterface)

Een model van de MPEG-4 SP decoder zoals die afgebeeld is op een platform met twee processors. Een synthesetraject levert zowel de gegeven afbeelding op, als de hoeveelheden systeembronnen die nodig zijn om een gewenste doorvoergarantie te realiseren.

Het belang van goede modellen

Het MPEG-4-voorbeeld schetst de essentie van wat modelgedreven ontwerp genoemd wordt. In een modelgedreven ontwerptraject zijn modellen leidend in het gehele traject; ontwerpen op verschillende niveaus van abstractie en detail zijn te allen tijde consistent met de modellen. Een van de belangrijkste uitdagingen in dergelijke ontwerptrajecten is de complexiteit van het geheel. Er zijn onvoorstelbaar veel mogelijke realisaties van een gegeven productidee. Het is onmogelijk om al deze realisaties in detail te bekijken. Zelfs als belangrijke keuzes, zoals in het geschetste voorbeeld de opsplitsing van een applicatie in actoren en de organisatie van het platform, al gemaakt zijn, dan nog blijft het aantal mogelijke realisaties te groot om ze allemaal in detail te overwegen. Het geschetste synthesetraject geeft dan ook geen garanties betreffende de optimaliteit van de gevonden afbeelding. Het geeft slechts garanties over de doorvoer die behaald

kan worden met deze afbeelding. Later kom ik terug op het complexiteitsvraagstuk. Ik ga nu eerst in op een andere belangrijke uitdaging in modelgedreven ontwerp, namelijk de ontwikkeling van goede modellen.

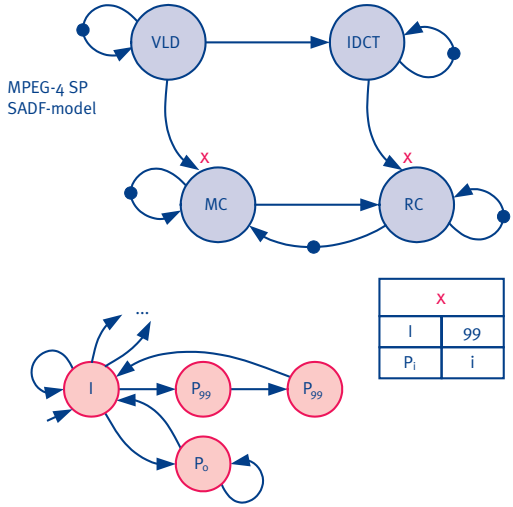


Figuur 4

Conflicterende wensen aan modellen. Eenvoudige weergave voor drie bekende modelleringstechnieken: *Kahn Process Networks* [12], SDF [13] en *Homogeneous* SDF [13].

Een belangrijk aspect in SDF-modellen is het feit dat actorexecutietijden constant zijn. Dit heeft als voordeel dat analyse en modelgedreven synthese van implementaties relatief eenvoudig zijn. Een nadeel is echter dat modelgedreven implementaties onnodig conservatief kunnen zijn. Als voor video- en audiotoepassingen bijvoorbeeld data-afhankelijke variaties in actorexecutietijden meegenomen worden in modellering en synthesetraject, dan leidt dit tot modelgedreven implementaties die efficiënter omgaan met de systeembronnen [19]. Een verfijnder model gaat echter doorgaans ten koste van analyse- en synthesemogelijkheden (Figuur 4). Ook de kosten van modelgedreven realisatie nemen doorgaans toe, omdat bijvoorbeeld meer informatie meegenomen moet worden in het vastleggen, plannen en uitvoeren van de executie van een applicatie. Alleen een bij de ontwerpsituatie passend compromis tussen al deze aspecten leidt uiteindelijk tot efficiënte modelgedreven implementaties.

Ter illustratie beschouwen we een *Scenario-Aware DataFlow* (SADF) [20] model van de MPEG-4 SP decoder (Figuur 5). Een SADF-model onderscheidt verschillende executiescenario's. Per scenario kunnen actorexecutietijden en de geconsumeerde en geproduceerde hoeveelheden data verschillen. Het MPEG-4 SADF-model onderscheidt negen scenario's, op basis van de wijze waarop een beeld gecodeerd is (I- of P-codering) en op basis van de mate van beweging van beeldobjecten tussen beelden. De tabel rechts in de figuur geeft actorexecutietijden per scenario. Parameter *x* representeert de variabele hoeveelheden gecommuniceerde data.



x	
I	99
P_i	i

executietijd (µs)		
VLD	P_0	0
	anders	33
IDCT	P_0	0
	anders	14
MC	I,P_0	0
	P_{30}	75
	P_{40}	121
	P_{50}	158
	P_{60}	196
	P_{70}	221
	P_{80}	258
	P_{99}	325
RC	I	292
	P_0	0
	P_{30}, P_{40}, P_{50}	208
	P_{60}	250
	P_{70}, P_{80}, P_{99}	267

Figuur 5

Een SADF-model van de MPEG-4 SP decoder. Het model onderscheidt negen verschillende executiescenario's (I, P_0, P_{30}, P_{40}, P_{50}, P_{60}, P_{70}, P_{80}, P_{99}). Actorexecutietijden en de geconsumeerde en geproduceerde hoeveelheden data kunnen per scenario verschillen.

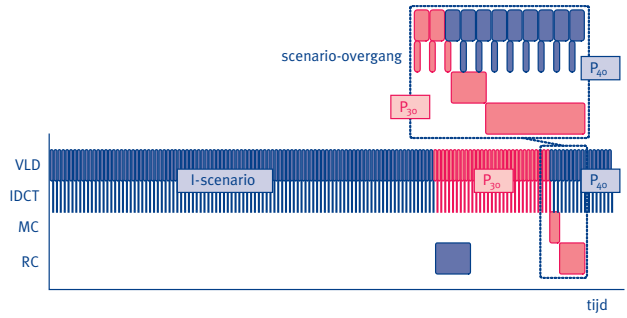
Het toestandsdiagram linksonder legt vast in welke volgordes scenario's kunnen optreden. De ingekleurde vakjes in de tabel met executietijden laten zien dat de langste executietijden voor de verschillende actoren in verschillende scenario's voorkomen. Dit is één van de redenen waarom het SADF-model een beter syntheseresultaat levert. Afbeelding op het tweeprocessorplatform uit het voorbeeld met behulp van een SADF-synthesetraject [19] geeft, voor dezelfde doorvoergarantie, een besparing van 70% op de benodigde processorcapaciteit (Figuur 6). Eenzelfde vergelijking voor een MP3-toepassing, de bekende audiostandaard, laat besparingen zien voor geheugen en bandbreedte (Figuur 6, rechts). Deze voorbeelden zijn illustratief. De systeembronnen waarop bespaard kan worden en de omvang van de besparing, hangt van vele aspecten af. De belangrijkste conclusie is dat er bespaard wordt. Daarbij verandert er niets in de functionaliteit van de uiteindelijke systemen. Beeld- en geluidskwaliteit bijvoorbeeld blijven gelijk. De besparingen worden behaald door het gebruik van goed passende modellen, SADF in plaats van SDF, en een aangepast synthesetraject. De besparingen, feitelijk dus de verfijnde modellen, leiden uiteindelijk tot goedkopere en energiezuinigere systemen, aangezien prijs en energieverbruik direct samenhangen met de gebruikte systeembronnen.

MPEG-4 SP				MP3
metriek	sdf	sadf	verbetering	verbetering
doorvoer-garantie	15 (beelden per seconde)	15 (beelden per seconde)	NVT	NVT
processor-capaciteit	88 %	26 %	70 %	-
geheugen-capaciteit	254 KB	254 KB	-	21 %
bandbreedte	33 %	33 %	-	23 %

Figuur 6

Door gebruik van SADF-modellen in plaats van SDF-modellen kunnen doorvoergaranties gegeven worden met een lager gebruik van systeembronnen. Links: vergelijking voor het MPEG-4-SP-voorbeeld. Rechts: besparingen voor een MP3-voorbeeld.

Een SADF-model van een applicatie is complexer dan een SDF-model van dezelfde applicatie. Het is mogelijk om systeemgedrag met een behoorlijke mate van detail te beschrijven. Figuur 7 laat een executie van het beschreven MPEG-4 SADF-model zien, uitgaande van een onbeperkte hoeveelheid beschikbare systeembronnen en instantane communicatie. Zelfs onder deze aannames is het gedrag complex.
We zien opeenvolgende scenario's die overlappen in tijd en actoren die in parallel executeren, daarin gestuurd door de te communiceren data, en met per scenario verschillende executietijden. De combinatie van deze aspecten maakt de analyse van eigenschappen zoals doorvoer en het synthesetraject voor multiprocessor-platformen tot een uitdaging. Ook de kosten die gemoeid zijn met de executie, bijvoorbeeld voor het claimen en weer vrijgeven van processors, zijn hoger. De mate van detail in een SADF-model blijkt voor multimedia-applicaties echter cruciaal voor de uiteindelijke efficiëntie van het syntheseresultaat. Daarmee illustreert



Figuur 7

Executie van het MPEG-4 SADF-model. Actoren executeren in parallel en scenario's overlappen in tijd. Deze combinatie vormt een uitdaging voor analyse en synthese.

de SDF-SADF-vergelijking precies het dilemma tussen enerzijds uitdrukkingskracht van modellen en anderzijds analyse- en synthesecomplexiteit en kosten van realisatie. Alleen een passende keuze geeft een goed eindresultaat.

Uitdagingen

SADF is bedacht en ontwikkeld in de *Electronic Systems* groep van de faculteit *Electrical Engineering*, de groep waarvan ook ik deel mag uitmaken. Het geschetste SADF-synthesetraject laat de potentie van modelgedreven ontwerp zien: kosten- en energie-efficiënte systemen met een gegarandeerde betrouwbaarheid en kwaliteit. Modelgedreven ontwerp heeft bovendien de potentie om ontwerptrajecten te versnellen, omdat er vanwege de verbeterde efficiëntie en kwaliteit minder iteraties in het ontwerpproces nodig zijn. Modelgedreven ontwerp is dan ook het belangrijkste onderwerp waar ik de komende jaren aan wil blijven werken.

Modelgedreven ontwerp van ingebedde systemen wordt vandaag de dag nog maar beperkt toegepast in de industriële praktijk. Er resteren verschillende uitdagingen. De huidige modellerings-, analyse- en synthesetechnieken zijn niet krachtig genoeg voor grootschalige toepassing. De belangrijkste uitdaging is de ontwikkeling van nieuwe, efficiënte analyse- en synthesemogelijkheden voor modellen met een grotere uitdrukkingskracht, zonder dat dit leidt tot een grote toename in de kosten van modelgedreven realisatie. Een complicerende factor is dat computersystemen continu in ontwikkeling zijn. Nieuwe ontwikkelingen in bijvoorbeeld communicatietechnologie en materialen staan ons toe om almaar complexere systemen te ontwerpen. Een belangrijke tweede uitdaging is dan ook het beheersbaar maken van de complexiteit van systemen en de bijbehorende ontwerptrajecten, waarover dadelijk meer.

In de aanpak van deze uitdagingen wil ik op zoek gaan naar synergie tussen en integratie van modelleringstechnieken. SADF bijvoorbeeld introduceert geen fundamenteel nieuwe concepten, maar combineert de essentiële elementen van dataflow-modellering, met actoren als het belangrijkste concept, en toestandsmodellering, met, de naam zegt het al, toestanden als het belangrijkste concept. SADF is intuïtief, conceptueel elegant en krachtig tegelijk, en illustreert daarmee de synergie die te bereiken is door integratie van concepten uit verschillende modelleringstechnieken.

Conceptuele integratie zal niet alleen leiden tot een beter begrip van verschillende technieken en hun potentie, maar het is ook een noodzakelijke voorwaarde voor de acceptatie van modelgedreven ontwerp in de industriële praktijk. Het belang van draagvlak en standaardisatie wordt duidelijk als we kijken naar gerelateerde ontwikkelingen uit het verleden, bijvoorbeeld betreffende programmeertalen en besturingssystemen. Daaruit blijkt telkens dat uiteindelijk slechts een beperkt aantal oplossingen breed wordt geaccepteerd. Dat zal ook gelden voor modelgedreven ontwerp. Alleen breed gedragen oplossingen, met bijvoorbeeld voldoende ondersteuning in de vorm van ontwikkelgereedschappen, zullen een kans van slagen hebben. Om dit draagvlak te creëren, zal de meerwaarde van modelgedreven ontwerp duidelijk moeten zijn. Het is mijn doel om deze meerwaarde zichtbaar te maken en te vergroten, en zo bij te dragen aan de acceptatie van modelgedreven ontwerp in de industriële praktijk.

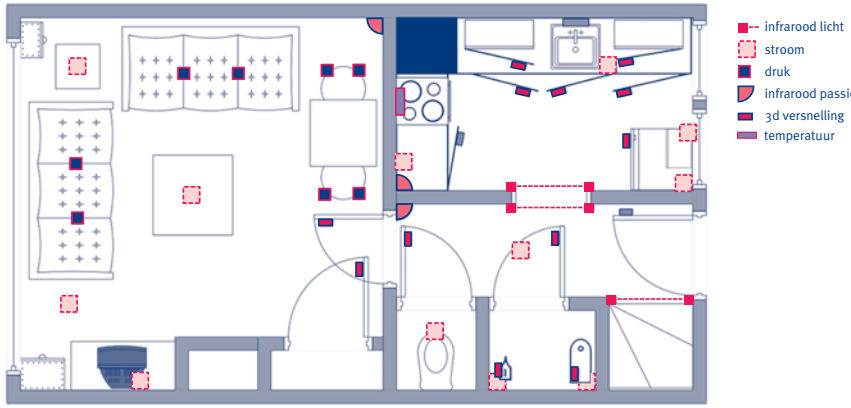
Ten slotte wil ik werken aan goed en aantrekkelijk modelleeronderwijs. Ook dat begint met de meerwaarde van modelgedreven ontwerp. Onderwijs dat enthousiasmeert zal bijdragen aan de praktische toepassing van modelgedreven ontwerp. Idealiter ontstaat er een wisselwerking tussen goed modelleeronderwijs en toepassing van modelgedreven ontwerp. Goed modelleeronderwijs dient zich te richten op de belangrijkste concepten en niet te veel op specifieke formalismen, aangezien die continu in ontwikkeling zijn. Verder is het essentieel om modelleervaardigheden aan te leren. Daarbij gaat het om conceptueel, logisch en abstract redeneren, het formuleren van wiskundige modellen, het vertalen van een vraag- of probleemstelling naar een goed model in een passend formalisme, en het redeneren over en met behulp van modellen. Het is te laat om hier op de universiteit mee te beginnen. Elementaire modelleervaardigheden zijn voor iedereen bruikbaar en dienen integraal in het hele onderwijstraject ontwikkeld te worden.
Ook daar wil ik me in de toekomst voor inzetten.

Complexiteit

De complexiteit van ingebedde computersystemen en de bijbehorende ontwerptrajecten is al enkele keren ter sprake gekomen. Deze complexiteit dient beheersbaar te worden om te komen tot betrouwbare systemen.

Bronnen van complexiteit

Waar komt de complexiteit van ingebedde systemen vandaan? Een eerste belangrijke bron van complexiteit zijn de ontwerpparameters van een systeem. Figuur 8 laat ter illustratie het schema zien van een experimenteel netwerk van sensorknopen in een thuisomgeving. Het doel is om de activiteiten van een COPD-patiënt in de thuisomgeving te herkennen en te monitoren, zonder daadwerkelijk aan het lichaam te meten. COPD is een longaandoening. Patiënten zijn gebaad bij activiteit, maar een overdaad aan activiteit is schadelijk. Coaching van COPD-patiënten kan veel ellende en veel medische kosten voorkomen. Sensornetwerktechnologie zou daarbij kunnen helpen. Voor de verdieping die de figuur laat zien, bestaat het netwerk uit 42 sensorknopen. Deze sensorknopen doen verschillende typen metingen en communiceren deze metingen draadloos naar een pc. De pc leidt hieruit de activiteiten van de patiënt af, en geeft feedback aan de patiënt.
Het zendvermogen waarmee informatie van de sensorknopen naar de pc verstuurd wordt, bepaalt in belangrijke mate de afstand waarover gecommuniceerd kan worden, de kwaliteit van de communicatie en het energieverbruik. Verschillende configuraties van het netwerk verschillen daarmee in snelheid en kwaliteit van communicatie, en de energie die nodig is om alle data te verzamelen. Uitgaande van alleen de parameter zendvermogen, en uitgaande van vier mogelijkheden per sensorknoop, kan het netwerk op 4^{42} verschillende manieren geconfigureerd worden. Dat komt neer op ongeveer $1.9 \cdot 10^{25}$ mogelijkheden. Ter vergelijking, het universum is naar schatting zo'n 14 miljard jaar oud, wat neerkomt op zo'n $4.4 \cdot 10^{17}$ seconden. Ingebedde systemen hebben doorgaans vele ontwerpparameters. Deze parameters leiden tot een veelvoud aan configuratiemogelijkheden. Een ontwerptraject voor ingebedde systemen zal op een slimme manier met het grote aantal configuratiemogelijkheden om moeten gaan.



Figuur 8 [Roessingh Research and Development, 2010 [15]]

Een netwerk van 42 sensorknopen met elk vier instellingen voor het zendvermogen kan op 4^{42} ($\approx 1.9 \cdot 10^{25}$) verschillende manieren ingesteld worden.

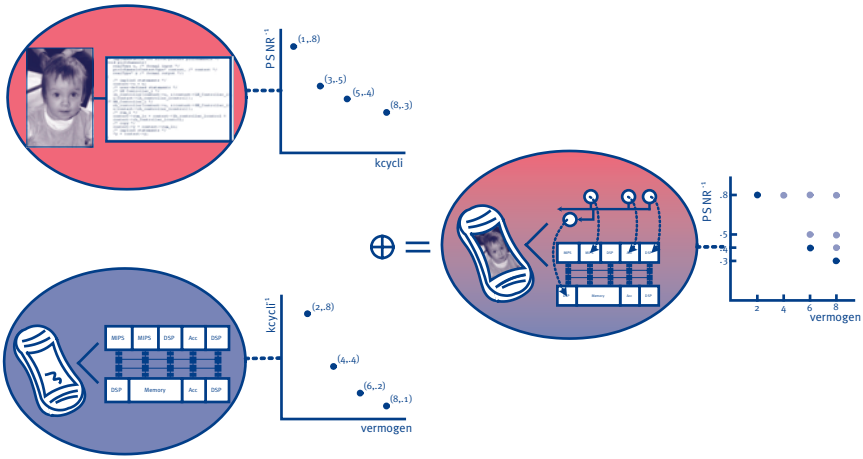
Een tweede belangrijke bron van complexiteit is parallellisme, bijvoorbeeld tussen processen op verschillende processors in een multiprocessorplatform, die beide lees- en schrijfacties op een gezamenlijk geheugen uitvoeren. Het aantal mogelijke executievolgordes van drie parallelle processen die elk twee opeenvolgende acties uitvoeren is (2·3)!/2!3 [16], wat gelijk is aan 90. Het aantal executievolgordes wordt snel groter als het aantal processen en acties groter wordt. Vier processen die elk tien acties uitvoeren, kunnen al op ruim $4.7 \cdot 10^{21}$ verschillende manieren uitgevoerd worden. De volgorde van uitvoering bepaalt doorgaans de uitkomsten en prestaties van het geheel.

Een derde bron van complexiteit is de al in het MPEG-4-voorbeeld geschetste afhankelijkheid van systeemprestaties van externe inputdata. Andere bronnen zijn variaties in productie, van bijvoorbeeld de chips die het hart vormen van elk ingebed systeem, het feit dat ontwerpprocessen inherent multidimensionaal zijn, het feit dat almaar meer functionaliteit geïntegreerd wordt in systemen, zoals intelligente telefoons, en het feit dat steeds meer systemen door middel van netwerken verbonden zijn. Trends zijn bovendien zodanig dat de complexiteit op basis van al deze aspecten alleen maar verder zal toenemen.

Hoe maken we complexiteit beheersbaar?

Ten eerste, door goed gebruik te maken van structuur die altijd in een te ontwerpen systeem aanwezig is. Ten tweede, door de juiste abstracties toe te passen.

De complexiteit van een ontwerptraject kan gereduceerd worden door een te ontwerpen systeem in componenten te structureren, deze componenten te ontwerpen en vervolgens deze componenten samen te stellen. Dit componentgebaseerd ontwerp klinkt eenvoudig, maar dat is het helaas niet. Het vereist dat de eigenschappen van componenten zeer precies vastgelegd worden, om problemen tijdens de integratie van componenten tot een systeem te voorkomen. Om daadwerkelijke reductie van complexiteit te bereiken, is het bovendien nodig dat componenten samenstelbaar zijn. Dat wil zeggen dat functionaliteit en prestaties van een component zonder detailkennis van het omvattende systeem bepaald kunnen worden en behouden blijven bij gebruik van een component in een groter systeem. Bovendien is compositionaliteit een vereiste. Het dient mogelijk te zijn om de eigenschappen van een systeem af te leiden uit de eigenschappen van de samenstellende componenten. Omgekeerd moet het mogelijk zijn om de gewenste eigenschappen van een systeem te vertalen naar gewenste eigenschappen van de componenten.



Figuur 9

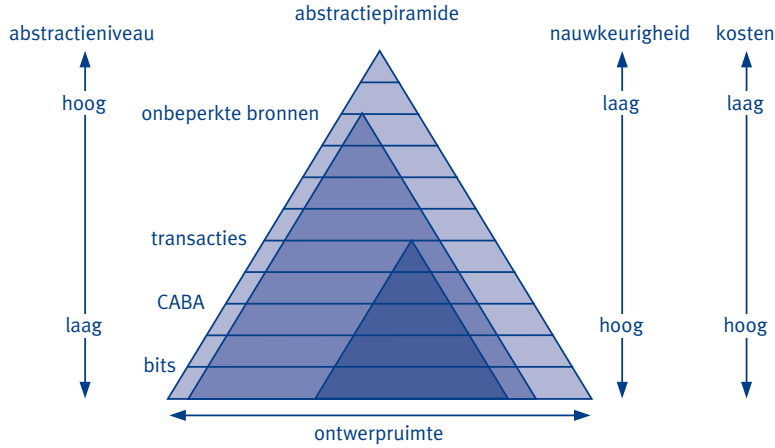
Compositionele berekening van configuratieopties in een vereenvoudigd voorbeeld van een intelligente telefoon. Gegeven optimale configuraties van een videoapplicatie en een ingebed platform kunnen de optimale configuraties van het systeem als geheel berekend worden.

Ter illustratie bekijken we een vereenvoudigd voorbeeld van compositioneel redeneren over configuraties van een intelligente telefoon (Figuur 9). Gegeven zijn vier optimale configuraties voor een videoapplicatie in twee dimensies, videokwaliteit en benodigde rekenkracht, en vier optimale rekenkracht-vermogen-configuraties voor een multiprocessorplatform. Hieruit kunnen de optimale configuraties voor het gehele systeem in termen van videokwaliteit en verbruikt vermogen berekend worden, waarbij gevraagde en geleverde rekenkracht passend moet zijn. Deze berekening kan efficiënt uitgevoerd worden met behulp van algebraïsche technieken [7]. Deze technieken maken het mogelijk om grote configuratieruimtes efficiënt te doorzoeken naar optimale configuraties, onder de voorwaarde dat er een zekere mate van structuur in deze ruimtes aanwezig is. Als we teruggaan naar het sensornetwerkvoorbeeld (Figuur 8) en aannemen dat data via een boomstructuur naar de pc gecommuniceerd wordt, dan kunnen we deze algebraïsche technieken gebruiken om via deze structuur de optimale netwerkconfiguraties te bepalen [9]. Zelfs voor hele grote ruimtes kost dit doorgaans niet meer dan enkele minuten op een gewone pc. De berekeningen zijn zelfs zo efficiënt dat configuraties van een operationeel systeem continu geoptimaliseerd kunnen worden [17].

Componentgebaseerd ontwerp, samenstelbaarheid en compositionaliteit zijn veelbelovende concepten om complexiteit hanteerbaar te maken, met in potentie nog vele andere voordelen, zoals een betere herbruikbaarheid van componenten, versnelde ontwikkeltrajecten en goedkopere producten. Er is echter nog het nodige fundamenteel-theoretisch onderzoek nodig en daarnaast ook een verdere praktische ontwikkeling om deze concepten toepasbaar te maken voor de ontwerpproblemen van de hedendaagse industrie van ingebedde systemen. Het is bijvoorbeeld doorgaans moeilijk om extra-functionele aspecten zoals prestatie en betrouwbaarheid op compositionele wijze te bepalen. Een belangrijk praktisch aspect dat aandacht verdient, is hoe om te gaan met bestaande componenten.

Een tweede belangrijk concept om complexiteit beheersbaar te maken is abstractie. Abstractie is het weglaten van niet-essentiële informatie om zodoende meer fundamentele structuren zichtbaar te maken [23]. Abstractie is dan ook sterk gerelateerd aan het zoeken naar structuren. Bovendien is er een sterke relatie met het ontwerpproces. In elke fase van het ontwerpproces dient de juiste abstractie gekozen te worden (Figuur 10). Hogere abstractieniveaus maken het mogelijk om grotere delen van de ontwerpruimte te doorzoeken, tegen relatief lage kosten in tijd en geld. De nauwkeurigheid is doorgaans echter ook lager dan op lagere abstractieniveaus. Het is daarom essentieel om vroeg in het ontwerpproces op hogere abstractieniveaus de juiste modellen met de juiste abstracties te kiezen.

Op basis van deze modellen kunnen dan goede ontwerpkeuzes gemaakt worden, waarna het vervolgtraject beperkt kan worden tot de interessante delen van de ontwerpruimte. Die kunnen dan verder doorzocht worden middels meer gedetailleerde modellen, enzovoort. Als dit proces goed ingericht wordt, dan zal de kwaliteit van het eindproduct verbeteren en het ontwerpproces versnellen.



Figuur 10

Abstractiepiramide. Op hogere abstractieniveaus kan een groter deel van de ontwerpruimte doorzocht worden, tegen lagere kosten, maar wel met een lagere nauwkeurigheid.

De uitdaging is het vinden van passende abstracties. De abstracties op lagere abstractieniveaus zijn breed geaccepteerd. De belangrijkste abstractie in de historie van computersystemen is de digitale abstractie, waarin alles gerepresenteerd wordt in termen van twee waarden, typisch weergegeven als 0 en 1, bits. De ICT-revolutie is feitelijk de digitale revolutie. De digitale abstractie heeft het mogelijk gemaakt om te abstraheren van de onderliggende fysische processen. Bits kunnen op velerlei manieren zeer betrouwbaar gerealiseerd worden. Dit vormt een basis om computersystemen te ontwerpen in termen van de informatie die verwerkt wordt en de transformaties die daarvoor nodig zijn. Andere algemeen geaccepteerde abstracties laag in de abstractiepiramide zijn de *standard cell*, de RTL (*register-transfer level*), CABA (*cycle-accurate-bit-accurate*) en ISA (*instruction set architecture*) abstractieniveaus. Voor sequentiële systemen is er ook redelijke overeenstemming over abstracties op de hogere niveaus, typisch gebaseerd op functionele abstractie zoals de lambda calculus, of operationele abstracties als automaten en de Turing-machine. Er is echter weinig eensgezindheid over de juiste abstracties op de hogere niveaus voor open, parallelle systemen die

interageren met elkaar, hun omgeving en gebruikers. Het overgrote deel van de systemen die vandaag de dag ontwikkeld worden, is van dat laatste type. We staan dus voor de uitdaging om deze abstracties te ontwikkelen met de bijbehorende analyse- en synthesetechnieken. Deze abstracties en technieken dienen ingebed te worden in een modelgedreven ontwerptraject. Kostenefficiëntie van modelgedreven implementaties is daarbij een noodzakelijke voorwaarde.

Concluderend kunnen we vaststellen dat de complexiteit van ingebedde systemen en hun ontwerptrajecten slechts beheersbaar zal worden als we erin slagen om de juiste structuren en abstracties te vinden, en deze op de juiste wijze te combineren in het ontwerptraject. De basis om dit tot een succes te maken is modelvorming.

Voorspelbare systemen

Het ultieme doel van modelvorming is dat we computersystemen kunnen ontwerpen die te vertrouwen zijn. Een noodzakelijke voorwaarde is voorspelbaarheid. Het zou een belangrijke stap voorwaarts zijn als we het gedrag en de prestaties van computersystemen voorafgaand aan hun realisatie zouden kunnen voorspellen. De neiging bestaat om betere voorspelbaarheid vooral te zoeken in betere modelvorming. Modelvorming is natuurlijk een cruciaal ingrediënt om te komen tot een betere voorspelbaarheid. Er is echter nog een belangrijke component, namelijk de systemen zelf. Computersystemen worden doorgaans ontworpen met kosten, energieverbruik en prestaties als belangrijkste criteria voor optimalisatie. Voor ingebedde systemen zouden we daar voorspelbaarheid aan toe moeten voegen. De wereld van ingenieurs is maakbaar. Modellen en systemen zouden hand in hand ontwikkeld moeten worden om te komen tot beter voorspelbare systemen. Neem als voorbeeld nogmaals de digitale abstractie. Sequentiële digitale systemen kunnen we goed beschrijven en voorspellen middels toestandsmodellen. Deze modellen vormen de basis voor vele analyse- en synthesegereedschappen voor digitale circuits. Uitgangspunt daarbij is dat een globale synchrone klok toestandsovergangen regisseert. Mede dankzij deze modelleringstechnieken is het synchroon geklokte digitale circuit uitgegroeid tot de belangrijkste bouwsteen van vrijwel elk computersysteem. Dit terwijl een synchrone klok ook grote nadelen heeft. Het belangrijkste daarvan is het hoge energieverbruik. Dit wordt echter voor lief genomen, en er zijn inmiddels allerlei oplossingen bedacht om in de context van synchroon geklokte systemen het energieverbruik te reduceren. Het artefact van de synchrone klok is bedacht om circuits beter voorspelbaar te maken. Als een systeem moeilijk te voorspellen is, dan kunnen we op zoek gaan naar betere modellen óf we passen het systeem aan. De kwaliteit van computersystemen is geen eenvoudige eendimensionale eenheid, maar een complexe eenheid met meerdere dimensies die vaak conflicteren. Voorspelbaarheid is een systeemeigenschap die in beperkte mate uitgewisseld kan worden tegen andere eigenschappen, zoals prestatie en energie-efficiëntie. Voorspelbaarheid verdient een prominente plaats in het ontwerp van ingebedde systemen, een prominentere plaats dan het vandaag de dag heeft.

Zichtbaar blijven

In het voorgaande heb ik een overzicht gegeven van de uitdagingen die ik zie om te komen tot computersystemen waarop we kunnen vertrouwen. Als we succesvol zijn, dan verdwijnt de computer. De computer wordt dan een integraal onderdeel van onze omgeving, van ons dagelijks leven, waar we ons niet langer bewust van zullen zijn, net zoals we ons niet bewust zijn van andere gebruiksvoorwerpen. Juist in die situatie zal het voor de ICT-gemeenschap belangrijk zijn om zichtbaar te blijven, om er voor te zorgen dat de maatschappij het belang van computers en van ICT in brede zin op waarde schat.

Het imago van ICT is slecht. Dat hebben we als ICT-gemeenschap in belangrijke mate aan onszelf te wijten. Ten eerste leveren we te vaak systemen af die onvoldoende betrouwbaar blijken. Ten tweede hebben we de neiging om in onze communicatie naar buiten toe, zowel in publieke media als bijvoorbeeld in de werving van onderzoeksgelden, de problemen te benadrukken en het bestaan van problemen aan te voeren als reden voor verdere investeringen. Deze combinatie leidt tot een afnemend publiek en politiek vertrouwen in de kwaliteit van ICT. Subsidie voor wetenschappelijk ICT-onderzoek, bijvoorbeeld, is de afgelopen vijf jaar meer dan gehalveerd [10].

Onderzoek en innovatie zijn cruciaal voor economische groei. Investeringen in onderzoek en innovatie blijken economisch zeer rendabel en hebben een positieve invloed op ons welzijn [11]. ICT-ontwikkelingen dragen daar sterk aan bij en zijn een onmisbare schakel in het geheel. Daarnaast is onderwijs cruciaal om kennis te behouden en op te bouwen, en om de maatschappij te voorzien van de benodigde specialisten. Het is dan ook belangrijk om als ICT-gemeenschap zichtbaar te blijven en wel in positieve zin. Niet alleen om maatschappelijk en politiek draagvlak te creëren, maar ook om ICT-gerelateerde opleidingen aantrekkelijker te maken. Om deze positieve zichtbaarheid te realiseren zullen we in de eerste plaats eenvoudigweg vaker betrouwbare systemen op moeten leveren. In de tweede plaats zullen we in onze communicatie de verworvenheden van ICT en de potentie van nieuwe ontwikkelingen veel sterker moeten benadrukken.

ICT heeft veel welvaart en welzijn gebracht. De uitvinding van de computer was de aanleiding voor de derde industriële revolutie [24]. De computer staat daarmee op gelijke hoogte met de stoommachine en elektriciteit. De potentie voor verdere ontwikkelingen is enorm. ICT-ontwikkelingen zullen cruciaal zijn voor de verdere ontwikkeling van communicatie en infotainment, professionele en industriële high-tech systemen, mobiliteit en transport, zorg, en infrastructuur. Daarmee zal ICT een belangrijke bijdrage leveren aan bijvoorbeeld de verduurzaming van onze samenleving en aan de oplossing van een maatschappelijk probleem als de vergrijzing. ICT is leuk, spannend en in vele opzichten bijzonder interessant. Nederland en de Nederlanders moeten de ambitie hebben om in een dergelijk belangrijk technologiegebied voorop te blijven lopen in de hele keten van onderwijs, onderzoek, innovatie en toepassing.

Tot besluit

Wetenschap is teamwerk. Dat geldt zeker voor ICT-onderzoek. Ingebedde systemen zijn letterlijk onderdeel van andere systemen, vaak uit andere toepassingsgebieden. De uitdagingen waar we voor staan kunnen alleen opgelost worden door samen te werken. Samenwerking tussen electrical engineering en informatica, de twee disciplines die samen het hart van de ICT vormen. De oprichting van EIRICT, het Eindhoven Institute for Research on ICT, is een goed begin; een bachelortrack Computer Engineering zou een mooie aanvulling zijn. Samenwerking ook met de aanpalende ingenieursdisciplines, met de pure wetenschappen en met de alpha- en gammawetenschappen. Al deze disciplines zijn nodig in de keten van idee tot toepassing. Ik kijk ernaar uit om samen tot betrouwbare toepassingen van ingebedde systemen te komen. De mogelijkheden zijn ongekend.

Wetenschap is teamwerk. Ook ik heb veel te danken aan de mensen in mijn omgeving. Mijn interesse voor wetenschap is ontstaan tijdens mijn afstudeerproject in 1992/1993 voor de toenmalige ingenieursopleiding Technische Informatica van de TU/e. Ik heb dat project gedaan aan de University of Waterloo, Canada, onder begeleiding van Jay Black. Dankzij de inspirerende omgeving die ik daar aantrof, heb ik voor de wetenschap gekozen. Ik ben Jay daar nog altijd dankbaar voor.

In mijn carrière tot dusverre zijn mijn direct leidinggevenden altijd een inspiratiebron geweest. Dieter Hammer, Jos Baeten, Kees van Hee, Wil van der Aalst, Jochen Jess, Ralph Otten, Ed Brinksma en Boudewijn Haverkort hebben mij het vak van wetenschapper geleerd. Een speciaal woord van dank gaat uit naar mijn promotoren Jos Baeten en Kees van Hee. Mijn promotieonderzoek vormt nog altijd een stevig fundament waar ik met plezier op verder bouw. Ook Ralph Otten, groepsleider van Electronic Systems en mijn leidinggevende gedurende de afgelopen tien jaar, wil ik bij deze bedanken. Ralph heeft me vertrouwen en ruimte gegeven. Ralph legt de lat hoog en doet dingen anders. Allemaal belangrijk in de wetenschap.

Door de jaren heen heb ik het genoegen gehad om met velen te mogen samenwerken. Veel van het daadwerkelijke onderzoek wordt gedaan door afstudeerders, promovendi en postdocs. De inhoudelijke discussies met hen, het zoeken naar antwoorden en oplossingen voor de vragen die we hebben en de voldoening die optreedt als alle puzzelstukjes blijken te passen, maken het werk leuk. Natuurlijk spelen ook alle andere collega's daarbij een belangrijke rol. Creativiteit gedijt bij goede sfeer. Ik heb altijd het geluk gehad te mogen werken in groepen waar de sfeer goed was. Ik wil iedereen bedanken die op enige wijze bijgedragen heeft aan mijn werk. Zonder anderen te kort te willen doen, wil ik Henk Corporaal, Marc Geilen, en Sander Stuijk hier in het bijzonder noemen. Het is een voorrecht om met hen te mogen werken.

Ik wil ook het College van Bestuur van de TU/e en het faculteitsbestuur van Electrical Engineering bedanken voor het instellen van de leerstoel en het in mij gestelde vertrouwen.

Ten slotte wil ik mijn naaste familie bedanken. Mijn ouders, voor alles, voor de basis die ik van hen meegekregen heb en voor het feit dat ze nog altijd iedere keer weer voor ons klaar staan. Isabelle, mijn vrouw, voor haar liefde, begrip en onvoorwaardelijke steun. Het is niet altijd eenvoudig om twee academische carrières en een gezin te combineren. Maar samen lukt het vrij aardig. Anne en Marc wil ik bedanken voor de vele leuke uren samen, die me eraan herinneren wat echt belangrijk is.

Ik wil besluiten met een citaat.

"*An understanding of the natural world and what's in it is a source of not only a great curiosity but great fulfillment*." David Attenborough

David Attenborough dacht bij deze uitspraak zeker niet aan computers. Computers zijn nu echter een natuurlijk onderdeel van onze wereld geworden, zoals Mark Weiser twee decennia geleden al voorzag. Deze uitspraak verwoordt daarmee heel goed wat voor mij onderzoek in het algemeen, en computermodellering in het bijzonder, zo mooi maakt. Nieuwsgierigheid, begrip, voldoening.

Ik heb gezegd.

Referenties

1. Artemis Industry Association. Artemis Strategic Research Agenda. 2006.
2. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Transactions on Dependable and Secure Computing, 1(1): 11-33, 2004.
3. BCC Inc. Embedded Systems: Technologies and Markets. Report IFT016C, april 2009. http://www.bccresearch.com/report/IFT016C.html (geraadpleegd op 5 februari 2011).
4. Eurostat. Real GDP Growth Rate. http://epp.eurostat.ec.europa.eu/ (geraadpleegd op 5 februari 2011).
5. FAST Gmbh. Study of Worldwide Trends and R&D Programmes in Embedded Systems in View of Maximizing the Impact of a Technology Platform in the Area, november 2005.
6. Gartner. 1 Billion PCs In Use Worldwide, juni 2008. http://www.gartner.com/it/page.jsp?id=703807 (geraadpleegd op 5 februari 2011).
7. M.C.W. Geilen, T. Basten, B.D. Theelen, R.H.J.M. Otten. An Algebra of Pareto Points. Fundamenta Informaticae. 78(1):35-74, 2007.
8. A.H. Ghamarian et al. Throughput Analysis of Synchronous Data Flow Graphs. In: Application of Concurrency to System Design, 6th Int. Conference, ACSD 2006, Proc., pagina's 25-34. IEEE CS Press, Los Alamitos, CA, VS, 2006.
9. R. Hoes et al. Quality-of-Service Trade-off Analysis for Wireless Sensor Networks. Performance Evaluation. 66(3-5):191-208, 2009.
10. Innovatieplatform, werkgroep KIA. Bouw op talent! Jaarlijkse evaluatie Kennisinvesteringsagenda (KIA) 2006-2016, maart 2010.
11. Innovatieplatform. Kennis en Innovatie Agenda 2011-2020, juli 2010.
12. G. Kahn. The Semantics of a Simple Language for Parallel Programming. In: Information Processing 74: IFIP Congress 74, Proc., pagina's 471-475. North-Holland, Amsterdam, 1974.
13. E.A. Lee, D.G. Messerschmitt. Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing. IEEE Transactions on Computers, 36(1): 24–35, 1987.
14. Moving Pictures Expert Group. http://mpeg.chiariglione.org/ (geraadpleegd op 5 februari 2011).

15. Roessingh Research and Development. ALWEN Experiment Protocol: ADL Recognition in the Home Environment. 2010.
16. J.W.J.M. Rutten. Persoonlijke communicatie.
17. H. Shojaei et al. A Parameterized Compositional Multi-dimensional Multiple-choice Knapsack Heuristic for CMP Run-time Management. In: 46th Design Automation Conference, DAC 2009, Proc., pagina's 917-922. ACM, New York, NY, VS, 2009.
18. S. Stuijk, T. Basten, M.C.W. Geilen, H. Corporaal. Multiprocessor Resource Allocation for Throughput-Constrained Synchronous Dataflow Graphs. In: 44th Design Automation Conference, DAC 2007, Proc., pagina's 777-782. ACM, New York, NY, VS, 2007.
19. S. Stuijk, M.C.W. Geilen, T. Basten. A Predictable Multiprocessor Design Flow for Streaming Applications with Dynamic Behaviour. In: Digital System Design, 13th EUROMICRO Conference, DSD 2010, Proc., pagina's 548-555. IEEE CS Press, Los Alamitos, CA, VS, 2010.
20. B.D. Theelen et al. A Scenario-Aware Data Flow Model for Combined Long-Run Average and Worst-Case Performance Analysis. In: Formal Methods and Models for CoDesign, 4th ACM & IEEE conference, MEMOCODE 2006, Proc., pagina's 185-194. IEEE CS Press, Los Alamitos, CA, VS, 2006.
21. Volkskrant. Toyota Verkoopt Meer Auto's, Ondanks Terugroepacties. 27 juli 2010. http://www.volkskrant.nl/vk/nl/2680/Economie/article/detail/1012697/2010/07/27/Toyota-verkoopt-meer-auto-s-ondanks-terugroepacties.dhtml (geraadpleegd op 5 februari 2011).
22. M. Weiser. The Computer for the 21st Century. Scientific American, 165(3):94–104, 1991, herdrukt in IEEE Pervasive Computing, januari-februari 2002, 19-25.
23. Wikipedia. Abstractie. http://nl.wikipedia.org/wiki/Abstractie (geraadpleegd op 5 februari 2011).
24. Wikipedia. Industriële revolutie. http://nl.wikipedia.org/wiki/Industri%C3%ABle_revolutie (geraadpleegd op 5 februari 2011).
25. Wikipedia. Ingebed systeem. http://nl.wikipedia.org/wiki/Ingebed_systeem (geraadpleegd op 5 februari 2011).
26. Wikipedia. Model (wetenschap). http://nl.wikipedia.org/wiki/Model_(wetenschap) (geraadpleegd op 5 februari 2011).

Curriculum vitae

Prof.dr.ir. Twan Basten is per 1 mei 2009 benoemd tot voltijdhoogleraar Computational Models for Networked Embedded Systems aan de faculteit Electrical Engineering van de Technische Universiteit Eindhoven (TU/e).

Twan Basten (1969) studeerde in 1993 cum laude af in de Technische Informatica aan de TU/e. In 1998 promoveerde hij op een proefschrift over de combinatie van twee wiskundige modelleringstechnieken voor de beschrijving van gedistribueerde systemen. De zoektocht naar synergie tussen modelleringstechnieken en naar praktische toepassing van modellen is sindsdien een rode draad in zijn werk.
Na zijn promotie begon Twan Basten als universitair docent aan de TU/e, eerst bij de faculteit Wiskunde en Informatica en sinds 1999 bij Elektrotechniek, nu Electrical Engineering. In 2004 is hij benoemd tot universitair hoofddocent en in 2009 tot hoogleraar. Sinds 2008 is Twan Basten tevens research fellow van het Embedded Systems Institute in Eindhoven. Eerder werkte hij als gastonderzoeker bij Philips Research Eindhoven en de Carnegie Mellon University in de Verenigde Staten. Zijn onderzoek betreft modelgedreven ontwerp van ingebedde systemen. Modelleringstechnieken met bijbehorende analyse- en synthesetechnieken hebben zijn bijzondere interesse.

Colofon

Productie
Communicatie Expertise
Centrum TU/e

Fotografie cover
Rob Stork, Eindhoven

Ontwerp
Grefo Prepress,
Sint-Oedenrode

Druk
Drukkerij Snep, Eindhoven

ISBN 978-90-386-2478-5
NUR 958

Digitale versie:
www.tue.nl/bib/

Bezoekadres
Den Dolech 2
5612 AZ Eindhoven

Postadres
Postbus 513
5600 MB Eindhoven

Tel. (040) 247 91 11
www.tue.nl

Technische Universiteit
Eindhoven
University of Technology