Architectuur-uitdagingen in embedded systemen

Hard- en software botsen vaak

In allerlei apparaten zit embedded software. De uitdaging is dat de virtuele wereld van de software moet worden gecombineerd met de fysieke wereld van het apparaat. Om met deze complexiteit om te gaan wordt systeemarchitectuur toegepast. De auteur illustreert dit aan de hand van het voorbeeld van de MRI-scanner.

Gerrit Muller

Bijna alle apparaten die we in het dagelijks leven gebruiken, bevatten een of meer processors waarop software draait. Voorbeelden zijn gsm's, tv's, printers, elektronenmicroscopen en MRI-scanners. De gebruiker is zich vaak niet bewust van de software in het apparaat. De processors en de software zijn dusdanig *embedded* dat ze een integraal onderdeel van het systeem zijn. Voor veel apparaten is de embedded software onmisbaar, want deze bepaalt vaak de beschikbare functionaliteit, performance en vele andere systeemkwaliteiten.

De wereld van de software is een virtuele wereld. Met bits en bytes wordt een complexe en abstracte structuur geschapen die de gewenste functionaliteit moet opleveren. De apparaten aan de andere kant zijn vaak fysieke systemen, die onderworpen zijn aan de wetten van de fysica. Deze fysieke wereld is complex door verstoringen, ruis, interferentie en vele andere fysische verschijnselen. Deze twee totaal verschillende vormen van complexiteit ontmoeten elkaar in embedded systemen. Systeemarchitectuur is een middel om om te gaan met deze complexiteiten en maakt het mogelijk met beperkte inspanning en in een beperkte tijd te komen tot goed werkende implementaties.

De MRI-scanner

Figuur 1 toont een MRI-scanner als voorbeeld van een embedded systeem. Deze systemen worden nu ruim 25 jaar gebruikt in ziekenhuizen. In deze systemen worden tientallen verschillende technologieën toegepast, waarvan de meest in het oog springende zijn: supergeleidende magneten, spoelen en versterkers om magnetische gradiëntvelden op te wekken, radiofrequente (RF) spoelen voor het zenden en ontvangen van radiovelden, mechanische constructies om het patiëntenbed te verplaatsen, analoge en digitale elektronica voor het aansturen en het ontvangen van meetdata, en allerlei verschillende processors met grote hoeveelheden software.

Philips biedt een breed spectrum van MRI-scanners met een rijkdom aan opties. Deze MRI-systemen zijn systemen met open magneten en conventionele cilindrische magneten van verschillende veldsterktes. De opties zijn onder andere gradiëntspoelen, RF spoelen, klinische accessoires en klinische applicatiepakketten. De systemen uit deze productfamilie met hun opties zijn afgeleid van een en hetzelfde platform.

De eerste generatie systemen gebruikte een standaardcomputer- en besturingssysteem: een VAX11/750 met VMS. Voor de medische markt

Samenvatting

Kenmerkend voor embedded systemen is de gedwongen samenwerking tussen softwareontwerpers en ontwerpers uit de hardware en fysica. Hierdoor ontstaat een extra dimensie van complexiteit, boven op de uitdagingen van marktdynamiek, interoperabiliteit van systemen, betrouwbaarheid, energiegebruik, beveiliging en creativiteit. Architectuur kan helpen de verschillen tussen disciplines te overbruggen.

Figuur 1. **Achieva MRI-scanner** *(foto: Philips Medical Systems)*

was dat uniek, want proprietary computers en besturingssystemen waren gebruikelijk. In de afgelopen 25 jaar is de software geëvolueerd van C-code via C++ en Java naar C# draaiend op Windows XP op standaard pc-hardware. Op dit moment werken er enkele honderden mensen aan de software van dit systeem, op verschillende locaties in de wereld (waaronder de Verenigde Staten, Nederland en India). De totale omvang van de software heeft de tien miljoen regels code overschreden.

Een MRI-scanner is een uiterst gevoelig meetinstrument dat uit heel zwakke radiosignalen uit het menselijk lichaam beelden en fysisch-chemische informatie opbouwt. Deze radiosignalen worden opgewekt door een uiterst nauwkeurige generatie van magneet- en radiovelden. Hiertoe moeten alle onderdelen, gerealiseerd in de genoemde verschillende technologieën, tegelijkertijd samenwerken. Al die onderdelen worden door honderden ontwerpers ontworpen en bouwen vaak voort op bestaande realisaties van duizenden manjaren werk.

Binnen de software zelf is ook sprake van heel verschillende klassen van problemen met de bijbehorende paradigma's, denk aan: *hard* (gegarandeerde) *real-time* controle en acquisitie; fysisch-elektronisch afregelen; zwaar algoritmisch rekenwerk; informatiemodellering en databases; beeldbewerking; grafische gebruikersinterface; klinische applicaties en workflowapplicaties; distributie van functionaliteit via client/server; besturingssystemen, abstractielagen en interne systeemboekhouding; systeemtesten en diagnose. De subsystemen en componenten van deze subsystemen hebben vaak met meerdere van deze klassen van problemen tegelijkertijd te maken.

Een deel van het systeem verzorgt de acquisitie en de reconstructie van de beelden: het instrumentdeel. Dit instrumentdeel is hard real-time en fysisch-elektronisch van aard. De software van het instrumentdeel is verdeeld over een hard real-time kernel en een algemeen zwaarder besturingssysteem. Communicatie met de rest van het systeem gebeurt in termen van gestandaardiseerde informatiemodellen. De informatiemodellen, gebaseerd op internationale standaarden zoals DICOM, beschrijven de patiëntgerelateerde informatie, de klinisch relevante informatie, de informatie over de beeldvorming en die over de workflow. Bovendien is er een achterliggend model over hoe met deze gegevens wordt omgegaan.

Voor dit soort systemen moet veel energie gestoken worden in het bereiken van gewenste systeemeigenschappen, zoals responstijden, *patient throughput*, beeldkwaliteit, veiligheid en gebruikersgemak. De architectuur is het middel om deze doelstellingen te realiseren en deze moet hiertoe het systeem vanuit tientallen gezichtspunten belichten. Architectuur in deze systemen gaat dan ook veel verder dan het opdelen van een systeem in subsystemen en componenten en het bepalen van de bijbehorende interfaces.

Een centraal element van de softwarearchitectuur is de ontkoppeling van de hoofdfuncties in de

MRI-scanners door middel van een lokale database. Het instrumentdeel van het systeem gebruikt een aantal transformatiestappen om van klinisch relevante parameters die door de operator worden ingesteld, te komen tot hardwareaansturing die hard real-time is: het zogenaamde initialisatie-conversie-executiepatroon. De daadwerkelijke executie gebruikt het, door de hardware ondersteunde, 'stretchpatroon' om het hard real-time gedrag te garanderen. Het stretchpatroon ontkoppelt de softwaregeneratie van uit te voeren instellingen en de hardware-uitvoering hiervan. Een stretch is een kleine tijdsperiode in de orde van milliseconden die de software voorligt op de hardware.

Trends

Bij MRI-scanners spelen veel trends die ook in andere embedded systemen te herkennen zijn: marktdynamiek, interoperabiliteit van systemen, betrouwbaarheid, energiegebruik, beveiliging en creativiteit.

Marktdynamiek

De gezondheidszorg is sterk in beweging door vergrijzing, kostendruk en veranderende rollen van bijvoorbeeld verzekeringsmaatschappijen, resulterend in marktdynamiek. De dynamiek wordt ook gedeeltelijk veroorzaakt door de globalisering. De concurrentie is wereldwijd, waardoor sociale, politieke en culturele factoren meespelen. De factor 'mens' is sterk bepalend in de markt. Dit kan leiden tot hype of modegevoeligheid. Draagbare telefoons zijn hiervan een extreem voorbeeld. Maar zelfs voor grote professionele systemen zoals MRI-scanners is het uiterlijk een verkoopfactor; het uiterlijk moet bij de tijd zijn en daarmee zijn dus zelfs dit soort grote professionele systemen modegevoelig.
Door de snelle technologische ontwikkelingen is de verwachting van klanten ook veranderd. Productvernieuwingscycli van jaren moeten voor veel functies terug naar maanden.

Interoperabiliteit van systemen

Interoperabiliteit van systemen is als klantwens vanzelfsprekend. In deze wereld spelen vele marktpartijen een rol, waarbij de markt van beeldvormende apparatuur wordt gedomineerd door drie grote bedrijven: GE, Siemens en Philips. Bedrijven gebruiken soms de architectuur als competitief wapen, bijvoorbeeld om de toekomstgerichtheid te garanderen, de uitbreidbaarheid mogelijk te maken of de zorgprocessen te stroomlijnen door het systeem te integreren in de omgeving (interoperabiliteit). Bijna alle functies op klantniveau vereisen de samenwerking van meerdere systemen. Zo moeten systemen van verschillende producenten met elkaar kunnen samenwerken, en datzelfde geldt voor applicaties uit geheel verschillende domeinen; zelfs systemen uit het verleden moeten kunnen samenwerken met systemen die nog niet eens bestaan.
Tijdens de interactie van meerdere systemen ontstaat er soms onverwacht nieuw gedrag: *emerging behavior*. Denk hierbij bijvoorbeeld in algemenere zin aan een beurskrach die mede wordt veroorzaakt door allerlei applicaties die reageren op beurssignalen en elkaar daardoor versterken.

Betrouwbaarheid

Betrouwbaarheid van apparaten is in toenemende mate een uitdaging. De betrouwbaarheid van continu veranderende apparaten, zoals een MRI-scanner, met zoveel verschillende technologieën en met miljoenen regels code is in het geheel niet vanzelfsprekend. Door de toename van de hoeveelheid software en de complexiteit blijven er meer verborgen fouten over na het testen. Als dan meerdere complexe systemen samen een functie moeten uitvoeren, gaan vele van deze verborgen fouten zich uiten.

Energiegebruik

Energiegebruik speelt om meerdere redenen een belangrijke rol. Om te beginnen begrenst het energiegebruik de afbeeldingmogelijkheden van de MRI-scanner. Zo wordt de snelheid van imaging begrensd door de hoeveelheid energie die nodig is om de magnetische gradiëntvelden op te wekken. Als we iets breder kijken naar embedded systemen, zien we dat energiegebruik een veelvoorkomende aansturende factor in het ontwerp is. Voor draagbare systemen bijvoorbeeld is de tijd dat het systeem gebruikt kan worden zonder bijladen belangrijk. Het systeem moet bovendien zo licht mogelijk blijven, dus moeten energiedragers zoals accu's zo licht en klein mogelijk blijven. Een andere drijfveer voor laag energiegebruik is de productie van warmte en lawaai: minder energiegebruik betekent minder koelingsproblemen en minder geluid. Op grotere schaal zijn de kosten

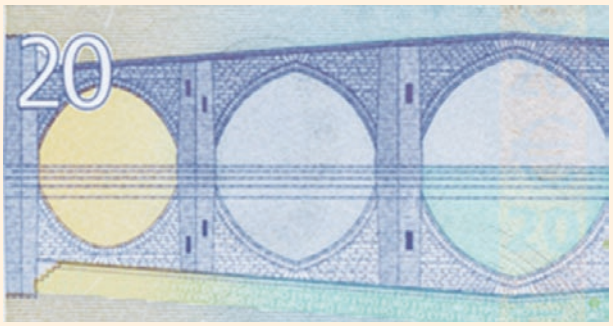
van energie significant en soms is zelfs de beschikbaarheid van voldoende energie een probleem.

Beveiliging
Beveiliging is in deze wereld vol bedreigingen (zoals virussen, Trojaanse paarden en spyware gemaakt door terroristen, illegalen, piraten of dieven) een noodzakelijk kwaad. De beveiliging van medische systemen tegen ongewenste indringers is actueel. In deze markt zijn er standaarden zoals HIPAA die voorschriften geven. Maar het gebruik van standaardcomputertechnologie opent ook een doos van Pandora, die met middelen zoals virusscanners en firewalls moet worden bestreden. Al deze middelen interfereren met de hoofdfuncties van het systeem. Maar al te vaak staan de beveiligingsmaatregelen op gespannen voet met de hoofdfunctie van het apparaat. Zo kan het scannen van virussen een directe negatieve impact op de responstijden hebben. Architectuur helpt betrouwbaarheid en beveiliging te bereiken, bijvoorbeeld door richtlijnen voor het indelen in meer geïsoleerde compartimenten.

Creativiteit
Creativiteit is in deze wereld met bijna onbegrensde technologische mogelijkheden een schaarse eigenschap. De MRI-techniek is nog zo jong en veelzijdig dat er nog heel veel ruimte is voor nieuwe of verbeterde applicaties. Creativiteit is daartoe een beperkende factor, hoewel in de praktijk de complexiteit van het apparaat een eerste rem op vernieuwing oplevert. Architectuur moet helpen de complexiteit te beteugelen, zodat creativiteit de enige beperking is voor het komen tot vernieuwing. Op dit moment loopt er een onderzoeksproject met de vraag hoe de huidige architectuur verbeterd kan worden om meer mogelijkheden voor creativiteit te scheppen (Embedded Systems Institute, z.j.).
Hoe kunnen ontwerpers erachter komen wat de latente behoefte van een groep klanten is? Hoe kan een ecosysteem ontstaan van nieuwe toepassingen die niet van tevoren zijn voorzien? Waar kan waarde gecreëerd worden die vervolgens leidt tot een nieuwe bron van inkomsten?

Uitdagingen

De hiervoor genoemde trends zijn typerend voor veel meer embedded systemen, van mobiele telefoons en televisies tot aan waferscanners. Technologisch gezien lijkt de wereld van de embedded systemen bepaald te worden door de wet van Moore: meer rekenvermogen, meer geheugencapa-



citeit, meer communicatiebandbreedte en betere sensoren, actuatoren en beeldschermen. Combineer dit met de juiste software en de mogelijkheden zijn bijna onbegrensd. Ondanks de razendsnelle technologische ontwikkelingen staan de ontwerpers van embedded systemen voor diverse uitdagingen: marktdynamiek, interoperabiliteit van systemen, betrouwbaarheid, energiegebruik, beveiliging en creativiteit.
Al deze trends, misschien met uitzondering van energiegebruik, zijn overigens niet uniek voor embedded systemen. De IT-industrie heeft eigenlijk ook met al deze trends te maken. Wat het domein van embedded systemen onderscheidt van andere sectoren, zijn de beperkingen en randvoorwaarden die volgen uit het fysieke apparaat in de fysische omgeving. De virtuele softwarewereld en de fysieke hardwarewereld ontmoeten elkaar in de embedded systemen.
Ontwerpers met totaal verschillende achtergronden moeten met elkaar samenwerken: software- en hardwareontwerpers, een hele uitdaging. Softwareontwerpers zijn opgegroeid in de 'maakbare' wereld: aangenomen wordt dat dat wat gespecificeerd wordt, gebouwd en getest kan worden en dat het daarna werkt het. De softwarewereld is abstract en virtueel en daardoor ongrijpbaar en ontoegankelijk voor niet-softwaremensen. Hardwareontwerpers zijn onderworpen aan de grillen van de natuur, die inherent vol zit met onzekerheid, onbekendheid en verstoringen. De hardwarewereld is vaak letterlijk tastbaar en veel fysische verschijnselen zijn voorstelbaar. Begrip van de hardwarewereld vereist acceptatie van en het kunnen omgaan met de onzekerheden die voortkomen uit de fysische natuur, want niet alles wat gespecificeerd wordt, kan daadwerkelijk gebouwd worden Vooral voor mensen die geschoold zijn in software, blijkt het vaak moeilijk om om te gaan met onzekerheden en verstoringen.
Een voorbeeld van het verschil in perspectief tussen hardware- en softwaremensen werd al vroeg in de MRI-ontwikkeling zichtbaar. De RF spoelen

in de MRI-scanner moeten getuned en gematcht worden met behulp van twee verstelbare condensatoren. Deze condensatoren werden versteld door een mechanische draaibeweging. Het eerste ontwerp gebruikte analoge motoren met een positiemeting via een encoder op de motoras. De softwaremensen moesten dus via een feedbackloopje de condensator verdraaien, een activiteit die niet volledig deterministisch is. De softwaremensen vroegen daarom naar een stappenmotorimplementatie, want dat is vanuit softwarestandpunt een *feed forward* en dus een deterministisch proces. Wat de softwaremensen zich echter niet realiseerden, was dat de hele keten veel meer onzekerheden bevat. Zo is er bijvoorbeeld zoveel mechanische overbrenging (onder andere een lange as) dat er hysterese optreedt: een verdraaiing aan het begin van de as wordt pas aan het eind voelbaar bij voldoende verdraaiing. De positie van de condensator zelf is niet goed voorspelbaar in de fysieke wereld. Het gevolg is dat de aansturingsalgoritmes robuust moeten zijn voor al deze vormen van onzekerheid. Het gebruik van stappenmotoren creëerde een onjuist gevoel van voorspelbaarheid.

Rol van architectuur

In de wereld van embedded systemen helpt architectuur bij:

- Het geven van overzicht aan alle projectleden: waar past mijn bijdrage in het systeem, hoe beïnvloedt mijn bijdrage de rest van het systeem?
- Ontwerp en implementatie door het geven van richtlijnen, bijvoorbeeld voor decompositie, interfacing, allocatie en het gebruik van resources.
- Het communiceren van de gronden voor specificatie en ontwerpkeuzes.
- Het realiseren van systeemeigenschappen die vaak het gevolg zijn van de som van alle gedetailleerde ontwerpkeuzes.
- Het ondersteunen van de systemen gedurende de volledige levensduur, die voor embedded systemen enkele jaren tot tientallen jaren kan beslaan.
- Het ondersteunen van een productfamilieaanpak gebaseerd op een gemeenschappelijk platform.

State-of-the-practice in embedded-systeemarchitectuur

In de praktijk is er een grote variatie in het niveau van de architectuurontwikkeling bij de verschillende bedrijven en zelfs afdelingen binnen een bedrijf. Succesvolle bedrijven hebben multidisciplinaire ontwikkelteams, waarbij een beperkt aantal mensen een systeemrol vervult. Zelfs de naamgeving van de systeemrol is heel divers: systeemarchitect, ontwerper, manager, engineer of coach. Productfamilies worden veelvuldig toegepast, waarbij meerdere verschillende producten van een gemeenschappelijke basis worden afgeleid. Desondanks worstelen bijna alle bedrijven met het bijbehorende configuratiemanagement en de balans tussen innovatie en beheersbaarheid. Theoretisch is architectuur de sleutel om beide na te streven; de hedendaagse praktijk is dat er nog veel geworsteld wordt. Veel bedrijven zijn in dit kader op zoek naar de heilige graal van onafhankelijke ontwikkeling en levering van componenten. De standaardelementen in deze zoektocht zijn frameworks, bijvoorbeeld .NET, service-oriented architectures, interfacespecificatie en beheer en modelgebaseerd ontwerpen. Hoewel dit veelbelovende technologieën zijn, is er nog geen zichtbare verbetering te zien in het omgaan met productdiversiteit en de steeds korter wordende ontwikkeldoorlooptijd.

Best practice in embedded-systeemarchitectuur

In enkele bedrijven wordt gewerkt met een referentiearchitectuur die de systeemarchitectuur van een enkel product in het bredere perspectief plaatst van markt en klantenwereld en die ontwerp-

richtlijnen geeft voor individuele systemen. Een referentiearchitectuur vangt de opgebouwde kennis en ervaring van werk in het verleden om doeltreffender te worden in de toekomst. De grootste uitdaging hierbij is om deze referentiearchitectuur concreet en praktisch toepasbaar te houden.

Ervaringen met architectuur in embedded systemen

Embedded systemen zijn van nature sterk multidisciplinair. Opleidingen echter zijn monodisciplinair. De natuurlijke tendens in veel bedrijven is om medewerkers ofwel technisch te laten groeien tot specialist, dan wel om hen als manager carrière te laten maken. Systeemarchitecten voor dit soort systemen hebben een brede technische bagage nodig, een brede set van sociale vaardigheden en een set van multidisciplinaire ontwerpmethoden. Systeemarchitecten met een dergelijk profiel blijken uiterst schaars te zijn.

Er lijkt bijna overal een gebrekkige aansluiting te zijn tussen de systeem- en hardwarewereld en de softwarewereld. Ook de aansluiting tussen de toepassingswereld en de technische wereld is moeizaam. In een flink aantal bedrijven zijn er twee disjuncte groepen actief op architectuurgebied: softwarearchitecten en (hardware- of applicatie)systeemarchitecten. De systeemarchitecten hebben er dan veel moeite mee om de software te begrijpen. Dit personele en organisatorische gat uit zich vaak door problemen gedurende de systeemintegratie, waarbij de verschillende onderdelen wel met elkaar moeten gaan functioneren.

Conclusie

Kenmerkend voor embedded systemen is de gedwongen samenwerking tussen softwareontwerpers en ontwerpers uit de hardware en fysica. Het gevolg is dat embedded-systeemontwerp een extra dimensie van complexiteit kent, boven op de uitdagingen die in vele andere sectoren ook aanwezig zijn, zoals marktdynamiek, interoperabiliteit van systemen, betrouwbaarheid, energiegebruik, beveiliging en creativiteit.

Architectuur maakt het omgaan met complexiteit eenvoudiger doordat het overzicht biedt, richtlijnen voor ontwerp en implementatie geeft, de beweegredenen communiceert, focust op de te realiseren systeemeigenschappen en focust op de gehele levensduur van de systemen en de volle breedte van de productfamilie.

De status-quo van architectuur in embedded systemen is dat er nog een gat gaapt tussen softwarearchitecten en (op hardware en fysica georienteerde) systeemarchitecten. Dit gat uit zich in problemen die opduiken tijdens de systeemintegratie.

Architectuur kan helpen de verschillen tussen disciplines te overbruggen. Hierbij is architectuur geen uniek alleenrecht van de architect. Architectuur werkt het best als alle spelers bereid zijn verder te kijken dan hun eigen discipline.

»Gebrekkige aansluiting tussen de systeem- en hardwarewereld en de softwarewereld«

Literatuur

Embedded Systems Institute (2006). ESI Research Agenda on Embedded Systems Engineering, www.esi.nl/site/institute/ESI_Research_Agenda_V7_11.pdf.

Embedded Systems Institute (z.j.). Darwin: Research Topic: System Evolvability, www.esi.nl/darwin.

Philips (2004). Nieuwe generatie Intera Achieva: omslag in MR-denken. *Medisch perspectief* 5, 03/2004, www.medical.philips.com/nl/assets/docs/news/publications/medical_perspective_5.pdf.

Gerrit Muller
is senior research fellow bij het Embedded Systems Institute en auteur van de Gaudi-website over systeemarchitectuur, www.gaudisite.nl.
E-mail: gerrit.muller@embeddedsystems.nl.