Een Raamwerk Voor Architectuur als Systematische Samenhang

Roel Wieringa*

23 oktober 2000

Samenvatting

Dit artikel geeft een raamwerk voor architectuur van organisaties en IT systemen bestaande uit drie productdimensies en twee procesdimensies. De productdimensies zijn *decompositie, verfijning* en *aspect* en de procesdimensies zijn *ontwerp* en *implementatie.* Het raamwerk wordt vervolgens toegepast op het alignmentconcept en op een referentiemodel voor een multi-channel mid-office architectuur. Tenslotte wordt het hier gepresenteerde raamwerk vergeleken met enkele andere raamwerken en met de probleemstructureringsbenadering van Jackson.

1 Inleiding

Architectuur is een onderwerp dat de status van hype dreigt te krijgen: Snel in de belangstelling gekomen, breed gedragen, ongedefinieerd, en snel uitgeblust. In een korte periode werd in rap tempo een aantal boeken gepubliceerd die, in steeds andere formaten, vrijwel dezelfde kennis bleken te herhalen [4, 6, 18]. Dat wijst op een gebrek aan vooruitgang. Toch zou het jammer zijn als de interesse voor software-architectuur zou afnemen. Het onderwerp is mijn inziens centraal in de overgang naar een netwerkmaatschappij, waarin systemen van ICT systemen een infrastructuur bieden om op flexibele wijze geografisch gedistribueerde samenwerkingsverbanden aan te gaan. De flexibiliteit in partnerkeuze die hierbij gewenst is, kan alleen onstaan als er op alle niveau's van het netwerk een stabiele architectuur is. Architectuur staan aan het hart van de technische infrastructuur van de netwerkmaatschappij.

In dit artikel geef ik een definitie van het begrip van architectuur die sterk verwant is aan het klassieke systeembegrip en werk dit vervolgens uit langs drie productgerichte en twee procesgerichte dimensies. Na een presentatie van het raamwerk analyseer ik de rol van alignement in dit raamwerk en pas ik het toe op een mid-office architectuur [14]. Vervolgens vergelijk ik het door mij geïntroduceerde architectuurconcept met enkele andere definities. Tenslotte bespreek ik een mogelijke rol van het hier gepresenteerde raamwerk in het klassificeren van de probleem-structureringstechnieken van Jackson.

*Faculteit Informatica, Universiteit Twente, Postbus 217, 7500 AE Enschede. email: roelw@cs.utwente.nl. http://www.cs.utwente.nl/∼roelw.

2 Definitie van Architectuur

De architect van een gebouw is de persoon die een ontwerp van dat gebouw heeft gemaakt, d.w.z. een schets van de elementen van het gebouw die gebruikt kan worden door de bouwer. De schets laat niet alleen de elementen zien, maar ook hun samenhang. Het laat zien hoe de elementen samenhangen en door die samenhang de eigenschappen van het gebouw als geheel creëeren. Die eigenschappen zijn bijvoorbeeld het aantal verdiepingen en kamers, het gemak waarmee men zich door het gebouw beweegt, de mogelijke functies van de kamers, de locatie van de in- en uitgangen ten opzichte van de geografie van de omgeving, en de "uitstraling" van het gebouw in zijn omgeving.[1]

Deze beschrijving van de taken van een architect wijst op enkele belangrijke eigenschappen van de taak van de architect:

- De architect bouwt niet, maar hij specificeert.
- De architect laat in zijn[2] specificatie zien hoe uit een samenvoeging van elementen een geheel met bepaalde eigenschappen ontstaat.

Anders gezegd, de architect *ontwerpt een systeem*. Dit behoeft voor informatici enige toelichting, omdat in de informatica de termen "ontwerp" en "systeem" zeer beperkte betekenissen hebben.

- *Ontwerpen* is het kiezen uit een aantal mogelijke handelingsalternatieven [2, 3, 7, 16].[3] Het plannen van een vakantie is een ontwerpaktiviteit: Waar gaan we heen, hoe gaan we daar heen, waar logeren we, etc. Het resultaat van deze ontwerpactiviteit is een vakantieplan. Het uitvoeren van dit plan is geen ontwerpactiviteit; het is het *uitvoeren* van het ontwerp. Ontwerpen resulteert in een plan voor actie, meestal specificatie genoemt. Essentieel voor ontwerpen is dat de ontwerper verschillende alternatieven evalueert op basis van de voorspelde eigenschappen van het gespecificeerde systeem. Als hij dat niet doet, dan is er geen sprake van ontwerp maar van evolutie: doe iets, kijk wat er gebeurd is, evalueer dat en doe dan weer iets. Bij ontwerpen vindt er daarentegen een evaluatie plaats *voordat* de handeling gedaan wordt.

  Zo bezien is het opstellen van een definitie van eisen een ontwerpactiviteit: De opsteller van de eisen kiest uit de verzameling van alle mogelijke eisen aan een systeem, en kiest die eisen die het probleem van de klant het beste oplossen. Ook het bedenken van een organisatie-strategie is een ontwerpactiviteit. Essentieel is in deze gevallen is dat verschillende handelingsalternatieven overwogen worden op basis van voorspellingen over de beschreven ontwerpen.

- Een *systeem* is een samenhangend geheel van elementen [5, 20]. Het begrip ontstond in de vijftiger jaren van de vorige eeuw en gaat gepaard met een aantal begrippen zoals subsysteem, aspectsysteem, input, output etc. die in zeer verscheidene situaties toegepast kunnen worden. Belangrijk is dat het begrip niet beperkt is tot software of hardware. Voorbeelden van systemen zijn organisaties, mensen, wetten en rivieren. Wat een systeem onderscheidt van een willekeurige hoop van elementen is dat er samenhang tussen de elementen is, zodanig dat door deze samenhang systeemeigenschappen ontstaan. Men spreekt ook wel van *emergente* eigenschappen: De systeemeigenschappen zijn niet vanuit de eigenschappen van de elementen alleen te verklaren, maar moeten vanuit de eigenschappen van de elementen *en* hun interactie verklaard

> worden. De kreet is zo versleten dat ik hem bijna niet durf op te schrijven, maar bij gebrek aan beter doe ik het toch: Bij een systeem is het geheel meer dan de som der delen.

Een architect ontwerpt systemen zodat op basis van het ontwerp in te zien is dat het systeem als geheel bepaalde, door de opdrachtgever gewenste, eigenschappen heeft. Dit vergt het vermogen van de architect om op basis van het ontwerp —de specificatie— de systeemeigenschappen te voorspellen. Voor een welbegrepen fysisch systeem zoals een auto of een gebouw kunnen systeemeigenschappen met een grote mate van zekerheid voorspeld worden door middel van bijvoorbeeld kwantitatieve berekeningen, simulaties of prototypes. Voor moeilijk te doorgronden conceptuele systemen zoals organisaties of een markt is dit moeilijker, maar niet onmogelijk. Een belangrijk deel van de mentale energie van topmanagers wordt besteed aan het doorgronden van maatschappelijke ontwikkelingen, het inschatten van de richting waarin de bedrijfsomgeving zich beweegt, en aan de effecten die herstructureringsmaatregelen van het bedrijf zullen hebben. Processen op dit niveau van aggregatie zijn contextgevoelig en niet volledig kwantitatief te specificeren, maar desalnietemin zal de manager voorspellingen doen voor verschillende handelingsalternatieven en zijn keuze op basis van die voorspellingen bepalen

In alle gevallen kunnen we zeggen dat de architectuur van het ontwerp de samenhang van het ontwerp is waarvan verwacht wordt dat daaruit de beoogde systeemeigenschappen zullen ontstaan. Dit leidt to de volgende kernachtige definitie van het begrip architectuur:

> De architectuur van een systeem is de samenhang van elementen die ervoor zorgt dat er globale systeemeigenschappen ontstaan.

3 Systeemdimensies

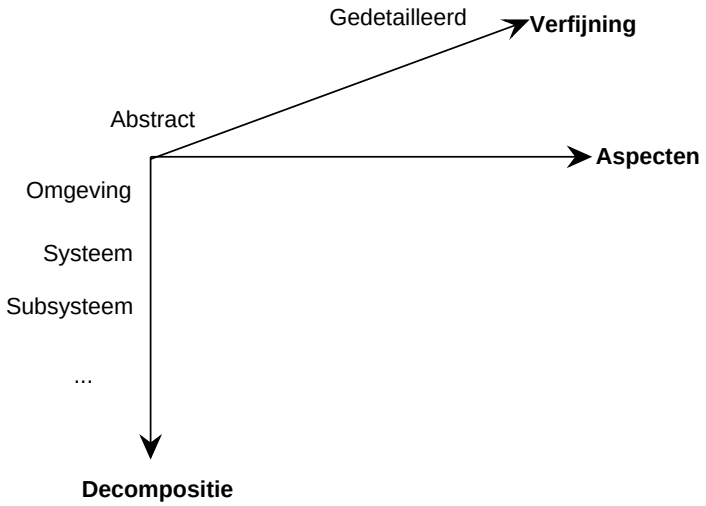
Een architectuur kan vanuit vele perspectieven beschreven worden. Om het begrip hanteerbaar te maken, wordt door meerdere auteurs een aantal dimensies onderscheiden waarbinnen architectuurkeuzes gemaakt kunnen worden. Die dimensies worden meestal een *architectuurraamwerk* genoemd. Ik bespreek in deze paragraaf een architectuurraamwerk dat gedefinieerd is op basis van een analyse van ongeveer 25 methodes voor software-ontwerp en een analyse van raamwerken voor industrieel product-ontwerp en van *systems engineering* [2, 3, 7, 8, 9, 16, 21, 22]. Het raamwerk voor systeemarchitectuur bestaat uit drie systeemdimensies en twee procesdimensies. Figuur 1 laat de drie systeemdimensies zien. Verderop bespreek ik de twee procesdimensies.

3.1 Decompositie

Dit is de dimensie waarin een systeem in subsystemen gedecomponeerd wordt die samen de gewenste systeemeigenschappen creëeren.

3.1.1 Conceptuele en fysieke architectuur

Bij puur fysieke systemen is er sprake van één soort decompositie: De componenten zijn fysiek bevat in het systeem. De componenten zijn kleiner en lichter dan het geaggregeerde systeem, en de meeste componenten zijn onzichtbaar voor waarnemers die buiten het systeem staan.



Figuur 1: De drie systeemdimensies van het raamwerk.

Voor organisaties en softwaresystemen is de situatie complexer omdat dit *conceptuele* systemen zijn, gerealiseerd in fysieke systemen. Een organisatie heeft fysieke componenten, zoals kantoorgebouwen, parkeerterrein, een centrale verwarming, electriciteit en de kabels voor telecom en dataverkeer; productiebedrijven hebben fysieke componenten zoals een productiestraat en magazijn. Maar een organisatie vindt zijn identiteit niet in deze fysieke componenten, maar in een stelsel van afspraken waarin vastgelegd heeft welke afdelingen bestaan, welke rollen er gespeeld moeten worden, welke taken uitgevoerd moeten worden en hoe de verantwoordingslijnen liggen. Deze conceptuele structuur is essentiëler dan de fysieke structuur. Een organisatie kan al zijn fysieke onderdelen vervangen door andere fysieke onderdelen, zonder van identiteit te veranderen. Het kan daarentegen zijn conceptuele structuur niet veranderen zonder een deel van zijn identiteit te veranderen.

Voor software geldt hetzelfde. Computers bestaan uit metaal, plastic en silicium en de decompositiehiërarchie voor deze onderdelen is zoals die van alle fysieke systemen: grotere, zwaardere onderdelen bestaan uit kleinere, lichtere onderdelen. Software daarentegen is conceptueel vanaf het niveau van bits en hoger. Door de interpretatie van een bepaalde spanning of een bepaalde magnetisering als een “0” of een “1” verlaten we de fysieke wereld en betreden we de conceptuele wereld. Booleaanse logica is geen onderdeel van de natuurkunde maar van de logica—en de logica houdt zich bezig met concepten. Het is in de informatica de gewoonte om alles wat op het niveau van middleware of lager zit “fysiek” te noemen. In dit artikel doe ik dat niet en gebruik het woord “fysiek” zoals het in de rest van de wereld gebruikt wordt, nl. om fysieke zaken aan te duiden, dingen die een massa en een omvang hebben. Dit zijn zaken die in de natuurkunde, scheikunde en biologie bestudeerd worden.

Een gevolg hiervan is dat organisaties twee soorten decompositie hebben: conceptueel en fysiek. Organisaties hebben dus ook twee decompositie-dimensies, een conceptuele en een fysieke.

- De fysieke architectuur van een organisatie bestaat uit de geografische locatie van zijn gebouwen, de architectuur van deze gebouwen en uit de mensen die voor de organisatie werken en de machines die door de organisatie gebruikt worden. Niet het eerste waar we aan denken als we aan organisatie-architectuur denken, maar bij nader inzien toch een belangrijke dimensie. Eindhoven of Amsterdam als locatie voor het hoofdkantoor? Zoetermeer of Den Haag als locatie voor het ministerie? Een kantoor op de bovenste verdieping of ergens middenin? Locatie van het magazijn centraal of decentraal?

- De conceptuele architectuur van een organisatie bestaat uit de afdelingen, verantwoordingslijnen, werkprocedures en duizenden andere conceptuele structuren die als geheel de organisatie uitmaken. Om te zien hoe groot de conceptuele structuur is die wij "organisatie" noemen, stelt u zich voor dat u deze structuur zou moeten uitleggen aan een wezen van een vreemde planeet uit een ver verwijderd sterrenstelsel. Wat dat wezen zou waarnemen is de fysieke wereld bestaande uit afstanden, massa's, magnetische velden, biologische organismen, etc. Waar wij een commissievergadering zien, ziet dit wezen dus een verzameling biologische organismen die hete lucht produceren.

Ik reken ook werknemers tot de conceptuele architectuur van een organisatie. Het begrip *werknemer* is een juridisch gedefinieerd begrip zonder fysieke component. Ons buitenaards wezen ziet geen werknemers in vergadering maar organismen die geluiden uitstoten. Wel is het zo dat *werknemer* een abstracte rol is die door een mens gespeeld kan worden. Mensen staan op het snijvlak van de conceptuele en fysieke wereld omdat ze deel uitmaken van beide.

Meer in het algemeen is het een belangrijk element van een organisatie-architectuur om aan te geven hoe de conceptuele architectuur van de organisatie gerealiseerd is in de fysieke architectuur. Waar vinden welke processen plaats, waar werken de leden van de commissie, en in welk gebouw zit welke afdeling? De relatie tussen de conceptuele en fysieke architectuur is er een van *allocatie*. Allocatie is zelf een conceptueel gedefinieerde relatie die, samen met alle andere conceptuele structuren van een organisatie, door mensen geïmplementeerd wordt omdat zij deze structuren gebruiken om hun werkzame leven in te richten.

Software, zoals gezegd, heeft geen fysieke component. Het bestaat uit concepten zoals datatypes, Boolean logica, instructies, besturingsstructuren, etc. Wel is het zo dat ook software *gealloceerd* wordt aan fysieke componenten zoals computers, printers, routers, telefooncentrales en componenten van autos. Deze allocatie-relatie is uiteindelijk fysiek geïmplementeerd als *toestand* van hardware, bv. als een magnetiseringspatroon van een geheugenschijf.

De conceptuele structuur van bedrijven en software wordt meestal door middel van diagrammen gespecificeerd. Voor software staat op dit moment de Unified Modeling Language (UML) sterk in de belangstelljng, die zich als de facto standaardnotatie voor de architectuur van object-georiënteerde software opwerpt. De UML bevat een notatie voor de componentenstructuur van een programma (het *component diagram*) en een notatie voor de fysieke architectuur van het netwerk (het *deployment diagram*), met een manier om aan te geven hoe softwarecomponenten aan fysieke resources gealloceerd worden [19]. In mijn terminologie zijn software-componenten onderdeel van de conceptuele structuur van de software en geeft het deployment diagram de echte fysieke structuur van het netwerk aan.

Bedrijven kunnen een deel van hun conceptuele architectuur realiseren in software. Dit gebeurt sinds het begin van de automatisering in de jaren zestig van de twintigste eeuw.

Het begon met processen die toen reeds op mechanistische wijze uitgevoerd werden, zoals salarisadministratie. Met de opkomst van netwerktechnologie in de laatste vijftien jaar worden in toenemende mate interacties tussen organisaties onderling en tussen organisaties en klanten gerealiseerd in software, eerst met EDI technologie, later op flexibeler wijze met Internettechnologie.

3.1.2 Componenten en lagen

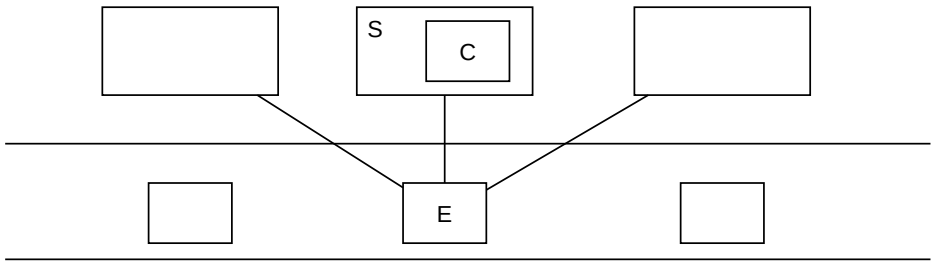
De klassieke systeembenadering gaat van fysieke decompositie uit en legt de nadruk op de manier waarop systeemeigenschappen uit de interacties van de componenten ontstaan. Het idee van allocatie is niet alleen van toepassing op de relatie tussen de conceptuele en fysieke architecturen maar kan ook gebruikt worden binnen elk van deze twee architecturen. De algemene idee is als volgt:

- In een top-down ontwerpbenadering stellen we gewenste systeemeigenschappen vast, decomponeren we het systeem in onderdelen en stellen voor elke systeemeigenschap vast welke interacties van componenten er voor zorgen dat de systeemeigenschap gerealiseerd wordt.
- In een bottom-up benadering beginnen we met de eigenschappen van componenten en kijken we welke systeemeigenschappen we daarmee kunnen realiseren. Men spreekt dan vaak over een *synergie* tussen de componenten, die op systeemniveau een meerwaarde creëert.

In de systeembenadering spreekt men in beide gevallen van *flowdown* van een systeemeigenschap naar subsysteemeigenschappen. Hiermee wordt bedoeld dat een systeemeigenschap wordt gerealiseerd door de eigenschappen van een aantal subsystemen en de interactie daartussen.

Een tweede centraal begip in de systeembenadering is dat de omgeving minstens even belangrijk is als het systeem zelf. Systeemeigenschappen ontstaan niet alleen door de interacties van de componenten onderling, maar door de interacties van de componenten met elkaar *en* met de omgeving. Beschouw bijvoorbeeld de eigenschap van betrouwbaarheid. Een bedrijf is betrouwbaar omdat de interne procedures en rollen gezamenlijk betrouwbaarheid realiseren; maar het is alleen maar in staat om die betrouwbaarheid te laten zien in interactie met externe partijen zoals klanten en toeleveranciers. Bovendien zal het deze eigenschappen niet in elke omgeving hebben. In extreme omstandigheden zoals natuurrampen of oorlogen zal de eigenschap verdwijnen. Systeemeigenschappen moeten daarom altijd in relatie tot omgevingseigenschappen bezien worden. Om een bedrijfsarchitectuur te begrijpen, moet de omgevingsarchitectuur mede gemodelleerd worden. De decompositiehiërarchie begint dus altijd één niveau hoger dan het beschouwde systeem, nl. bij de omgeving waar het systeem onderdeel van is.

We kunnen zowel in een fysieke als in een conceptuele architectuur systeemeigenschappen niet alleen realiseren door middel van flowdown naar interne componenten, maar ook door *outsourcing* naar externe systemen in een z.g. lagenarchitectuur. Een lagenarchitectuur is een verzameling systemen die in lagen geordend is zodanig dat systemen in een lagere laag diensten aanbieden aan systemen in hogere lagen. Het verschil tussen een component C van systeem S en een entiteit E in een onderliggende laag is dat C alleen ten dienste van S staat terwijl E in de onderliggende laag diensten aanbiedt aan meer systemen dan alleen aan S



Figuur 2: Componenten versus lagen.

(figuur 2). Dit gebeurt in organisaties zowel als in software. De schoonmaakdienst wordt uitbesteed aan een extern bedrijf, de logistiek wordt uitbesteed aan een vervoersbedrijf, en het is niet ondenkbaar dat in de toekomst banken het administreren van bankrekeningen uitbesteden aan een extern bedrijf. In software is de lagenarchitectuur gemeengoed, met als meest in het oog springende voorbeeld de ISO OSI stapel. Op hoger aggregatieniveau herhaalt dit zich doordat organisaties applicaties kunnen uitbesteden aan application service providers.

Mengvormen van de twee structureringsprincipes —flowdown naar componenten, outsourcing naar onderliggende lagen— zijn ook mogelijk. Een organisatie kan een IT subsysteem bevatten die diensten aanbiedt aan alle andere onderdelen van de organisatie. Binnen de organisatie is er dan spraken van een lagenarchitectuur. Ook een enkel softwaresysteem kan intern in lagen opgebouwd zijn. Het klassieke onderscheid tussen informatiesystemen en de technische infrastructuur is hier ook een voorbeeld van:

- Informatiesystem zijn conceptuele systemen gerealiseerd door (gealloceerd aan) mensen en computersystemen die samenwerken om informatievoorzieningsdiensten te leveren.
- De technische infrastructuur is een conceptueel systeem dat diensten levert waarmee de software-componenten van informatiesystemen gerealiseerd kunnen worden.
- De technische infrastructuur zelf is weer in een fysieke structuur van blik, silicium en plastic gerealiseerd.

De decompositiedimensie van een systeem S stopt waar taken aan externe dienstverlenende systemen in een onderliggende systeemlaag afgestoten worden. Zolang S —een organisatie of een software-systeem— in onderdelen gedecomponeerd wordt moet aangegeven worden hoe de systeemeigenschappen door de onderdelen gerealiseerd worden. Dit herhaalt zich voor elk onderdeel apart totdat taken aan een onderliggende laag dienstenaanbieders buiten S worden afgestoten. Decompositie stopt daar, en de rol van allocatie wordt overgenomen door een contract tussen S en de dienstenaanbieder.

De twee decompositiedimensies laten zien dat er verschillende soorten alignment langs deze dimensies mogelijk zijn:

- Alignment van (conceptuele resp. fysieke) subsystemen met de doelen en gewenste eigenschappen van het grotere (conceptuele resp. fysieke) systeem waar ze onderdeel van zijn.

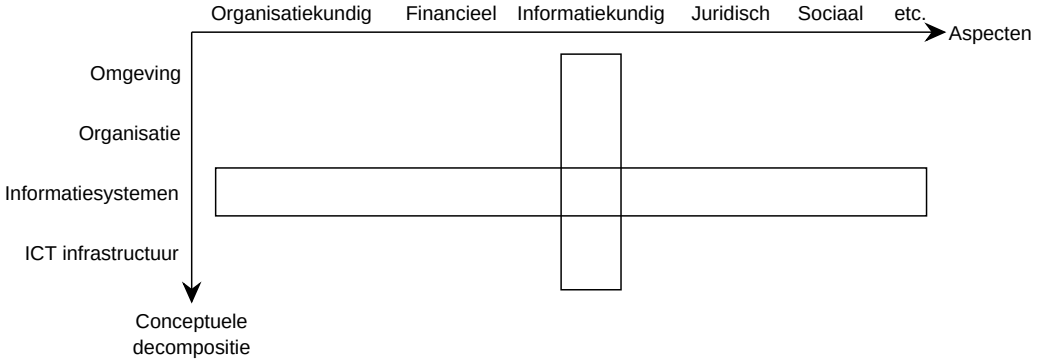
- Alignment van de conceptuele en fysieke architecturen onderling.

3.2 Aspect

Een aspect van een systeem is een verzameling eigenschappen van dat systeem. Een gebouw heeft vele aspecten: Het bouwfysische aspect, het gebruiksaspect, het financiële aspect, het juridische aspect, het esthetische aspect, etc. Elk van deze gebouwaspecten bestaat uit een aantal eigenschappen. Bouwfysische eigenschappen zijn bv. de sterkte van de vloeren en de warmtecapaciteit van de kamers; voorbeelden van gebruikseigenschappen zijn het gemak waarmee de de ruimtes in het gebouw toegankelijk zijn, de functionaliteit van de voorzieningen in de kamers van het gebouw, de hoeveelheid licht in de kamers, etc.; eigenschappen die onder het juridische aspect vallen zijn bijvoorbeeld de eigendomsrelaties van het gebouw, de aansprakelijkheid voor de kwaliteit van onderdelen van het gebouw, onderhoudscontracten, etc. Vaak is het zo dat verschillende aspecten van een systeem gekoppeld zijn aan een bepaald specialisme, waarin kennis over de eigenschappen die onder dat aspect vallen bestaat. Specialisten die verschillende aspecten van een gebouw kunnen analyseren zijn de bouwfysicus, binnenhuisarchitect, de hypotheekspecialist, een jurist, etc.

Verschillende aspecten zijn niet disjunct maar vullen elkaar aan. Vaak is het zo dat eigenschappen op verschillende manieren geclassificeerd kunnen worden en dus onder verschillende aspecten beschouwd kunnen worden. Zo kan het een eigenschap van een informatiesysteem zijn dat het in staat is queries over afgesloten hypotheken te beantwoorden. Die eigenschap kan beschreven worden onder het gebruiksaspect (heeft de query een gebruiksvriendelijke interface?), het juridische aspect (worden privacyregels gerespecteerd?), het beveiligingsaspect (geen ongeautoriseerde toegang?), het informatiekundige aspect (informatie opgeleverd die gewenst is?), etc.

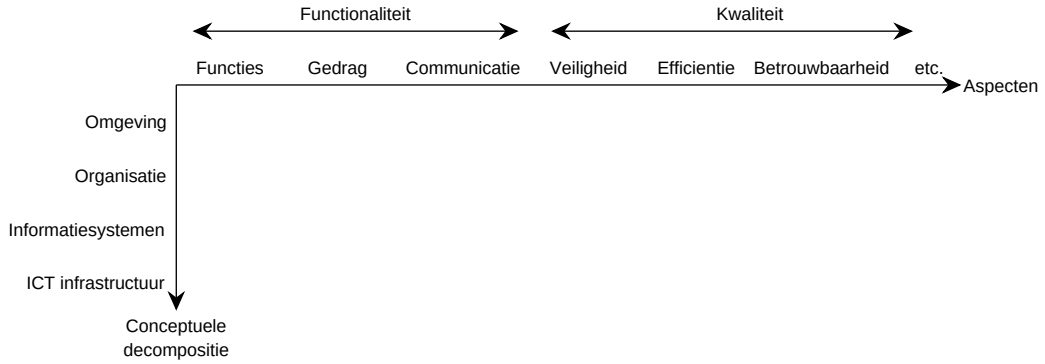
Ook organisaties en software hebben vele aspecten. Sommige aspecten zijn van toepassing op zowel organisaties als software, zoals bijvoorbeeld informatiekundige, juridische, sociaal-psychologische en economische aspecten. Andere zijn specifiek voor alleen software of alleen organisaties. Het ergonomische aspect is wel op software en niet op organisaties van toepassing, het culturele aspect is niet op software en wel op organisaties van toepassing.



Figuur 3: Voorbeelden van algemene systeemaspecten.

Figuur 3 laat een aantal aspecten van organisaties en software zien en geeft vier aggre-

gatieniveaus aan, bestaande uit de omgeving en drie organisatie-niveaus. Alle vertoonde aspecten betreffen zowel de omgeving als alle aggregatieniveaus in een organisatie. Bijvoorbeeld wordt informatie uitgewisseld tussen een organisatie en zijn omgeving, tussen organisatie-onderdelen, en tussen informatiesystemen van een organisatie. Dit is aangegeven door de verticale rechthoek in figuur 3. Omgekeerd hebben informatiesystemen, als subsystemen van een organisatie, niet alleen een informatiekundige aspect maar ook een organisatiekundig, financieel, juridisch, sociaal etc. aspect. Dit is aangegeven door de horizontale rechthoek in figuur 3. De intersectie van de twee rechthoeken geeft het deel van de informatievoorziening aan dat geformaliseerd is in informatiesystemen.



Figuur 4: Voorbeelden van technische systeemaspecten.

Figuur 4 laat een aantal meer specialistische aspecten van systemen zien. Elke organisatie heeft functies voor zijn omgeving en elk informatiesysteem heeft functies voor *zijn* omgeving. (De omgeving van een informatiesysteem bestaat uit de rest van de organisatie en mogelijk uit onderdelen van de externe omgeving van de organisatie.) De drie functionaliteitsaspecten, t.w. functies, gedrag en communicatie, zijn als volgt gedefinieerd:

- *Functies* zijn de diensten die de software of organisatie aan zijn omgeving levert.
- *Gedrag* is de wijze waarop deze diensten in de tijd geordend zijn. Dit betreft bijvoorbeeld de reactie op temporele gebeurtenissen (de productie van de maandelijkse salarisrun, de productie van herinneringen wanneer een deadline gepasseerd is, etc.), het protocol waarmee met andere systemen (software of organisaties) gecommuniceerd wordt, etc.
- *Communicatie* is de wijze waarop diensten in de "ruimte" geordend zijn, of anders gezegd de wijze waarop in de uitoefening van deze diensten met andere systemen (software of organisaties) samengewerkt wordt.

Het is de gewoonte om naast deze drie modellen ook altijd een *gegevensmodel* op te leveren. Dat model zit reeds in figuur 4 verborgen, omdat het deel uitmaakt van het omgevingsmodel. In een echt implementatievrije beschrijving van een informatiesysteem geven we niet de interne structuur van het systeem aan maar definieëren we wat de betekenis is van de berichten die het systeem met zijn omgeving uitwisselt; en die betekenis ligt in de omgeving van het systeem. Om de betekenis van de berichtenuitwisseling met de omgeving vast

te leggen maken we dus een omgevingsmodel. Een simpel omgevingsmodel bevat altijd een entiteitenmodel van de omgeving, dat aangeeft over welke entiteiten in de omgeving gecommuniceerd wordt.

In klassieke informatiesystemen speelt de specificatie van gedrag en communicatie een ondergeschikte rol. Met de opkomst van netwerktechnologie en toepassingen in EDI en e-commerce worden ook die aspecten van belang.

Figuren 3 en 4 maken duidelijk dat een architectuur niet geïsoleerd onder één aspect beschouwd kan worden. Alignment langs de aspectdimensie betreft het afstemmen van de organisatiekundige, financieële, informatiekundige en andere aspecten van een organisatie op al zijn niveaus, inclusief het niveau van informatiesystemenen en ICT infrastructuur. Gelukkig zijn niet alle aspecten in elke situatie relevant. De keuze van relevante aspecten wordt bepaald door ervaring, inzicht, tijdsdruk, politiek krachtenveld, etc.

3.3 Verfijning

De verfijningsdimensie ordent architectuurbeschrijvingen van abstract (weinig detail) naar gedetailleerd. Belangrijk is dat deze orthogonaal is ten opzichte van zowel de decompositie- als de aspectdimensie. Beschouwen we een organisatie op hoog abstractieniveau, dan kijken we naar zijn missie, rol in de omgeving, positionering in de markt etc. Beschouwen we de organisatie op een laag abstractieniveau, dan beschrijven we de details van de interactie van de organisatie met zijn omgeving. Op het hoge niveau is de beschouwingswijze strategisch, op het lage niveau is het operationeel.

Precies hetzelfde onderscheid tussen hoog en laag abstractiniveau bestaat op lagere decompositieniveaus. Een hoog niveau beschrijving van een softwaresysteem geeft de missie van het systeem en de rol van het systeem in zijn omgeving (andere systemen binnen en buiten het bedrijf). Op laag niveau beschrijven we de gedetailleerde interactieprotocollen die het systeem met zijn omgeving heeft. Ook software-objecten kunnen we abstract of gedetailleerd beschrijven: op hoog niveau de missie en verantwoordelijkheden van het object, op laag niveau zijn communicatieprotocollen en uitgewisselde boodschappen.

Ook hier is er sprake van afstemmingsrelaties tussen strategisch en operationeel niveau. Alignment in zijn volle complexiteit houdt in dat op elk aggregatieniveau van de organisatie, zowel conceptueel als fysiek, de strategische visie op alle aspecten afgestemd moet zijn met de operationele uitwerking.

4 Procesdimensies

Behalve de drie systeemdimensies van het architectuurrraamwerk zijn der twee procesdimensies, nl. ontwerp en implementatie.

4.1 Ontwerp

Systemen worden ontworpen. De essentie van systeemontwerp zoals ik die hier in brede zin definieer is dat tussen alternatieven gekozen wordt op basis van voorspelde eigenschappen van de ontwerpkeuzes. Ontwerpen in deze algemene zin bestaat uit probleemanalyse en oplossingsspecificatie [16]:

- *Probleemanalyse* is de analyse van de huidige situatie, inclusief belanghebbenden, problemen, doelen, trends en randvoorwaarden.
- *Oplossingsspecificatie* bestaat uit
  - het *genereren* van mogelijke oplossingen,
  - het zo goed mogelijk *voorspellen* van de eigenschappen van die oplossingen, en
  - het *evalueren* van de voorspelde eigenschappen op hun probleemoplossend vermogen.

Zoals eerder uitgelegd is de ontwerper in brede zin een architect en omgekeerd is de architect een ontwerper. Een ontwerper, en dus een architect, ontwerpt, maar bouwt niet. Zijn taak is belanghebbenden, problemen, doelen, trends en randvoorwaarden te koppelen aan een samenhangende oplossing. De gespecificeerde architectuur moet geanalyseerd zijn op zijn probleemoplossend vermogen. De architect zorgt dus voor de alignment tussen systeem en omgeving.

De ontwerpdimensie geeft de logische structuur van de ontwerpactiviteit weer. Dit is ook de structuur die we gebruiken als we een rechtvaardiging van een ontwerpkeuze geven: Alternatieven die overwogen zijn, voorspelde eigenschappen op basis waarvan geëvalueerd is, probleemanalyse die de basis was voor de gemaakte keuze. de bovengenoemde ontwerptaken geven dus in feite de struktuur van een *ontwerpargument* weer.

De relatie tussen de rechtvaardigingsdimensie en de drie systeemdimensies is dat de drie systeemdimensies ontwerpkeuzes klassificeren. De architect maakt keuzes in de conceptuele en fysieke decompositie van een systeem, maakt deze keuzes voor bepaalde eigenschappen die onder bepaalde aspecten vallen, en maakt deze keuzes op verschillende abstractieniveaus. Elk van deze keuzes moet gerechtvaardigd worden en deze rechtvaardiging moet de hierboven aangegeven structuur hebben: probleemanalyse, oplossingsspecificatie, eigenschappenanalyse en evaluatie.

4.2 Implementatie

De tweede procesdimensie is implementatie. Verandering worden niet alleen ontworpen maar ook geïmplementeerd. Dit kan op vele manieren: lineair, evolutionair, incrementeel, experimenteel, parallel, etc.

- In een lineair implementatieproces wordt eerst de volledige oplossing gespecificeerd, die dan dan top-down als geheel wordt geïmplementeerd.
- In een incrementeel implementatieproces wordt eerst de volledige oplossing gespecificeerd, die dan stuksgewijs geïmplementeerd wordt.
- In een evolutionair proces wordt eerst een deel van de oplossing gespecificeerd en geïmplementeerd, die dan geëvalueerd wordt. Dat leidt tot een verbeterde oplossingspecificatie, etc.
- Bij een experimentele implementatie wordt met een mogelijke oplossing geëxperimenteerd door het in een reëele maar niet-kritische situatie toe te passen.

- In een parallele implementatie vinden ontwerp en implementatie tegelijk plaats en zijn er permanente feed forward en feedback lussen tussen de ontwerp- en implementatie-processen.

De implementatiestrategieën geven verschillende manieren aan om ontwerpbeslissingen in de tijd te ordenen. In die zin is de implementatiedimensie orthogonaal aan de ontwerpdimensie. Elk van de implementatie-activiteiten bevat ontwerptaken die ieder de logische structuur van het boven geschetste ontwerpargument hebben.

In al deze gevallen speelt alignment een centrale rol. Wat geïmplementeerd wordt is vaak niet precies, en soms helemaal niet, wat gespecificeerd was. De omgeving is ondertussen veranderd, de voorspelde eigenschappen zijn toch niet helemaal, of helemaal niet, gerealiseerd, etc. Essentieel in elke implementatiestrategie is dus de evaluatie achteraf, waarin bekeken wordt hoe het gerealiseerde ontwerp in de praktijk functioneert. Het gaat hier niet om de alignment die ontworpen is, maar om de alignment die gerealiseerd is. Hoewel dit niet in de eerste plaats een taak van de architect-ontwerper is, is het wel belangrijk dat de architect informatie verzamelt over deze evaluaties. Dit zal in de toekomst de kwaliteit van zijn voorspellingen verbeteren. Ontwerpen onder architectuur is een continu leerproces.

5 Toepassing

5.1 Alignment

In het voorgaande is al aangegeven dat het raamwerk een klassificatie geeft van verschillende alignmentdoelstellingen:

- Alignment van verschillende decompositieniveaus in de conceptuele en in de fysieke architectuur.
- Alignment tussen de conceptuele en fysieke architectuur.
- Alignment tussen een systeem en zijn omgeving. In een lagenarchitectuur omvat dit alignment met toeleveranciers van diensten in een onderliggende laag.
- Alignment tussen ontwerpkeuzes in verschillende aspecten van het systeem.
- Alignment tussen strategische en operationele keuzes, die op een hoog resp. laag abstractieniveau gemaakt worden.

Deze alignments maken onderdeel uit van de alignment die een architect tot stand probeert te brengen tussen een probleemanalyse en de oplossingsarchitectuur. Ze moeten na implementatie van de architectuur geëvalueerd worden opdat toekomstige ontwerpen een betere alignment tot stand kunnen brengen.

Het raamwerk is een handvat voor klassificatie en taakverdeling maar geeft geen richtlijnen voor het maken van keuzes. Dit is een eigenschap van raamwerken in het algemeen. Ik kom hierop terug nadat ik mijn raamwerk met andere raamwerken vergeleken heb.

5.2 Een mid-office architectuur

Parol [14] schetst een mid-office architectuur voor bankverzekeraars.

- De back office bevat informatiesystemen over zorg, leven, hypotheken, kredieten, schade, beleggen, financiën en personeelszaken.
- De front-offices zijn de medewerkers, agenten en web interfaces die met de klant communiceren.
- De mid-office presenteert de informatie uit de back office op uniforme wijze naar de front-offices. De back-office informatie wordt door de mid office gegroepeerd in acceptatieregels, commerciële producten, contracten, consumentgegevens, tussenpersoonsgegevens en contactgegevens.

Parol's beschrijving is als volgt gepositioneerd langs de drie systeemdimensies.

1. Het artikel schetst een conceptuele, geen fysieke architectuur op het niveau van informatiesystemen.
2. De architectuur representeert de aspecten van functionaliteit en communicatie (figuur 4) en geen andere.
3. De beschrijving van de architectuur in Parol's artikel heeft een hoog abstractieniveau: het artikel geeft de missie en doelstelling van van de front office, midoffice en back office (dit zijn de hoog-niveau functies) en geeft in een paar diagrammen weer welke communicaties er zijn.

Kijken we nauwkeuriger naar de decompositiedimensie van de door Parol geschetste architectuur, dan zien we dat hij drie niveaus onderscheidt: een schets van de omgeving en de bedrijfsprocessen heet *conceptueel*, de vertaling naar applicaties die de processen ondersteunen heet *logisch*, en de vertaling naar onderliggende implementatietechnologie heet *fysiek*. In mijn terminologie zijn dit drie aggregatieniveaus in de conceptuele decompositiedimensie. Deze niveaus hebben een bedrijfsinterne lagenstructuur: de implementatietechnologie ondersteunt verschillende applicaties binnen het bedrijf en de applicaties ondersteunen verschillende processen in het bedrijf.

Op alle drie niveaus worden ontwerpkeuzes gemaakt: bedrijfsprocessen worden heringericht om aan het mid-office concept gestalte te geven, applicaties moeten gespecificeerd worden en de juiste netwerk-infrastructuur moet beschikbaar gemaakt worden. De gemaakte ontwerpkeuzes kunnen leiden tot herimplementatie met zelfgebouwde of ingekochte technologie.

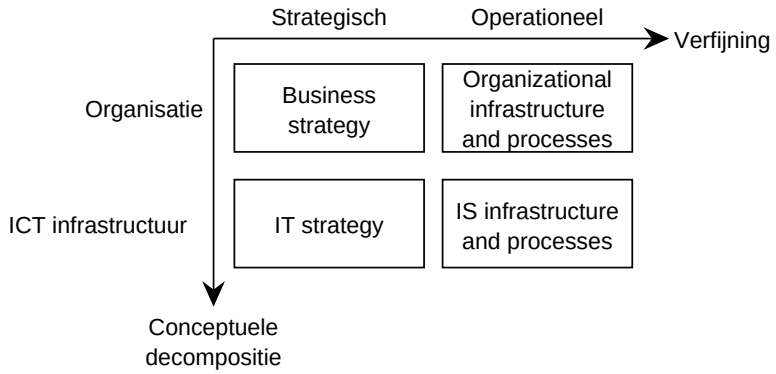
Parol beschrijft ook elementen van de twee procesdimensies van de voorgestelde architectuur.

1. Hij geeft een aantal karakteristieken van de probleemsituatie, nl. de behoefte om 7x24 uur beschikbaar te zijn en om doelgroepgericht in plaats van productgericht te werken. Daarnaast geeft hij een aantal karakteristieken van de mid-office oplossing: gegevens worden optimaal geclusterd zowel in de mid office als in de back office, distributiekanalen worden uniform aangestuurd, etc. Impliciet wordt op basis van deze eigenschappen voorspeld dat de mid-office architectuur aan de gestelde doelen beantwoordt.
2. Tenslotte geeft Parol enkele implementatierichtlijnen, zoals de volgorde waarin front- mid- en back office gerealiseerd moeten worden.

Samenvattend kan de door Parol geschetste architectuur volledig volgens het door mij geschetste raamwerk gestructureerd worden. Dit illustreert de functie van het raamwerk: het ordenen van architectuurspecificaties en van de ontwerpbeslissingen die in de specificatie gemaakt zijn.

6 Vergelijking met Andere Raamwerken

Het klassieke alignmentmodel van Henderson en Venkatraman [10] kan eenvoudig op mijn architectuurrraamwerk afgebeeld worden. Figuur 5 laat het resultaat zien. Het model van Henderson en Venkatraman is hierbij rond de NW-ZO diagonaal gespiegeld. Het moet hier



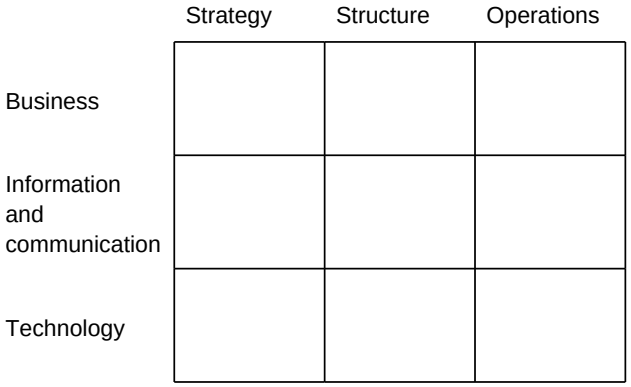
Figuur 5: Het alignment model van Henderson en Venkatraman [10] afgebeeld op het architectuurraamwerk.

benadrukt worden dat elke strategische beschouwingswijze van een systeem, of het nu een organisatie of een IT systeem is, de omgeving van het systeem bij de observaties betrekt.

Maes et al. [13] breiden het model van Henderson en Venkatraman uit met een structuurlaag tussen het strategische en operationele niveau, en met een informatie- en communicatielaag tussen organisatie en technologische infrastructuur. Zie figuur 6, op dezelfde manier georiënteerd als figuur 5. Deze uitbreiding voegt twee elementen toe vanuit de decompositie- resp. de aspectdimensie.

- Een blik op figuur 5 maakt duidelijk dat de structuurlaag van Maes et al. niet tussen het strategische en operationele verfijningsniveau ingepast moet worden, maar een aparte dimensie is, nl. de decompositiedimensie. Vermenging van deze twee dimensies verduistert het feit dat op we op elk structuurniveau van een organisatie strategisch kunnen nadenken over missie, omgeving en lange-termijn ontwikkelingen en op elk niveau strategieën gedetailleerd kunnen uitwerken.

- De informatie- en communicatielaag tussen organisatie en technologie is een aspect die aan de decompositiedimensie toegevoegd wordt. Maar zoals figuren 3 en 4 laten zien, is informatie en communicatie een aspect van de organisatie (en van de omgeving) op elk niveau van aggregatie. Het uitgebreide raamwerk van Maes et al. dus een representatie van bepaalde elementen van het door mij voorgestelde raamwerk.



Figuur 6: Het door Maes et al. uitgebreide Henderson-Venkatraman framework.

Maes et al. [13] voegen nog een vierde dimensie toe, namelijk die van de ontwikkelfasen, t.w. de contextuele, conceptuele, logische en fysieke fases, en de transformatie naar de nieuwe situatie. We zien hier de twee procesdimensies en een systeemdimensie in één geschoven.

- De contextuele tot en met de fysieke fases vermengen de ontwerpdimensie met de decompositiedimensie. De ontwerpdimensie omvat probleemanalyse en -oplossingsspecificatie, analyse en -evaluatie. Leggen we deze top-down langs de decompositiedimensie dan ontstaat het beeld van een ontwerpproces waarin we eerst de context analyseren en dan top-down een oplossing specificeren door eerst bedrijfsprocessen te modelleren (het "conceptuele niveau"), applicatie-architectuur te ontwerpen (het "logische niveau") en tenslotte deze op een ICT infrastructuur af te beelden (het "fysieke niveau").
- De transformatiefase is de tweede veranderdimensie, door mij implementatie genoemd. In het raamwerk van Maes et al. vind deze na de ontwerpfase plaats.

Ook hier worden er dus geen nieuwe begrippen of dimensies toegevoegd.

7 Discussie en Conclusies

Mijn raamwerk geeft, zoals andere raamwerken, slechts een classificatie van systeemeigenschappen en procesdimensies. De systeemeigenschappen zijn geclassificeerd naar decompositieniveau, aspect en verfijningsniveau (abstract naar gedetailleerd) en de geïdentificeerde procesdimensies zijn de ontwerpcyclus (probleemanalyse en oplossingsspecificatie) en het implementatieproces (een cyclus van implementeren en evalueren). Dat geeft een manier om beslissingen te ordenen maar niet om beslissingen te maken. Er zijn echter ook reeds veel ontwerprichtlijnen voor goede, samenhangende organisatie- en ICT architecturen. Adriaanse en Parol geven bijvoorbeeld richtlijnen voor het ontwerp van een mid-office gebaseerde multi-channel architectuur [1, 14]. Het hier gepresenteerde raamwerk maakt het mogelijk deze richtlijnen op gestructureerde wijze te presenteren. Zo bezien kan mijn raamwerk fungeren als een ontsluitingsmechanisme voor ontwerprichtlijnen per aggregatieniveau, aspect, en verfijningsniveau.

Er zijn aanzetten tot een domein-onafhankelijke verzameling van principes van goed architectuur-ontwerp [15]. Het nadeel van een domein-onafhankelijke benadering is dat de principes zo algemeen zijn dat ze bijna inhoudsloos worden. Beter is het om ze per domein te verzamelen, waarbij ik met *domein* een probleemklasse bedoel (bijvoorbeeld informatievoorzieningsproblemen, besturingsproblemen, communicatieproblemen, etc.). Dit sluit aan bij recente ontwikkelingen in de requirements engineering om herbruikbare probleemraamwerken te definiëren [11, 12]. Jackson's probleemraamwerken zijn combinaties van domeinen (d.w.z. probleemklassen) met oplossingsrichtingen. Behalve de drie systeemdimensies van het raamwerk zal dan dus het domein als ingang voor het vinden van ontwerprichtlijnen gebruikt kunnen worden. Dit is voorwerp van toekomstig onderzoek.

Noten

[1] Het woord "architect" komt van het grieks *architektōn*, van *archi-* (voornaamste) + *tektōn* (timmerman, handwerksman, kunstenaar, schepper), verwant met *technè* (techniek). (Van Dale's *Etymologisch Woordenboek*, Van Dale Lexicografie, 1991.) Dit wijst reeds op het feit dat een architect overzicht moet houden op zowel het gebouw als de bouwwerkzaamheden.

[2] In dit stuk is "hij" synoniem met "hij of zij", "zijn" synoniem met "zijn of haar" en "hem" synoniem met "hem of haar".

[3] Het Engelse woord *design* geeft goed de intentie van het woord "ontwerp" weer. Een *design* kan zijn

- A particular purpose held in view by an individual or group.
- A mental project or scheme in which means to an end are laid down.
- A preliminary sketch or outline showing the main features of something to be executed.
- An underlying scheme that governs functioning, developing, or unfolding ("the general design of the epic").
- A plan or protocol for carrying out or accomplishing something.
- The arrangement of elements or details in a product or work of art.
- A decorative pattern.

(*Webster's Ninth New Collegiate Dictionary*, Merriam-Webster, 1988.) Het woord werd voor het eerst in 1588 in de Engelse taal gesignaleerd, kort nadat Andrea Palladio (1508-80) het eerste boek met architectuurplaatjes publiceerde, bedoeld voor de meestal ongeletterde architecten van die tijd [17].

Het Engelse woord benadrukt bovendien de verwantschap met het woord *sign*, een zichtbaar teken dat verwijst naar iets dat op dit moment onzichtbaar is. Een *design* beschrijft het nu nog onzichtbare ontworpen systeem, laat zien hoe de systeemeigenschappen uit de elementen ontstaan, en functioneert als richtlijn voor het bouwen van dat systeem.

Het Nederlandse woord "ontwerp" heeft vrijwel dezelfde betekenis. Het bestaat uit *ont+werpen*, vermoedelijk van het Latijns *projectare*. (Van Dale's *Etymologisch Woordenboek*, Van Dale Lexicografie, 1991.)

Referenties

[1] P. Adriaanse. Multi-channel architectuur in een e-business omgeving. In *Landelijke Architectuurconferentie*, 23—24 november 2000.

[2] B.L. Archer. The structure of the design process. In G. Broadbent and A. Ward, editors, *Design methods in Architecture*, pages 76–102. Lund Humphries, 1969.

[3] M. Asimov. *Introduction to Design*. Prentice-Hall, 1962.

[4] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 1998.

[5] B. S. Blanchard and W. J. Fabrycky. *Systems Engineering and Analysis*. Prentice-Hall, 1990.

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *A System of Patterns: Pattern-Oriented Software Architecture*. Wiley, 1996.

[7] N. Cross. *Engineering Design Methods: Strategies for Product Design. Second Edition*. Wiley, 1994.

[8] A.D. Hall. *A Methodology for Systems Engineering*. Van Nostrand, 1962.

[9] A.D. Hall. Three-dimensional morphology of systems engineering. *IEEE Transactions on System Science and Cybernetics*, SSC-5(2):156–160, 1969.

[10] J.C. Henderson and N. Venkatraman. Strategic alignment: leveraging information technology for transforming operations. *IBM Systems journal*, 32(1), 1993.

[11] M.A. Jackson. *Software Requirements and Specifications: A lexicon of practice, principles and prejudices*. Addison-Wesley, 1995.

[12] M.A. Jackson. *Problem Frames: Analysing and Structuring Software Development Problems*. Addison-Wesley, 2000.

[13] R. Maes, D. Rijsenbrij, A. Truijens, and H. Goedvolk. Redefining business–IT alignment through a unified framework. In *Landelijke Architectuurconferentie*, 23—24 november 2000.

[14] A.J.A. Parol. E-architectuur: Het integreren van verschillende applicaties. In *Landelijke Architectuurconferentie*, 23-24 november 2000.

[15] E. Rechtin. *Systems Architecting: Creating and Building Complex Systems*. Prentice-Hall, 1991.

[16] N.F.M. Roozenburg and J. Eekels. *Productontwerpen, Struktuur en Methoden*. Lemma B.V., 1991.

[17] P. Routio. Arteology or the science of artefacts. Technical report, University of Art and Design Helsinki, April 1999. http://www.uiah.fi/projects/metodi/.

[18] M. Shaw and D. Garlan. *Software Architecture: Perspective on an Emerging Discipline*. Prentice Hall, 1996.

[19] UML Revision Task Force. *OMG UML Specification*. Object Management Group, march 1999. http://uml.shl.com.

[20] J. in't Veld. *Analyse van Organisatieproblemen: Een Toepassing van Denken in Systemen en Processen*, vijfde druk. Stenfert Kroese, 1988.

[21] R.J. Wieringa. *Requirements Engineering: Frameworks for Understanding*. Wiley, 1996. http://www.cs.utwente.nl/~roelw/RE1.ps.

[22] R.J. Wieringa. A survey of structured and object-oriented software specification methods and techniques. *ACM Computing Surveys*, 30(4):459–527, December 1998.