De pragmatiek van Architectuur

W.J. van den Heuvel
WJHeuvel@uvt.nl
Infolab, Dept. Informatiekunde
Universiteit van Tilburg

H.A. Proper
E.Proper@acm.org
Information Retrieval & Information Systems
Katholieke Universiteit Nijmegen

Gepubliceerd als:
W.-J. van~den Heuvel and H.A. Proper. De pragmatiek van architectuur.
Informatie, 44(11):12--14, 2002..

1. Inleiding

In de afgelopen jaren zijn er zowel binnen de industrie als binnen de wetenschappelijke gemeenschap belangrijke ontwikkelingen te zien geweest in het domein van ICT gerelateerde architecturen. Deze ontwikkelingen hebben zich gemanifesteerd op verschillende gebieden. Bijvoorbeeld binnen het gebied van methodieken voor het modelleren en ontwerpen van architecturen middels grafische en tekstuele representatie talen. Maar denk ook aan de formering van werkgroepen voor standaardisatie, bijvoorbeeld op het internationale vlak aan de architectuur-werkgroepen binnen het IEEE, en meer locaal in Nederland, de architectuur werkgroepen binnen het NGI, GIA en recentelijk het NAF. Daarnaast heeft de internationale onderzoeks- en industrie community actief gewerkt aan het ontwikkelen en testen van architectuur-raamwerken, welke het concept architectuur beschouwen vanuit verschillende, aan elkaar gerelateerde, ooghoeken.
Alhoewel deze ontwikkelingen in belangrijke mate hebben bijgedragen aan het definieren van de structuur en betekenis van het concept "architectuur" binnen organisaties, stellen wij in dit artikel dat de pragmatiek van architecturen en het "architectuur-denken" meer in het algemeen, tot nu toe onvoldoende is belicht.

In dit artikel over de pragmatiek van architectuur richten we ons natuurlijk vooral op ICT gerelateerde architectuur; ook wel "de digitale architectuur" genoemd [2]. In deze context zullen we regelmatig over systemen praten. De lezer dient te beseffen dat we hierbij systemen in de originele zin des woords bedoelen, en *niet* slechts de software-systemen. In de ICT wereld wordt systeem nog wel eens als synoniem voor software systeem gebruikt. In de context van architectuur, waar we niet alleen te maken hebben met software systemen, maar ook met organisatie systemen, business-systemen, etc., leidt dat al gauw tot verwarring.

2. Fundamenten van Architectuur

In het (nabije) verleden is er in diverse gremia uitgebreid gediscussieerd over de definitie van architectuur. In dit artikel gaan we deze discussie absoluut niet voortzetten maar nemen we de IEEE definitie van architectuur [1] als uitgangspunt. Hierin wordt architectuur, of liever gezegd de *architectuur beschrijving*, gedefinieerd als "de fundamentele organisatie van een systeem zoals deze wordt vormgegeven door zijn componenten, hun onderlinge verbanden als mede die met de omgeving, en de principes welke sturend zijn voor hun ontwerp en evolutie" (IEEE Std 1471, red. vertaald uit engels). Toch laat deze definitie een onbevredigend gevoel achter. Uit de definitie volgt wat precies een architectuur is in *syntactische* zin, en gaat daarbij in eerste instantie voorbij aan de *pragmatiek* van architectuur.

Met pragmatiek wordt grofweg de doelgerichte toepassing van het architectuur-concept bedoeld. Vanuit een pragmatisch perspectief reflecteren architecturen dus de intenties van bepaalde stakeholders in een organisatie. Pragmatiek kan worden beschreven doordat betekenis kan worden toegekend aan het medium (tekstueel of grafisch) dat wordt gebruikt voor het conceptualiseren van een architectuur. Wanneer architecturen zuiver semantisch worden beschouwd kan dit in principe vanuit 2 standpunten: het interne en het externe standpunt. Het interne standpunt schrijft de veroorloofde architectonische vormen voor, terwijl het externe standpunt bepaald of een architectuur geoorloofd is binnen een bepaalde systeemcontext. Algemeen wordt aangenomen dat de semantiek van architecturen kan worden gedefinieerd met gebruikmaking van een beperkte set van toegestane architectonische bouwstenen en relaties daartussen. Deze worden gedefinieerd in de syntax van een architectuur.

In de IEEE standaard wordt niet alleen een syntactische definitie van architectuur gegeven. Men gaat verder dan dat door een aantal manieren te identificeren waarop architectuur (beschrijvingen) als effectief middel ingezet kan worden:

1. Als een communicatiemiddel tussen de verschillende partijen die een belang hebben bij het beschouwde systeem.
2. Het bieden van een kader waarbinnen het beschouwde systeem in de toekomst kan evolueren.
3. Als basis voor het evalueren en vergelijken van alternatieve ontwerpen van een systeem.
4. Als plannings- en stuurinstrument voor de daadwerkelijke ontwikkeling en realisatie van het systeem.
5. Als eikpunt om de implementatie van een systeem aan te verifieren.

Feitelijk geeft bovenstaande dus een pragmatische definitie van architectuur weer. Met andere woorden, gericht op de vraag “wat kun je er mee?”. Naar onze mening is, geredeneerd vanuit de behoeften van de praktijk, deze laatste vraag ook veel belangrijker en relevanter dan de vraag wat een architectuur syntactisch gezien nu precies is. Hetgeen overigens los gezien moet worden van de noodzaak om de modellen waarmee een architectuur wordt vastgelegd te voorzien van een welgedefinieerde syntax en **semantiek**. Zonder dat wordt het moeilijk om, bijvoorbeeld, architecuur als communicatiemiddel in te zetten. Wat communiceer je immers?

Wanneer we voor een systeem een architectuur opstellen, dan kunnen we dit doen vanuit verschillende perspectieven. Perspectieven spelen en belangrijke rol bij het bepalen van de pragmatiek van een architectuur binnen een bepaalde context, bijv. een bankorganisatie die een architectuur wil opstellen voor de afhandeling van vragen mbt effecten-beheer van verzoek van een klant tot en met uitvoering, vereist een andere invulling van het concept architectuur dan een verzekeringsorganisatie die zijn informatiehuishouding op orde wil krijgen, bijv. rondom de notie van klantendossiers, middels een informatie-architectuur. De perspectieven die momenteel een dominante positie lijken in te nemen zijn:

- Een *procesperspectief*, vanwaar processen-aspecten die zich in het systeem afspelen benadrukken.
- Een *gegevensperspectief*, wat zich richt op de vraag welke gegevens er door het systeem gemanipuleerd zullen worden en wat de structuur en/of eigenschappen van deze gegevens zijn.
- Een *technologisch perspectief*, waarmee in kaart gebracht kan worden welke technologische componenten er in het systeem gebruikt worden, bijv. de systeem-topologie.
- Een *strategisch perspectief*, waarin de langere termijn visie (zeg 10 jaar) op het systeem naar voren komt.

Deze perspectieven kunnen niet los van elkaar worden gezien. Vaak is een combinatie van meerdere perspectieven gewenst om een systeem-architectuur vorm te geven, bijv. een proces- en gegevens-perspectief van een informatie-systeem. Diverse auteurs hebben even zoveel verschillende raamwerken van verschillende raamwerken voorgesteld om de perspectieven op een gestructureerde wijze te relateren. We noemen er hier enkele van de meest vooraanstaande:

- *Zachman* [8]: dit klassieke, generieke raamwerk is eind jaren tachtig binnen IBM ontwikkeld, en definieert maar liefst 36 verschillende perspectieven op bedrijfsbrede architecturen. Het raamwerk bestaat uit twee dimensies: (1) de rollen van de stakeholders tijdens een architectuur-ontwikkel proces, (2) en verschillende aspecten die van belang zijn voor het managen van het ontwikkelproces. De tweede dimensie bestaat uit de volgende aspecten: wat (data), hoe (functies), waar (netwerk), wie (personen), wanneer (tijd) en tenslotte het waarom (de motivatie) van een architectuur ontwikkel proces.
- *Tapscott & Caston* [9]: dit raamwerk is voorgesteld als middel voor organisaties die de omslag aan het maken zijn naar het werken met architectuur om ICT en bedrijfsvoering beter op elkaar af te stemmen. Het raamwerk onderscheidt vijf elkaar aanvullende, en contrasterende, perspectieven: business, work, information, application en technology.
- *RM-ODP* [10]: de ISO RM-ODP architecuur definieert en relateert (formeel) de volgende vier complementaire perspectieven: enterprise, informational, computational, engineering en technologie. Deze standaard richt zich met name op de specificatie van gedistribueerde informatie-systemen. De Model-Driven Architectuur welke nu door de OMG wordt ontwikkeld is geinspireerd door deze architectuur.
- *Kruchten's 4+1 raamwerk* [11]: dit raamwerk is afkomstig uit het domein van object-georienteerd modelleren, en meer in het bijzonder UML. Het raamwerk specificeert 4 verschillende perspectieven op een systeem: logical, process, development en physical, welke worden verbonden middels een verbindend perspectief (de “+1”): scenarios.
- *Soni, Nord en Hofmeister* [12]: dit raamwerk is gelijksoortig aan het Kruchten raamwerk, en onderscheid de volgende perspectieven: conceptual architecture, module architecture, execution architecture en code architecture. Een overkoepelend perspectief, zoals het scenario perspectief ontbreekt echter.

De verschillende perspectieven die gekozen kunnen worden brengen idealiter de interesses (concerns) van de verschillende belanghebbenden (waaronder bijvoorbeeld de gebruikers, opdrachtgevers, ontwerpers, ontwikkelaars en beheerder) tot uitdrukking. Bijvoorbeeld in termen van een strategisch-, bedrijfs-, process-, data-, en beheerstandpunt. Diverse van de in Nederland opererende ICT dienstverleners en hun klanten hebben ook hun eigen raamwerken van architectuurperspectieven ontwikkeld.

Er lijkt dus een waar oerwoud te ontstaan met daarin diverse perspectieven en raamwerken van perspectieven om de architectuur van een systeem in kaart te brengen. De auteurs van de IEEE architectuur standaard [1] beseften dit ook, en hebben met het oog hierop drie belangrijke concepten ingevoerd:

1. Een *view* is een concrete invulling van een bepaald perspectief op de architectuur van een systeem.
2. Een *viewpoint* is een specificatie van de conventies voor de constructie en het gebruik van een view. Het een kader voor het maken van views. Dit kader zal doorgaans bestaan uit de doelstellingen en het beoogde publiek van de resulterende view, evenals de benodigde technieken en notatiewijzen voor de creatie en analyse van de view.
3. Een *viewmodel* is een stelsel van viewpoints welke tot doel heeft om vanuit meerdere perspectieven de architectuur van een systeem.

Recentelijk, werd hier door het Software Engineering Institute [3] een verfijning aan toegevoegd, namelijk in de vorm van het *viewtype* concept. Een *viewtype* wordt gedefinieerd als zijnde een specificatie van de soort informatie die door een view geleverd moet worden. Met andere woorden, welke aspecten van een systeem worden door een specifieke view belicht? We kunnen viewtypes dus zien als een abstractie van viewpoints; een abstractie die zich puur richt op de vraag *welke informatie* er over het systeem en haar architectuur wordt aangeboden en niet op de vraag *hoe* (in termen van technieken) deze informatie wordt aangeboden.

Middels het viewtype, viewpoint en viewmodel kan de jungle aan architectuur perspectieven en raamwerken al aardig in kaart gebracht worden. Echter een probleem met de tot op heden ontwikkelde architectuur concepten en raamwerken blijft het ontbreken van handvaten en heuristieken om ze in de praktijk te kunnen implementeren. Doorgaans dient men bijvoorbeeld om een abstract architectuur model te kunnen implementeren, zelf te bepalen wat de relevante doelstellingen (zowel organisationeel als voor de architectuur), interesses (voor verschillende belanghebbenden) en viewpoints zijn. Deze aspecten zijn dus in sterke mate situatie, of context, afhankelijk.

Met andere woorden, wat er volgens ons nodig is, is een "contingency model" dat situationeel bepaalde variabelen beschrijft voor het toepassen van architecturen in specifieke situaties. Het contingency model benadrukt dus de variabiliteit van de pragmatiek van architecturen binnen een specifieke (situationeel bepaalde) context. Door het invullen van deze variabelen naar aanleiding van een bepaalde situatie, zou de architect nu op een consistente en effectieve wijze de benodigde toepassing van een architectuurraamwerk kunnen vaststellen. Contingency modellen worden overigens al enige jaren succesvol toegepast binnen het domein van organisatie-theorie.

Een eerste aanzet tot zo'n situationeel bepaald raamwerk, dat nodig is om het werkbaar te maken in de praktijk, wordt in dit artikel geboden, mede op basis van de viewpoint en viewmodel concepten. We starten met het bespreken van de bouwstenen van zo'n raamwerk in de volgende paragraaf. Daarna zetten we uiteen hoe enkele van deze dimensies praktisch vorm kunnen worden gegeven.

3. Methodisch raamwerk voor het werken met Architectuur

Om meer grip te krijgen op de 'architectuur jungle' stellen we voor om terug te grijpen op een bewezen raamwerk dat is gebruikt om de zogenaamde 'methoden jungle' te ordenen. Aan het einde van de tachtiger jaren van de vorige eeuw ontstond er een groeiende behoefte om verschillende informatiesysteem ontwikkelingsmethoden met elkaar te vergelijken. In [4,5,6] worden verschillende raamwerken en strategien besproken om ontwikkelmethoden onderling te positioneren. In dit artikel zullen we het methodische raamwerk uit [4] en [5] als basis nemen. Dit raamwerk dient als fundament van het situationeel bepaalde raamwerk dat in paragraaf-5 wordt uiteengezet. Het raamwerk onderscheid van origine vijf aspecten van een methode om langs die weg een methode in kaart te brengen (zie rechterkant in Figuur-1). Dit raamwerk is in [7] is uitgebreid met een zesde aspect. In het resulterende raamwerk worden de volgende aspecten van een methode onderscheiden:

- De *denkwijze* richt zich op de aannamen ten aanzien van het soort probleemdomeinen, oplossingen, analysten en ontwerpers.
- De *modelleerwijze* definieert de *concepten* van de taal (of talen) die gebruikt mogen worden om systeembeschrijvingen te denoteren, analyseren, visualiseren of animeren.
- De *communicatiewijze* beschrijft hoe de abstracte concepten van de modelleerwijze worden gecommuniceerd. Bijvoorbeeld in termen van een textuele of een grafische notatie.
- De *werkwijze* **structureert** (delen van) de manier waarop de methode wordt uitgevoerd. Bijvoorbeeld in termen van taken, sub-taken, opeenvolging, etc. Maar ook richtlijnen en heuristieken ten aanzien van de uitvoering van deze taken.
- De *besturingswijze* richt zich op de managerial aspecten van de systeemontwikkeling. Denk hierbij aan zaken zoals humaan resource management, kwaliteit- en voortgangscontroles, evaluatie van plannen, etc.
- De *ondersteuningswijze* heeft betrekking op de (mogelijk geautomatiseerde) hulpmiddelen die gebruikt kunnen worden bij de systeemontwikkeling om het gebruik van de methode te ondersteunen

Als we, denkende in termen van deze 'wijzen', onze blik weer wenden tot de viewtype, viewpoint en viewmodel concepten, dan kunnen we deze nu een heel stuk preciezer definieren:

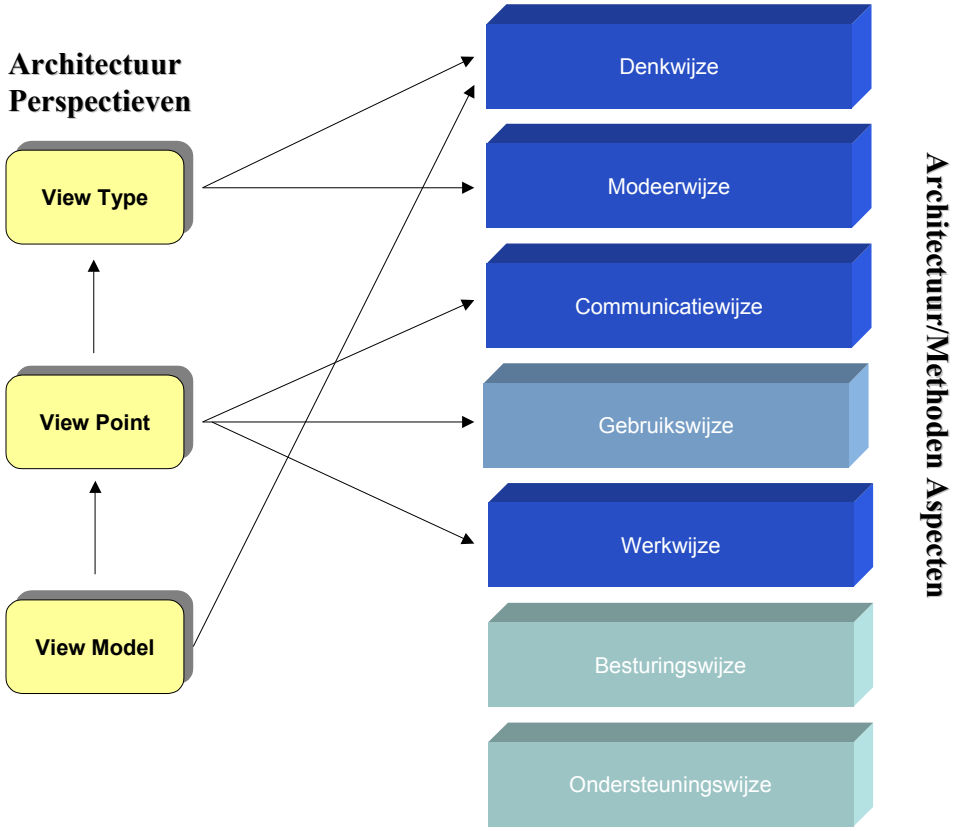
- Een *viewtype* is een combinatie van een denkwijze en een modelleerwijze, en richt zich daarmee op de concepten die vanuit dit perspectief in een systeem, en haar architectuur, worden onderkent.
- Een *viewpoint* is de combinatie van een viewtype, een communicatiewijze, een werkwijze en een gebruikswijze.

De zojuist genoemde *gebruikswijze* is feitelijk een nieuw aspect. Voor viewpoints is het belangrijk om te weten in welke situaties en voor welk doel de viewpoint bruikbaar is, en hoe deze eventueel is af te stemmen op een specifieke situatie. Net zoals een methode, mag een viewpoint *nooit* een keurslijf worden. De *gebruikswijze* is er onder andere voor bedoelt

te voorkomen dat alles er uit ziet als de spreekwoordelijke spijker. Tenslotte kunnen we een viewmodel nu als volgt definieren:

- Een *viewmodel* is een (consistente) combinatie van een denkwijze en een samenstel van viewpoints.

Een viewmodel dient een consistent geheel te zijn. Denk hierbij aan denkwijzen die compatibel moeten zijn met elkaar, concepten die elkaar moeten aanvullen (en zeker niet tegenspreken), compatibiliteit van werkwijzen, etc.



Figuur 1 Structureren van Architectuur-aanpakken met Perspectieven gebaseerd op Methode Aspecten

Middels het bovenstaande raamwerk van viewtypes, viewpoints en viewmodel zouden we, gelijkelijk aan de 'methoden jungle', de 'architectuur jungle' beter in kaart kunnen brengen. Dit alles met in het achterhoofd dat de pragmatiek van alle-dag vraagt om een situationeel-afhankelijke aanpak.

We kunnen nu exacter aangeven van wordt bedoeld met de contingency-aanpak voor architecturen: men moet in staat zijn om gericht op een specifieke context een optimaal **viewmodel** samen te stellen (dat dus is gebaseerd op viewpoints en viewtypes), of in ieder geval bewust een bestaand raamwerk te selecteren.

4. Dimensies van Architectuur

Bij het selecteren en structureren van viewmodels/types/points is het belangrijk te weten voor welk **doel** men gaat architectureren. De volgende aspecten reflecteren doorgaans de doelstelling van het opzetten van een architectuur en tijdens het definieren van situationeel-bepaalde architectuur-aanpak dient hier dus rekening mee te worden gehouden. Deze aspecten vormen normaliter het resultaat van het collectieve definitie-proces tussen de belanghebbenden:

1. *Scope van het systeemdomein*
   De scope van het systeemdomein definieert de aspecten welke wel, en welke niet in beschouwing worden genomen. Grofweg bepaald de scope van het systeemmodel bovendien het aggregatie/abstractie-niveau waarop het systeemmodel wordt beschouwd, bijv. in het geval van een organisatie-systeem, zou op afdelings-, project- or departmentaal niveau kunnen worden gekeken. Het aggregatie-niveau bepaald het oplossend vermogen (de resolutie) van het systeemmodel. Bovendien, dient de precisie van het domein te worden bepaald: dit bepaald immers hoe scherp concepten uit het systeemdomein uit elkaar kunnen worden gehouden (een formeel gedefinieerd wiskundig concept is bijvoorbeeld veel preciezer omschreven dan die van onderzoeker).

2. *Tijdshorizon waarmee we naar het systeemdomein kijken*
   Behalve een afbakening van het systeem-domein, is het ook belangrijk om aan te geven over wat voor termijn een systeemmodel wordt geconstrueerd. In het geval van een lange-termijn model dient rekening te worden gehouden met allerlei dynamische en onbekende (meestal externe) invloeden op het systeemdomein.

3. *Levenscyclus fase van de ontwikkeling van het systeemdomein: requirements, design, implementatie, evolutie*
   Gedurende het ontwikkelproces van een architectuurmodel worden verschillende stadia doorlopen. Doorgaans transformeren de artifacten welke tijdens een dergelijk traject, van analyse tot constructie, worden opgeleverd van erg abstact (niet precies) tot concreet (precies).

4. *Abstractie-niveau waarop we naar het systeemdomein willen kijken*
   Zoals in het bovenstaande reeds aangegeven, speelt het abstratie niveau een doorslaggevende rol bij het vaststellen van de scope vh systeemdomein. Dit aspect is met name belangrijk voor het beteugelen van de complexiteit van architectureren. Door een systeem te beschrijven vanuit verschillende abstractie-niveaus kunnen problemen met betrekking tot de kritische massa van de architectuur worden omzeild.

   Er zijn verschillende abstractie-mechanismen welke kunnen worden gehanteerd, bijv. door gebruikmaking van decompositie, maar ook typering (het groeperen van systeemelementen met het zelfde gedrag en dezelfde karakteristieken), overerving en delegatie-mechanismen kunnen hiervoor gebruikt worden. Een bekende vorm van abstractie is de abstractie van implementatieaspecten, zoals bijvoorbeeld naar voren komt in een functionele of conceptuele kijk op een systeem en een fysieke kijk op een systeem. Tenslotte kunnen ook meer geavanceerde mechanismen worden gebruikt zoals encapsulatie, waarmee een architectuursysteem van buitenaf wordt bekeken en interne complexiteiten verborgen worden gehouden voor de ontwerpers.

5. *Aspecten van het beschouwde systeemdomein: Informatie, Processen, Data, Events, Producten, ...*
   De aard van systeeemdomeinen is afhankelijk van het viewmodel dat wordt gekozen. We kunnen bijvoorbeeld de volgende vier domeinen in de werkelijkheid onderscheiden: informatie-domein, proces-domain, data-domein, fysiek domein.

De bovenstaande lijst van aspecten welke van belang zijn voor het opstellen van de systeemarchitectuur binnen een situationeel bepaalde context, is niet volledig. Zij omvat slechts een eerste aanzet tot een volledige lijst van invloedfactoren die van belang zijn op het architectuur systeemdomein. Zo'n lijst zou onderwerp van toekomstige onderzoeksinspanningen kunnen zijn, welke een sterke voedingsbodem zou moeten hebben in de empirie.

6. Pragmatiek van Architectuur

In het bovenstaande hebben we een eerste aanzet gegeven tot het definiëren van een verzameling van gerelateerde variabelen welke bepalend zijn voor het ontwikkelen van systeem-architecturen binnen een specifieke context. Deze context wordt vormgegeven middels een viewmodel dat is gebaseerd op achtereenvolgens viewpoints en viewtypes. Het resulterende "contingency model" is naar onze mening een eerste goede stap in de richting om meer orde te scheppen in de architecturele jungle. Echter, alvorens dit model in de praktijk kan worden toegepast dienen nog verschillende vragen te worden uitgediept, waaronder:

- Is het mogelijk, en zinvol, om een generieke verzameling viewtypes, viewpoints en view models te definiëren welke kan worden geinstantieerd voor een bepaalde situatie?
- Welke heuristieken (gebruikswijzen!) zijn er te benoemen om eea in de praktijk werkbaar te maken en toe te kunnen snijden op specifieke situaties?
- Het opstellen van een view conform een bepaald viewpoint kost tijd en geld. Daar staat tegenover dat het iets oplevert in termen van inzicht, sturing, etc. Hoe lang blijft die waarde behouden? Wat is de levensduur van zo'n architectuurplaat? Weegt dit op tegen de kosten van het opstellen? Hoe kan men daarop sturen? Bijvoorbeeld: Het Zachman raamwerk bevat 32 specifieke viewpoints. Is het altijd zinnig om alle 32 vakjes in te vullen?
- Kunnen niet-functionele eisen voldoende worden weergegeven in dergelijke modellen? Is het mogelijk om op basis van de niet functionele eigenschappen van een component, de niet-functionele eigenschappen van een architectuur af te leiden? Valt dit proces te formaliseren en te gieten in een raamwerk?
- Biedt zo'n contingency-model een zinvolle ondersteuning voor het omgaan met veranderingen mbt de architectuur?

Genoeg pragmatische vragen voor interessant, doch pragmatisch, promotieonderzoek. Wie durft?

Verwijzingen

[1] IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee. *Recommended Practice for Architectural Description of Software Intensive Systems*, September 2000. ISBN 0738125180

[2] D.B. Rijsenbrij, J. Schekkerman, en H. Hendrickx. Architectuur, besturingsinstrument voor adaptieve organisaties – De rol van architectuur in het besluitvormingsproces en de vormgeving van de informatievoorziening. Lemma, Utrecht, Nederland, 2002. ISBN 9059310934
[3] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, en J. Stafford. Documenting Software Architectures: Views and Beyond. Addison-Wesley, Reading, Massachusetts. ASIN 0201703726
[4] P.S. Seligmann, G.M. Wijers, en H.G. Sol. Analyzing the structure of I.S. methodologies, an alternative approach. In R. Maes, editor, *Proceedings of the First Dutch Conference on Information Systems*, Amersfoort, Nederland, 1989.
[5] G.M. Wijers en H. Heijes. Automated Support of the Modelling Process: A view based on experiments with expert information engineers. In B. Steinholz, A. Sølvberg, and L. Bergman, editors, *Proceedings of the Second Nordic Conference CAiSE'90 on Advanced Information Systems Engineering*, volume 436 of *Lecture Notes in Computer Science*, pages 88-108, Stockholm, Sweden, 1990. Springer-Verlag. ISBN 3540526250
[6] A.H.M. ter Hofstede en Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57-65, January 1992.
[7] H.A. Proper en Th.P. van der Weide. EVORM: A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313-359, 1994.
[8] J.A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.
[9] D. Tapscott en A. Caston. Paradigm Shift - The New Promise of Information Technology. McGraw-Hill, New York, New York, 1993. ASIN 0070628572
[10] ISO, Reference Model for Open Distributed Processing, 1998. ISO/IEC 10746-1:1998
[11] P.B. Kruchten, The 4+1 View Model of Architecture, IEEE Software, 12(6):42-50, 1995.
[12] D. Soni, R.L. Nord, en C. Hofmeister. Software Architecture in Industrial Applications, in R. Jeffrey en D. Notkin, editors, *Proceedings of the 17^{h} International Conference on Software Engineering*, pages 196-207, ACM Press, New York, New York.

De Auteurs

– Willem-Jan van de Heuvel is als Universitair Docent werkzaam binnen het Departement Informatiekkunde aan de Universiteit van Tilburg. Zijn onderzoek richt zich op het ontwikkelen van oplossingen voor integratie-problemen tussen legacy- en nieuwe bedrijfssystemen. Hierbij spelen issues zoals architecturen, semantische ineroperabiliteit, interface (type) conformance, e.d., een grote rol.
– Erik Proper is Hoogleraar Informatiekunde aan de Katholieke Universiteit Nijmegen. Hij is lid van de afdeling Information Retrieval & Information Systems, waarbinnen hij het architectuuronderzoek trekt. Daarnaast is hij eindverantwoordelijke voor de Nijmeegse Informatiekunde opleiding. Tevens is hij medeoprichter van het Nederlandse Architectuurforum.