Expliciet Gestructureerde Informatie Acquisitie Tijdens het Architectuurproces

Reinier l'Abée

15 augustus 2006

Master Software Engineering
Universiteit van Amsterdam

Afstudeerdocent:	Drs. Hans Dekkers
Bedrijfsbegeleiders:	Drs. Rik Farenhorst, Drs. Job Vondeling
Opdrachtgever:	Getronics PinkRoccade
Versie:	1.0
Publicatiestatus:	Openbaar

UNIVERSITEIT VAN AMSTERDAM

Getronics PinkRoccade

Voorwoord

Voor u ligt het resultaat van een onderzoek dat is uitgevoerd in samenwerking met de Universiteit van Amsterdam en Getronics PinkRoccade naar aanleiding van het afstudeerprogramma van de opleiding Software Engineering.

Graag wil ik Thiel Chang en Jelle Gerbrandy, managers afdeling Research & Development, en Dennis Kerssens, manager Serviceline Architectuur, bedanken voor het mogelijk maken en het aanbieden van deze onderzoeksopdracht binnen Getronics PinkRoccade.

Ik wil Job Vondeling en Rik Farenhorst graag bedanken voor het geven van alle feedback tijdens het onderzoek, voor het bewaken van het gehele onderzoeksproces en voor het faciliteren van afspraken met de juiste personen binnen de organisatie van Getronics PinkRoccade.

Ik wil Hans Dekkers bedanken voor alle begeleiding vanuit de universiteit, voor het geven van alle feedback en voor het bewaken van de structuur en het academische niveau van de scriptie.

Ik wil alle mensen waarmee ik binnen Getronics PinkRoccade heb samengewerkt bedanken voor de prettige werksfeer en alle leerervaringen die ik heb opgedaan. In het bijzonder bedank ik:

Marieke Raat, Peter Jasperse, Maarten Boeree, Barend Jan van Eijk, John van de Mortel en Anne Teunissen voor het deelnemen aan interviews, het valideren van bevindingen, het evalueren van de oplossingen en het bieden van informatie over de gemaakte architectuurframeworks en de FrameWorkBench.

Samenvatting

Het bewaren en hergebruiken van architectuurkennis maakt het voor een organisatie mogelijk om sterker te worden door architectuurbeslissingen en oplossingen iteratief te verbeteren. Getronics PinkRoccade (GPR) heeft hiervoor een oplossingen ontwikkeld genaamd de FrameWorkBench (FWB). De tool bewaart alle opgedane kennis en ervaringen van de architecten en maakt deze bruikbaar voor andere projecten. In de huidige situatie heeft deze tool te weinig draagvlak bij de architect. Een van de oorzaken is dat de intakefase van de tool niet goed aansluit op het beslissingsproces en de informatiebehoeftes van de architecten. In dit onderzoek zijn de beslissingsgebieden en informatiebehoeftes achterhaald en is er een verbetervoorstel opgezet voor een verbeterde intakefase in de huidige ondersteuning.

De beslissingen zijn aan de hand van twee architectuurframeworks en vele interviews met de architecten in kaart gebracht. Bij elke beslissing is in kaart gebracht om wat voor soort beslissing het gaat en in welk beslissingsgebied de beslissing valt. Van elk architectuurframework is een conceptueel model met beslissingsgebieden gemaakt dat een weergave vormt van het beslissingsproces van de architect. De twee conceptuele modellen zijn gevalideerd met de betrokken architecten en vervolgens verenigd tot één generiek beslissingsmodel. Op dit model is een analyse gedaan waarbij is ingegaan op de afhankelijkheden tussen de beslissingsgebieden, de optimale volgorde en de beslissingsruimte in de verschillende gebieden. Uit deze analyse zijn een aantal gebieden gekomen waar de architect de meeste moeite mee heeft en waar de ondersteuning het beste toegepast zou kunnen worden.

Een informatiebehoefte ontstaat uit een beslissing die een architect dient te nemen. Voor het achterhalen van de juiste informatiebehoeftes is daarom gebruik gemaakt van het ontwikkelde generieke beslissingsmodel. Samen met de architecten is gekeken welke informatie nodig is om in elk beslissingsgebied tot een keuze te kunnen komen. In de analyse van de informatiebehoeftes is in kaart gebracht aan welke informatie de architect de meeste behoefte aan heeft en welke informatie de meeste winst kan opleveren in het beslissingsproces.

Uit de analyse van de beslissingsgebieden en informatiebehoeftes zijn zes succesfactoren opgesteld die aangeven waaraan een verbeterde intakefase van de ondersteuning moet voldoen om aan te sluiten op het beslissingsproces en de informatiebehoeftes van een architect binnen GPR. Deze succesfactoren zijn verwerkt tot een verbeteradvies.

In het verbeteradvies is een stappenplan voorgesteld dat gebruikt kan worden in de intakefase van de ondersteuning. Dit stappenplan speelt in op de antwoorden van de architect en laat duidelijk zien hoe een architect wordt geholpen bij het maken van beslissingen. De relaties tussen de verschillende stappen zijn afgeleid van de gevonden afhankelijkheden tussen de beslissingsgebieden in het generieke beslissingsmodel en het onderscheid tussen de stappen is kenbaar gemaakt door deze te koppelen aan de gevonden informatiebehoeftes. De onderlinge relaties tussen de objecten in de verbeterde intakefase zijn concreet gemaakt met een datamodel. De werking van het geheel is visueel gemaakt met het bieden van een voorbeeldimplementatie. Uiteindelijk is de oplossing met een fictieve case en een architect doorlopen. Hierbij heeft de architect feedback gegeven op de verschillende stappen en is hij ingegaan op de het gegeven voorbeeld advies wat mogelijk gegeneerd zou kunnen worden om te voldoen aan de juiste informatiebehoeftes.

Inhoudsopgave

1. Introductie

1.1. Het algemene beslissingsproces van een architect

Een architectuur omschrijft een systeem op een bepaald abstractieniveau en maakt het voor alle stakeholders mogelijk om het systeem te begrijpen en om erover te communiceren. Deze omschrijving van het systeem kan op meerdere domeinen betrekking hebben, bijvoorbeeld het bedrijfsproces, de organisatie of de software. In deze scriptie wordt ingegaan op de softwarearchitectuur van systemen. Tijdens dit onderzoek is de beschrijving uit [Jansen 05] gebruikt. Jansen en Bosch hebben zeer veel onderzoek gedaan naar architecturen en omschrijven een softwarearchitectuur als een verzameling beslissingen. Elke architectuurbeslissing kent een aantal eigenschappen:

1. Rationale
2. Ontwerpregels en ontwerpbeperkingen
3. Aanvullende eisen

1. Rationale

De reden en gedachte achter een genomen beslissing. De rationale beschrijft *waarom* een bepaalde keuze is gemaakt. Deze eigenschap van de beslissing gaat vaak verloren als impliciete kennis in de architectuuromschrijving, waardoor het zeer lastig te achterhalen is wat de gevolgen zijn als een beslissing moet worden gewijzigd. [Tyree 05] toont aan dat het bijhouden van rationale een zeer belangrijke rol kan spelen bij het onderhoudbaar maken van de architectuur en het besparen van kosten.

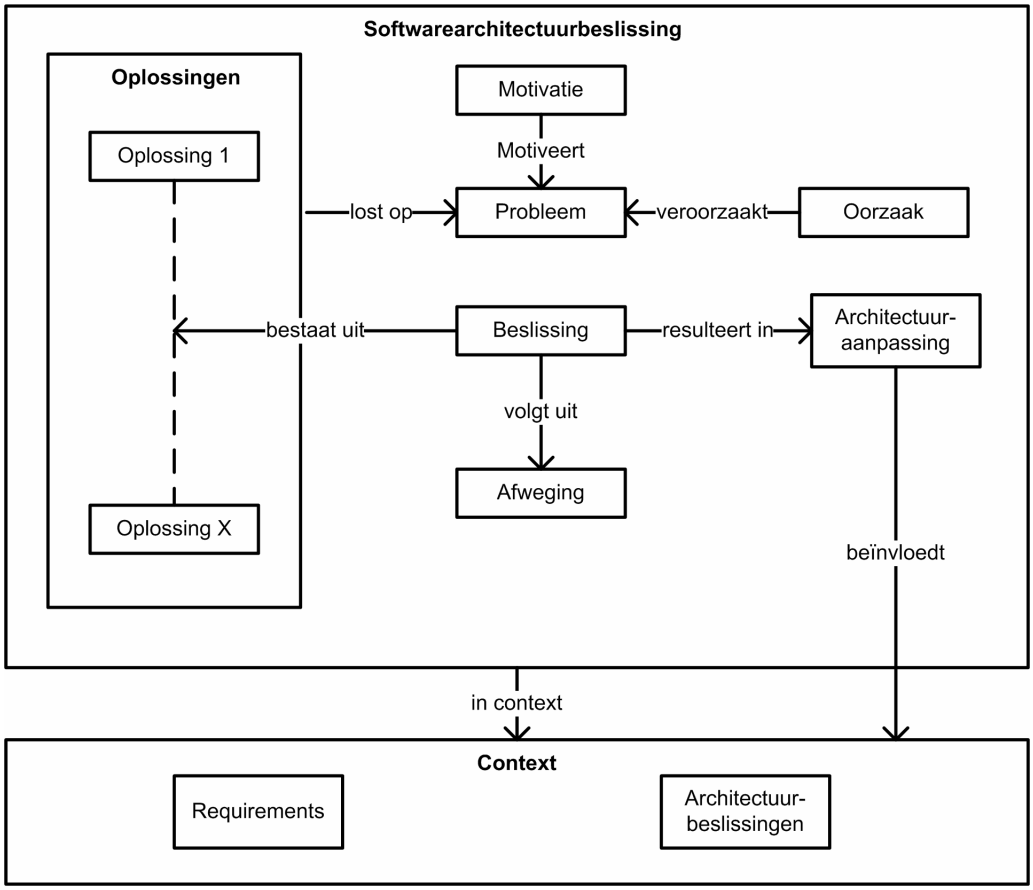
2. Ontwerpregels en ontwerpbeperkingen

Uit een beslissing kunnen ontwerpregels en ontwerpbeperkingen volgen. Ontwerpregels zijn verplichte richtlijnen die gevolgd moeten worden. De ontwerpbeperkingen beschrijven de beperkingen die gelden voor de overige beslissingen. Genomen beslissingen hebben gevolgen voor de overige beslissingen. Het is zeer belangrijk om deze gevolgen te erkennen, omdat architectuurbeslissingen de grote lijnen van het systeem bepalen en niet gemakkelijk te wijzigen zijn. [Kruchten 04] gaat in op de soorten relatie tussen architectuurbeslissingen en laat zien dat beslissingen grote gevolgen kunnen hebben voor de verdere ontwikkeling van de architectuur. Kruchten laat bijvoorbeeld zien dat beslissingen andere beslissingen kunnen uitsluiten of verplichten. De keuze voor een MS .NET ontwikkelomgeving heeft bijvoorbeeld tot gevolg dat er verplicht voor het besturingssysteem Windows moet worden gekozen en het sluit uit dat gebouwde JAVA componenten uit het verleden hergebruikt kunnen worden.

3. Aanvullende eisen

Een beslissing kan tot gevolg hebben dat er nieuwe eisen gedekt moeten worden door de architectuur. Hiermee ontstaan nieuwe beslissingen die genomen moeten worden.

Het resultaat van de genomen architectuurbeslissingen vormen samen de architectuur van het systeem (Figuur 1).



Figuur 1: Algemeen beslissingsproces van een architect [Jansen 05]

Het opstellen van een softwarearchitectuur bestaat uit meerdere stappen. Vaak begint een softwareontwikkelingsproject direct of indirect met een verzameling business eisen. Deze eisen geven aan hoe de te bouwen ICT oplossing een bijdrage kan leveren aan de strategische bedrijfsdoelen van de klant. De realisatie van de ICT oplossing wordt uitgedrukt in functionele eisen en kwaliteitseisen. De business eisen, functionele eisen en kwaliteitseisen vormen samen de begininput voor een architect bij het maken van een architectuur, de architectuur drivers. Als eerst bestudeert de architect de context van het systeem en de knelpunten waarover beslist moet worden. Om deze knelpunten te kunnen bepalen, dient de architect de problemen omtrent het systeem te begrijpen en in te schatten. Bij het nemen van een beslissing over een bepaald knelpunt, maakt een architect een afweging (trade-off) tussen de mogelijke oplossingen en de wensen van de klant. Met de uitkomst van een dergelijke afweging bepaalt de architect bewust of onbewust de grote lijnen voor het te bouwen systeem. Deze grote lijnen bepalen voor een groot deel op welke kwaliteitsattributen van het systeem de nadruk ligt. Een systeem kan namelijk niet op alle vlakken evenveel kwaliteit hebben. In de afweging maakt een architect een keuze tussen de verschillende alternatieven en daarmee tussen verschillende kwaliteitsattributen. Als een architect bijvoorbeeld kiest voor versleuteling in het dataverkeer, dan komt dat de beveiliging sterk ten goede, maar gaat het ten koste van de prestaties omdat er een aantal handelingen op de gegevens moeten worden gedaan voordat er mee gewerkt kan worden. In die zin maakt de architect bij elke beslissingen een afweging tussen de gevolgen en de wensen van de klant. Het is daarom van groot belang dat de architect degelijke afwegingen maakt bij het opzetten van een architectuur. Een kwaliteitsattribuut bestaat uit een niet-functionele eis, zoals bijvoorbeeld beveiliging, prestatie, flexibiliteit, aanpasbaarheid of onderhoudbaarheid. Een complete lijst van kwaliteitsattributen kan gevonden worden in het ISO model 9126 [ISO 91]. [IEEE 92] omschrijft de kwaliteit van een software systeem als de mate waarin de software de gewenste combinatie van kwaliteitsattributen bezit.

1.2. Begrippen / definities

Er is nu in het kort beschreven hoe in dit onderzoek tegen softwarearchitecturen wordt aangekeken. Een aantal begrippen kan echter nog op meerdere manier geïnterpreteerd worden. Om hierin duidelijkheid te krijgen worden de veelvoorkomende begrippen toegelicht met een definitie.

Architectuur
Definitie van IEEE 1471 vertaald uit [IEEE 00]:
"De fundamentele organisatie van een systeem zoals dat tot uitdrukking komt in zijn componenten en hun relaties tot elkaar en de omgeving en de principes die het ontwerp en ontwikkeling bepalen."

Architectuurbeslissing
Definitie vertaald uit [Jansen 05]:
"Een beschrijving van de verzameling architecturale toevoegingen, verwijderingen en wijzigingen aan de softwarearchitectuur, de rationale en de ontwerpregels, ontwerpbeperkingen en bijkomende vereisten die (gedeeltelijk) één of meerdere vereisten in een gegeven architectuur realiseren."

Stakeholder
Definitie van IEEE 1471 vertaald uit [IEEE 00]:
"Een individu, team of organisatie (of klasses daarvan) met belangen in of met betrekking tot een systeem"

Framework
Definitie zoals een framework binnen GPR geinterpreteerd wordt [MArch 05]:
"Een raamwerk waarin wordt vastgelegd op welke wijze een ICT gerelateerde projectopdracht dient te worden uitgevoerd."

Eis
Definitie van IEEE 610.12 vertaald uit [IEEE 90]:
"Een voorwaarde of capaciteit die een gebruiker nodig heeft om een probleem op te lossen of een doel te bereiken."

Referentiearchitectuur
"*Een raamwerk van principes en richtlijnen die samen het beleid van de klant omschrijven op het gebied van werken met architecturen.*"

1.3. Achtergrond en Context

Getronics PinkRoccade (vanaf nu GPR) houdt zich bezig met het ontwikkelen en onderhouden van grote informatiesystemen. De organisatie van GPR is verdeeld in verschillende sectoren. Dit onderzoek is gehouden in de Sector Public, de sector die zich richt op systemen voor (semi)overheid. De Sector Public heeft een eigen architectuurafdeling, architectuurmethode en tools om deze methode te ondersteunen. Voor het ontwikkelen en onderhouden van applicaties gebruikt GPR Sector Public zogenaamde architectuurframeworks, raamwerken, meestal in de vorm van Word documenten, waarin wordt vastgelegd op welke wijze een ICT-gerelateerde projectopdracht dient te worden uitgevoerd. Daarmee garandeert GPR dat dergelijke projectopdrachten worden uitgevoerd volgens van tevoren afgesproken procedures en richtlijnen. Tevens wordt duidelijk van welk middelen (tools, software, hardware, infrastructuur) gebruikt wordt gemaakt. Een architectuurframework kan gezien worden als de architectuurbeschrijving en de richtlijnen die gebruikt worden tijdens het project. Het opzetten van een architectuur en de indeling van het framework wordt ondersteund door de methode MArch en de tool genaamd FrameWorkBench. Hierbij moet worden opgemerkt dat de overige sectoren van GPR zich ook bezig houden met architecturen en dat de werkwijze van deze sectoren niet in het onderzoek zijn meegenomen.

1.3.1. MArch

MArch staat voor **M**ethodische aanpak voor **Arch**itectuur [MArch 05] en beschrijft de visie van GPR over architecturen. MArch beschrijft globaal welke architectuurdomeinen onderscheiden worden en wat voor informatie

elk architectuurdomein moet bevatten. Om de business-ICT alignment te verbeteren en om de verschillende afhankelijkheden in een project kenbaar te maken maakt MArch gebruik van vijf soorten basisarchitecturen; product, proces, organisatie, informatievoorziening en infrastructuur. De volgende toelichting van de verschillende domeinen is gedestilleerd uit de bron [MArch 05].

Business kant

De eerste drie domeinen vallen in de business kant van MArch. De **productarchitectuur** beschrijft de functionaliteit van het product en hoe deze functionaliteit bediend kan worden. De **procesarchitectuur** beschrijft de activiteiten die worden uitgevoerd voor het samenstellen van het product. Per proces wordt aangegeven wat de input en de output is van het proces en welke actoren bij het proces betrokken zijn. De rollen van de actoren vormen een koppeling met de organisatiearchitectuur. Het coördineren van de rollen binnen elk proces wordt beschreven door de **organisatiearchitectuur**. Per rol in elk proces wordt aangegeven welke taken, verantwoordelijkheden en bevoegdheden bij de rol horen. Tevens geeft de organisatiearchitectuur de hiërarchische verhoudingen weer tussen alle rollen.

ICT kant

De laatste twee domeinen hebben betrekking op de ICT kant van MArch. De **informatievoorzieningarchitectuur** ondersteunt de business-kant en beschrijft welke gegevens nodig zijn binnen de product-, proces- en organisatiearchitectuur. Daarnaast beschrijft het de gegevensverwerking die plaatsvindt voor uitvoering en ondersteuning van de activiteiten in de processen en met welke middelen dit gerealiseerd wordt. De informatievoorzieningarchitectuur wordt opgedeeld in drie verschillende perspectieven:

- *Functionele architectuur (gebruikersperspectief)*
  Geeft aan welke clusters aan functionaliteiten onderscheiden kunnen worden en hoe deze met elkaar samenhangen.

- *Softwarearchitectuur (ontwikkelaarperspectief)*
  Deelt de verschillende onderdelen van het informatiesysteem in naar taak.

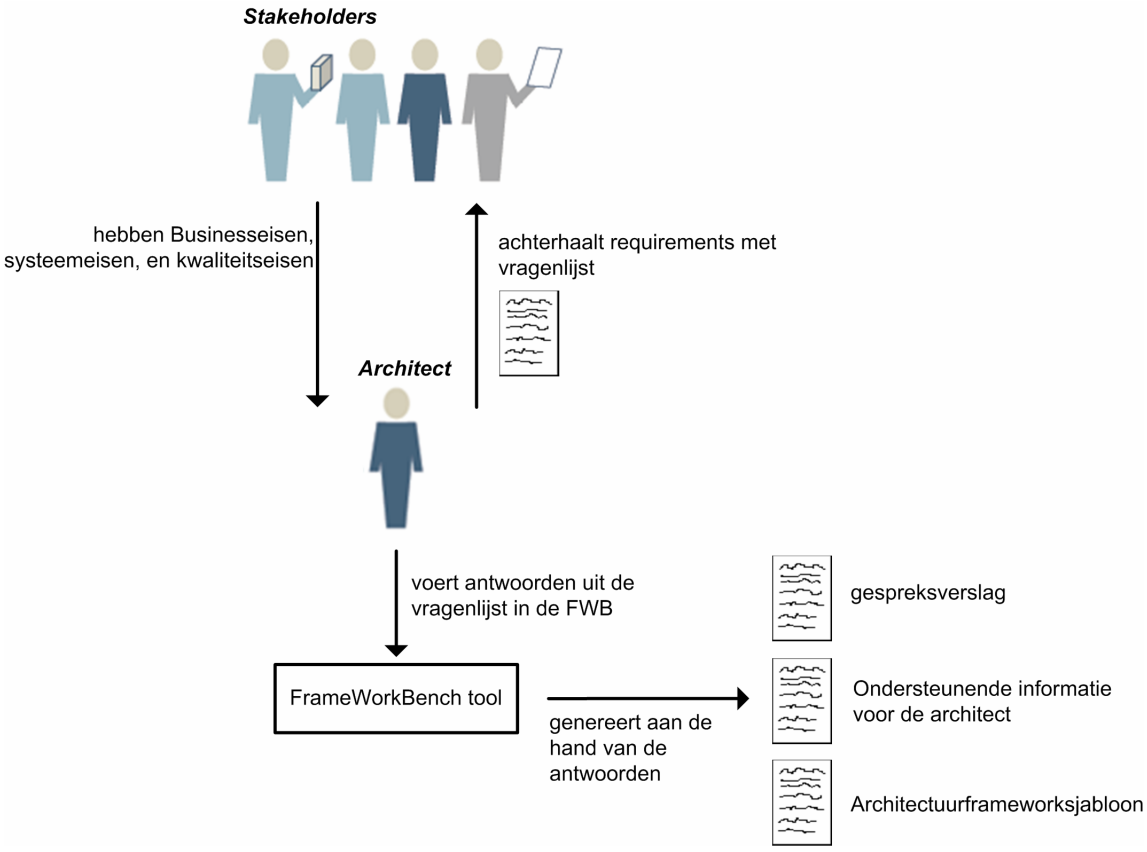
- *Runtimearchitectuur (beheerderperspectief)*
  Onderscheidt de verschillende runtime componenten van het informatiesysteem in de verschillende omgevingen (testomgeving, productieomgeving).

De koppeling tussen de business en de informatievoorziening wordt gevormd door de infrastructuur. De **infrastructuurarchitectuur** beschrijft de middelen waarop de informatievoorziening beschikbaar wordt gesteld. Hierbij kan gedacht worden aan de specificaties van netwerken, clients, servers, printers, systeemsoftware en middleware.

MArch beschrijft geen oplossingen voor een architectuur en kan ook niet vergeleken worden met een referentiearchitectuur, maar omschrijft de visie en begrippen die binnen GPR gehandhaafd worden. Het onderzoek richt zich op de ICT kant van MArch, de informatievoorzieningarchitectuur en infrastructuurarchitectuur. Hiervoor is gekozen omdat de planning het niet toelaat om alle domeinen te onderzoeken en omdat de meest recente frameworks binnen GPR voornamelijk gericht zijn op de ICT kant.

1.3.2. De FrameWorkBench

Om het architectuurbeleid (MArch) en good practices te beheren en eenvoudig en gericht te kunnen toepassen in projecten, is er door GPR een FrameWorkBench (vanaf nu FWB) ontwikkeld. In dit hoofdstuk wordt een ideaalomschrijving gegeven van de rol en het gebruik van de FWB. Figuur 2 geeft een overzicht van de rol van de FWB in de ideaalsituatie. Dit figuur dient ervoor om een gevoel te krijgen bij de rol van de FWB en wordt hier nader toegelicht.



Figuur 2: Rol van de FrameWorkBench

Om architecturen te managen worden ICT-ontwikkelingen onderzocht en wordt de omgeving van het bedrijf en de klanten continu bijgehouden. Samen met nieuwe voorschriften, richtlijnen en oplossingen worden deze gegevens in de FWB vastgelegd. De FWB wordt binnen GPR gebruikt voor het ondersteunen van de architecten bij het opstellen van een architectuur. De intakefase van de tool bestaat uit een lijst met vragen over het project en het te bouwen systeem [Bijlage A]. Deze vragenlijst gaat in op het type project, de organisatie van de klant, de voorkeuren in technologieën, het type functionaliteit, de gewenste kwaliteitseisen en de exploitatie van het project. Bij het gebruik van de FWB wordt deze vragenlijst ingezet als ondersteuning voor de elicitatiefase van de architect. De vragen helpen de architect bij het interactief verkrijgen van informatie voor de architectuur van de diverse stakeholders. Aan de hand van de antwoorden op de vragen genereert de FWB een gespreksverslag, een document met ondersteunende informatie voor de architect en een architectuurframeworksjabloon. Het gespreksverslag kan gezien worden als de notulen van de gesprekken met de stakeholders. Deze notulen bevatten de gegeven antwoorden op de vragen uit de vragenlijst. Het document met ondersteunende informatie voor de architect bevat een aantal richtlijnen voor het gebruik van tools, technologieën, methodieken, good practices en documentverwijzingen. Een aantal voorbeelden van deze richtlijnen zijn:

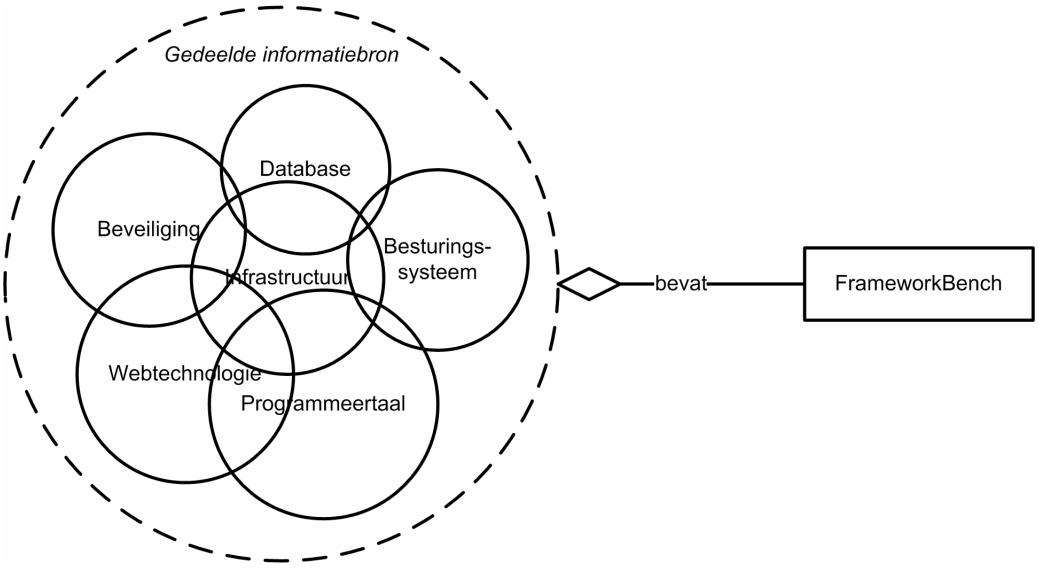
- PowerDesigner mag gebruikt worden voor datamodellering
- Voor alle modellen in de Functionele Architectuur wordt aanbevolen om Bizzdesigner te gebruiken
- De installatie dient te worden opgezet met Visual Studio Installer
- Gebruik HTTPS voor de gevoelige informatie
- Voor de authenticatie kan een combinatie worden gebruikt tussen Basic Authentication en SSL
- Het is niet toegestaan om privacy gevoelige informatie binnen de DMZ op te slaan

De richtlijnen geven voornamelijk de mogelijke alternatieven voor een bepaalde beslissing en de oplossingen die zich in het verleden bij de klanten van GPR hebben bewezen. Het gegenereerde frameworksjabloon dient als basisdocument waar de architect de relevante modellen/schema's, benodigde extra tekst en rationales in kan verwerken. Afhankelijk van de gegeven antwoorden in de vragenlijst bepaalt de FWB welke hoofdstukken wel of niet gegenereert worden. Elk hoofdstuk in het sjabloon is voorzien van een omschrijving en uitleg welke informatie de architect in dat hoofdstuk dient te plaatsen. Het genereren van de ondersteunende informatie en het frameworksjabloon gebeurt door middel van een kennisdatabase. De kennisdatabase bevat de informatie en relaties tussen objecten om de gewenste informatie voor de architect te genereren. De antwoorden op de vragen be-

invloeden de relaties uit de kennisdatabase en daarmee de informatie die gegenereerd wordt. De kennisdatabase wordt up-to-date gehouden met projectevaluaties die door de beheerder van de FWB verwerkt worden in de database (in dit ideaalbeeld).

Het doel van de FrameWorkBench

[Bass 03, p. 6] beschrijft wat voor invloed de architectuur en het systeem op de omgeving hebben. De relatie tussen bedrijfsdoelen, product eisen, ervaring van de architect, de architectuur en de gerelateerde systemen vormen samen een cirkel van feedback. Deze cirkel van feedback moet door de business beheerd worden om opgedane ervaringen en architectuurkennis te kunnen bewaren en te hergebruiken. Een architectuur kan helpen bij het verbeteren van prestaties, omdat de resultaten van een gebouwd systeem op een gestructureerde manier bewaard en hergebruikt kunnen worden. Dit maakt het mogelijk voor een organisatie om te leren van fouten en successen om op die manier sterker te worden. De feedbackcirkel uit [Bass 03, p.11] geeft aan op welke gebieden een organisatie sterker kan worden door de resultaten van een gebouwde architectuur op te vangen. De FWB is een door GPR ontwikkelde oplossing om deze kenniscirkel te ondersteunen en de architectuurfeedback te bewaren en te hergebruiken. Elke architect heeft specifieke expertise in huis en beïnvloedt daarmee de architectuur. De FWB maakt het mogelijk om deze specifieke kennis te delen om zo een gezamenlijke informatiebron aan kennis te creëren. Figuur 3 geeft een beeld van de kennisdeling, dit diagram is niet allesomvattend, maar is puur ter illustratie. Een architect is sterk op zijn/haar eigen expertisegebied en met het delen van expertise kan bereikt worden dat een architectuur diepgang krijgt op alle gebieden, onafhankelijk van de architect die de architectuur maakt. Dit komt overeen met de algemene voordelen van kennisbeheer, namelijk “het versterken van de capaciteiten van de organisatie door beter gebruik te maken van de individuele en collectieve kennisbronnen” [Probst 98], waarbij kennisbronnen kunnen bestaan uit vaardigheden, ervaring, routines, normen en technologieën.



Figuur 3: Kennisdeling

1.4. Analyse probleemgebied

De FWB bewaart architectuurkennis en maakt dit bruikbaar voor volgende projecten. Zonder FWB en ondersteuning kan een architect echter ook tot een architectuur komen, de FWB is een middel om iets te bereiken. Wat wil men bereiken en wat zijn de gevolgen als de FWB niet gebruikt wordt? In dit hoofdstuk wordt antwoord gegeven op deze vraag, om zo het werkelijke probleem en de doelstelling van het onderzoek scherper te krijgen.

Om de richting van het onderzoek en de juiste onderzoeksvragen te kunnen bepalen, zijn tijdens de probleemfase van het onderzoek verschillende stakeholders binnen GPR geïnterviewd. Het doel van deze interviews was om de problemen omtrent het gebruik van de FWB, de frameworks en het nemen van architectuurbeslissingen in kaart te brengen. De volgende stakeholders zijn individueel geïnterviewd: zes architecten, twee ontwerpers, een tester, een beheerder, een projectleider en een projectmanager. De interviews met de architecten waren voornamelijk gericht op het nemen van beslissingen, het gebruik van de FWB, het raadplegen van de verschillende sta-

keholders / informatiebronnen en de problemen waar een architect mee te maken krijgt. Bij de overige stakeholders is voornamelijk ingegaan op het gebruik van het architectuurframework en de beslissingen die nog genomen moeten worden door de stakeholders zelf.

Het nemen van architectuurbeslissingen

Uit de resultaten van de interviews is gebleken dat de architecten moeite hebben met het bepalen of ze alle kritische beslissingen hebben genomen en of ze alle alternatieven hebben overwogen bij elke beslissing. Het gevolg hiervan is dat een technisch ontwerper vaak nog allerlei architectuurbeslissingen moet nemen. Dit is bevestigd door de interviews met verschillende technisch ontwerpers en een bestudering van een technisch ontwerp van één van de projecten binnen GPR. Een architectuurbeslissing is bij de bestudering van het TO gekenmerkt als een systeembrede beslissing met directe gevolgen voor één of meerdere kwaliteitsattributen van het systeem. Uit deze bestudering is gebleken dat meerdere architectuurbeslissingen verspreid staan door het TO heen. Het feit dat een technisch ontwerper architectuurbeslissingen maakt kan weer negatieve gevolgen hebben voor de kwaliteit van het systeem en de tevredenheid van de klant:

- *Het maken van een verantwoorde afweging*
  Een ontwerper heeft geen volledig overzicht van de gevolgen van een beslissing. Welke beslissingen worden bijvoorbeeld nog meer beïnvloed? Doordat er geen rekening wordt gehouden met het totaaloverzicht kan de ontwerper geen verantwoorde afweging maken tussen de verschillende alternatieven en kan er afgeweken worden van de gestelde kwaliteitseisen. Een ontwerper maakt een keuze die op dat moment het beste resultaat oplevert. Terwijl er in de afweging juist rekening moet worden gehouden met de gewenste kwaliteitseisen op lang termijn. De geïnterviewde ontwerpers hebben zelf aangegeven dat bij het maken van een keuze de gevolgen van de alternatieve opties moeilijk voor de klant in kaart gebracht kunnen worden. De ontwerpers maken daarom vaak naar eigen inzicht een technische keuze.

- *Benodigde expertise*
  Uit interviews blijkt dat een ontwerper veel expertise moet opdoen om tot een technische keuze te kunnen komen. Als de ontwerper nog een architectuurbeslissing moet maken dan probeert hij een aantal oplossingen uit in een stuk code om zo te kijken wat het beste werkt. Een architect heeft vaak de expertise, ervaring en ondersteuning om meteen de beste keuze in te zien. Als de klant een keuze moet maken tussen een aantal technische oplossingen, dan kan een architect de gevolgen per keuze beter aan de klant voorleggen dan een technisch ontwerper. Hiermee kan de klant een technische keuze maken die overeenkomt met zijn overweging (budget, kwaliteit en time-to-market).

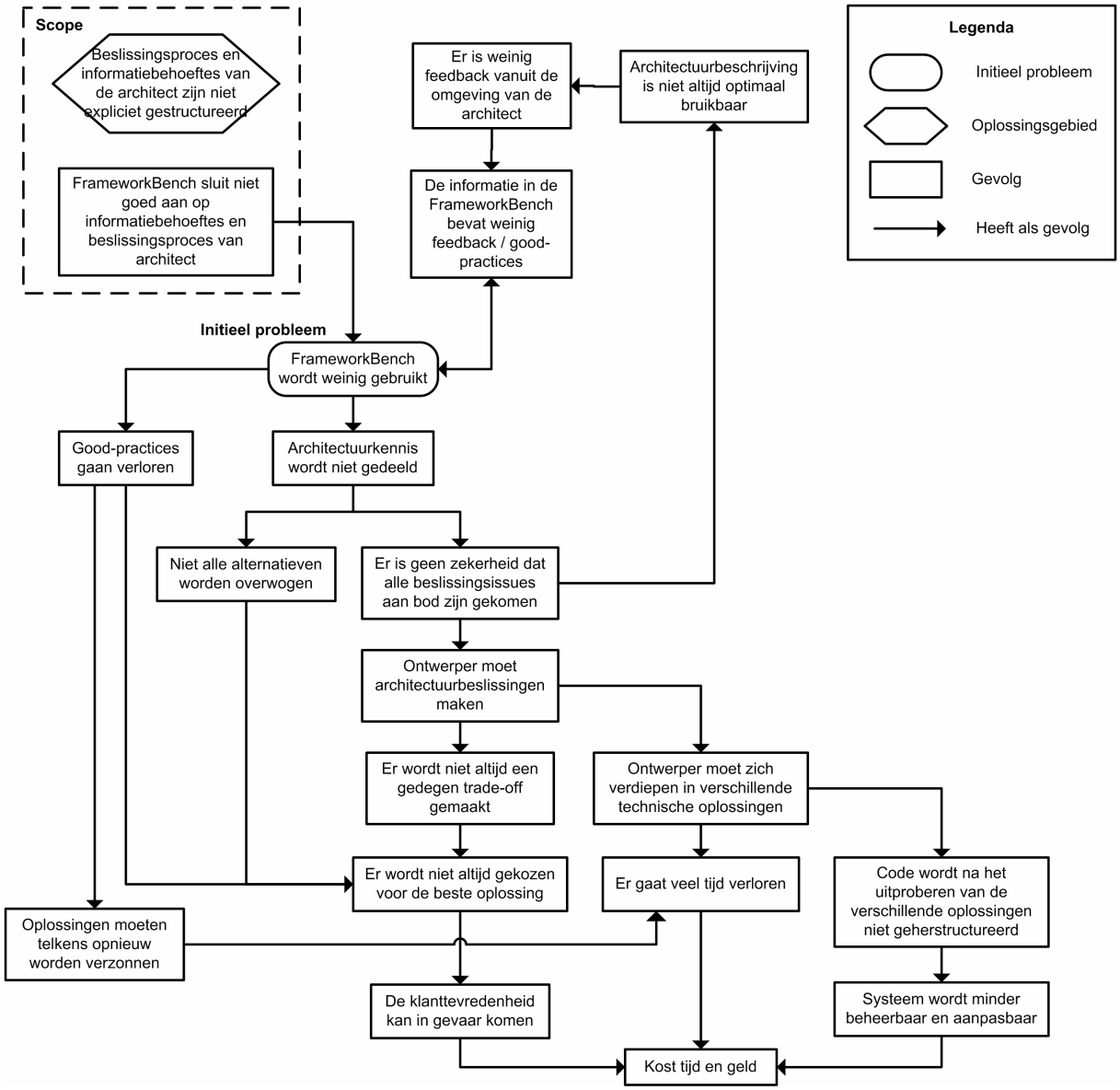
- *Onderhoudbaarheid van de code*
  Omdat de ontwerper nog allerlei complexe keuzes moet maken, wordt de code ook vaak veel complexer en uitgebreider dan nodig is. De ontwerper heeft aangegeven dat de restanten van uitgeprobeerde oplossingen vaak in de code blijven staan, omdat er wegens druk geen tijd is om de code te herstructureren en op te schonen. Als de complexe beslissingen van te voren al worden gemaakt, geeft de ontwerper aan, dan kan er een beter onderhoudbaar systeem gemaakt worden door de code simpel te houden.

Een ander gevolg van het feit dat een architect moeite heeft met het bepalen of alle bekende alternatieven overwogen zijn, is dat er niet altijd gekozen wordt voor de optimale oplossing. Voor GPR heeft dit tot gevolg dat er veel tijd en geld verloren kan gaan, doordat er bijvoorbeeld niet wordt overwogen om standaard componenten te gebruiken of de oplossing te laten aansluiten op de aanwezige expertise. De klant heeft het risico dat de gekozen oplossing niet voldoende kwaliteit biedt.

Het verlies van architectuurkennis

In de interviews is bevestigd dat de FWB zeer weinig gebruikt wordt door de architecten. Dit heeft als gevolg dat de kennis in de kennisdatabase niet wordt bijgevuld met nieuwe good practices en dat de oplossingen van architecten verloren gaan. Verder is er zeer weinig feedback tussen de ontwikkelaars en de architecten. Een architect kan op die manier niet controleren wat er met de architectuur gebeurt tijdens de ontwikkeling van het systeem. De gevolgen van bepaalde keuzes komen niet terug bij de architect en worden daarom niet verwerkt in de kennisdatabase. [Babar 05 b] geeft aan dat ontwerpfouten niet geïdentificeerd kunnen worden als er geen terugkoppeling plaatsvindt tussen het ontwikkelteam en de architect. Hierdoor kunnen beslissingen niet iteratief verbeterd worden en loopt de architect de kans op veelvoorkomende fouten opnieuw te maken.

Figuur 4 laat zien hoe de gevonden problemen met elkaar in verband staan en waarom de problemen werkelijk een probleem vormen voor GPR.



Figuur 4: Probleemgebied

1.4.1. Het gebruik van de intakevragenlijst van de FrameWorkBench

Uit de interviews is gebleken dat de FWB te weinig draagvlak heeft bij de architecten, waardoor de tool te weinig ondersteuning kan bieden en niet voldoende up-to-date blijft wat betreft technologieën en good practices. De architecten geven aan dat de intakevragenlijst van de FWB niet goed aansluit op hun beslissingproces en informatiebehoeftes. Veel vragen worden niet als relevant beschouwd, veel vragen zijn niet te herleiden naar beslissingen en de volgorde van vragen wordt als onlogisch gezien. De FWB kan op meerdere onderdelen verbeterd worden, zoals de intakevragenlijst, gegenereerde output, de rules-engine, de kennisdatabase, de onderhoudbaarheid. De scope van dit onderzoek is gericht op een verbetering van de intakefase van de ondersteuning.

De vragenlijst is bedoeld om de richtlijnen die gegenereerd worden te beïnvloeden, zodat de architect een advies krijgt dat aansluit op de gegeven antwoorden. Om een algemeen beeld te krijgen van de invloed en impact van de vragenlijst zijn twee cases doorlopen waarmee de auteur goed bekend was. De cases, die hieronder beschre-

ven worden, verschillende in belangrijke mate van elkaar. Hierdoor kan op een snelle manier een goed gevoel verkregen worden bij de problematiek en meerwaarde van de vragenlijst. Beide cases zijn gebaseerd op werkelijke projecten, maar niet op projecten binnen GPR.

Case 1:
De E-learning case van het vak Software Requirements op de Universiteit van Amsterdam (UvA). Een sofwarehouse heeft een applicatie ontwikkeld voor een makelaarskantoor. Omdat dit softwarehouse helpdeskkosten wil besparen en het makelaarskantoor beter wil begeleiden bij het gebruiken van de applicatie, wil het een E-learning systeem ontwikkelen. Dit E-learning systeem diende via de browser benaderbaar te zijn.
Specificaties:
Werkt via Internet (browser), doorlooptijd van 12 maanden, 600 functiepunten, werkproces, secundair bedrijfsproces, thuiswerken mogelijk (netwerk), 100 gebruikers, geen eisen voor beschikbaarheid, geen specifieke eisen voor budget.

Case 2:
De restaurant case van het vak Software Requirements op de UvA. Het ontwikkelen van een kassa-systeem voor een restaurant. Dit systeem draaide niet op een browser en draaide, in tegenstelling tot case 1, in het primaire bedrijfsproces. Beschikbaarheid van het systeem was daarom uitermate belangrijk.
Specificaties:
Geen internet, doorlooptijd van 6 maanden, minder dan 300 functiepunten, scope: bedrijfsproces, primair bedrijfsproces, wireless verbinding met laptop, 10 gebruikers, hoge beschikbaarheidseisen, zeer strikte eisen aan het budget.

Beide cases vallen in de het projecttype systeemontwikkeling, maar verschillen verder op alle gebieden uit de vragenlijst. Zeer opvallend was dat de gegenereerde richtlijnen van beide cases zo goed als identiek waren aan elkaar. Het uiteindelijke advies van de FWB wordt dus nauwelijks beïnvloed door de vragenlijst. Dit geeft een architect het gevoel dat de meeste vragen niet relevant zijn en dat het onnodig is om 62 vragen in te moeten vullen.

1.4.2. Conclusies uit de probleemanalyse

Uit de probleemanalyse is gebleken dat het wegvallen van de ondersteuning bij het opzetten van een architectuur kan leiden tot extra kosten in tijd en geld. Als de FWB goed gebruikt wordt, dan kunnen architecturen sneller en met hogere kwaliteit worden opgeleverd en kan de organisatie kan leren van fouten en zich iteratief versterken op het gebied van architectuur. De FWB ondersteunt tenslotte een feedback cirkel voor architectuurkennis. Bij het wegvallen van de FWB in het beslissingsproces van de architect, wordt de feedback cirkel doorbroken en kan de architectuurkennis in de kennisdatabase niet iteratief verbeterd worden. Er is geconstateerd dat de FWB niet voldoende wordt gebruikt en dat deze niet goed aansluit op de belevingswereld van de architecten. Dit laatste is het hoofdprobleem in het onderzoek:

De huidige ondersteuning sluit niet goed aan op het beslissingsproces en informatiebehoeftes van een architect.

Het doel van het onderzoek is dan ook het expliciet maken en structureren van het beslissingsproces en de informatiebehoeftes van de architect, om daarna een brug te slaan naar de huidige ondersteuning. Door deze ondersteuning te verbeteren kan GPR tijd en geld besparen en zal de kwaliteit van het beslissingsproces verbeteren.

1.5. Doelstelling / scoping

Het verbeteren van de ondersteuning kan op meerdere manieren bereikt worden. Omdat de uiteindelijke bevindingen worden doorgevoerd in de intakefase van de FWB (op dit moment een vragenlijst), richt het onderzoek zich op verbeteringen die gerelateerd kunnen worden aan de intakefase. De doelstelling van het onderzoek is daarom gericht op de volgorde van beslissingen en de informatiebehoeftes van een architect tijdens het opzetten van een architectuur. De doelstelling kan opgedeeld worden in drie delen:

1. ***Het bepalen van de optimale volgorde in het nemen van beslissingen***
Het aangeven van de optimale volgorde betekent dat het beslissingsproces expliciet gemaakt moet worden. Er wordt in kaart gebracht wat de afhankelijkheden zijn tussen de verschillende beslissingen en welke beslissingen de meeste impact hebben. Met deze afhankelijkheden kan een architect gemakkelijker bepalen wat de gevolgen zijn per beslissing. Het optimale in de volgorde wordt bereikt door te bepalen hoe de architect in zo min mogelijk stappen het beslissingsproces kan doorlopen. De architect wint hiermee een hoop tijd, waardoor architecturen sneller opgeleverd kunnen worden. Met een snelle oplevering van architecturen kan GPR een voorsprong creëren op de concurrenten. Het expliciet gemaakte beslissingsproces geeft GPR de mogelijkheid om iteratieve verbeteringen toe te passen en om opgedane ervaringen gemakkelijker in het beslissingsproces mee te nemen. [Babar 05 a] geeft aan dat een organisatie alleen kan leren van architectuurfouten en informatie kan hergebruiken als het proces van het opzetten van de architectuur expliciet en gestructureerd is. [Jansen 05] geeft tevens aan dat een architect beter kan werken aan kwaliteitseisen door de architectuur te zien als een expliciete verzameling aan beslissingen. De architect kan een “wat als…” scenario beter analyseren als hij gestructureerd door het beslissingsproces loopt en een expliciete volgorde aanhoudt.

2. ***Het bepalen waar de meeste behoefte is aan informatie en om welke informatie het gaat***
De intakevragenlijst van de FWB gaat in op een aantal architectuuronderwerpen en stuurt daarbij op de informatiebehoeftes van een architect. De architect wil bijvoorbeeld graag weten welke dataopslag het beste gebruikt kan worden en de vragenlijst bevat daarom onderwerpen als vereiste prestaties, intensiteit van het gebruik van het systeem, aantal databasetransacties en omvang gegevensverzameling. Om de ondersteuning en de vragenlijst te verbeteren is het van belang dat er onderzocht wordt waar een architect werkelijk behoefte aan heeft. De eerste doelstelling speelt hierbij een belangrijke rol, omdat een informatiebehoefte ontstaat uit een beslissing die een architect moet nemen. Eerst dient achterhaald te worden welke beslissingen en daarmee kennisonderwerpen aan bod komen bij het opzetten van een architectuur, voordat bepaald kan worden wat voor informatie een architect nodig heeft. Verder kan een architect zelf niet aangeven hoe hij een architectuur opzet en welke informatie/kennis hij gebruikt. Het expliciet maken van de beslissingen die genomen moeten worden en de kennisonderwerpen die hierbij aan bod komen (doelstelling 1) helpen de architect bij het aangeven wat de lastige onderwerpen zijn en aan welke informatie hij werkelijk behoefte heeft.

3. ***Het verbeteren van de koppeling tussen het beslissingsproces en de informatiebehoeftes van een architect en de huidige ondersteuning, zodat de FWB beter ingezet kan worden bij het ondersteunen van de architect.***
Om de bevindingen van doelstelling 1 en 2 bruikbaar te maken in de praktijk wordt er een brug worden geslagen naar de intakefase van de huidige ondersteuning. Doelstelling 1 helpt bij het bepalen van de onderwerpen/volgorde die in de vragenlijst terug moeten komen. Met de optimale volgorde kan een dynamische vragenlijst opgesteld worden, waarbij de architect alleen de vragen moet beantwoorden die nog relevant zijn. Met doelstelling 2 is achterhaald waar de architect precies behoefte aan heeft en waar de vragen op in moeten gaan om de juiste informatie te genereren voor de architect.

1.6. Onderzoeksvragen

Naar aanleiding van de hiervoor beschreven doelstelling kunnen de volgende onderzoeksvragen opgesteld worden:

1e Hoofdvraag *(Doelstelling 1)*
Hoe kan in zo min mogelijk stappen zo veel mogelijk beslist worden in de architecturen informatievoorziening en infrastructuur?

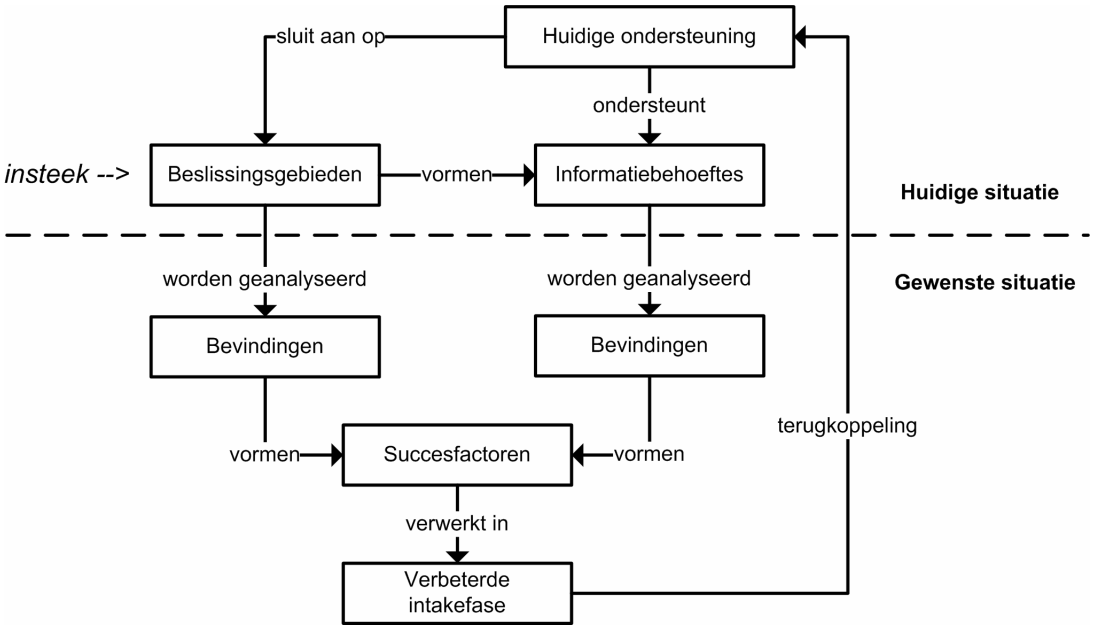
Subvragen
1.1 Welke beslissingen worden genomen in de informatievoorzieningarchitectuur en de infrastructuurarchitectuur?
1.3 In welke groepering/volgorde zijn deze beslissingen te verdelen?
1.2 Welke afhankelijkheden zijn te vinden tussen de beslissingen?
1.4 Welke beslissingen hebben de meeste impact en moeten als eerste genomen worden?

2e Hoofdvraag *(Doelstelling 2)*
In welke beslissingsgebieden kan met welke informatieondersteuning de meeste winst behaald worden?

Subvragen
2.1 Op basis van welke informatie en kennis maakt de architect elke beslissing?
2.2 Welke informatiebronnen zijn aanwezig?
2.3 Wat zijn de informatiebehoeftes van een architect?

1.7. Onderzoeksaanpak

De doelstellingen van het onderzoek zijn op te delen in drie fases. In deze onderzoeksaanpak komt daar een vierde fase bij waarin het verbetervoorstel gevalideerd worden. De onderzoeksaanpak bestaat daarmee uit vier fases. Elk van deze fases is nodig om de volgende fase uit te kunnen voeren. Een ondersteuning in het beslissingsproces van een architect moet aansluiten op de beslissingen die een architect moet nemen en de informatiebehoeftes die hij daarbij heeft. Het onderzoek richt zich daarom op het achterhalen van de te nemen architectuurbeslissingen in een project binnen GPR en vervolgens de informatie die nodig is om bij elke beslissing tot een keuze te kunnen komen. Figuur 5 laat zien hoe de verschillende fases uit het plan van aanpak in relatie tot elkaar staan. Het figuur heeft betrekking op een ideaalsituatie. Een ondersteuning moet voorzien in de informatiebehoeftes van een architect en sluit als het goed is aan op de beslissingen die een architect moet nemen. In de huidige situatie is gebleken dat dit niet het geval is. Er zal daarom als eerst onderzoek worden gedaan naar de beslissingen die een architect moet nemen in het architectuurproces. **Waar** kan een architect in ondersteund worden? Als dat onderzocht is, kan er achterhaald worden **hoe** een architect ondersteund kan worden. Aan welke informatie heeft de architect behoefte? Deze beslissingen en informatiebehoeftes zullen geanalyseerd worden aan de hand van de hiervoor genoemde onderzoeksvragen. Met de resultaten van deze analyse worden een aantal succesfactoren opgesteld, die aangeven waaraan voldaan moet worden om de verbeterde ondersteuning te laten aansluiten op het beslissingsproces en de informatiebehoeftes van de architect.



Figuur 5: Onderzoeksaanpak

Fase 1: Het achterhalen van de verschillende beslissingen die op dit moment genomen worden
Een architect kan zelf niet altijd aangeven hoe een architectuur tot stand komt en waar de lastige punten zitten in het nemen van beslissingen. In deze fase wordt onderzocht welke beslissingen genomen worden bij het opzetten van een architectuur. Deze fase kan gekoppeld worden aan de eerste doelstelling van het onderzoek.

Fase 2: Het achterhalen van de verschillende informatiebehoeftes van de architect
Na het in kaart brengen van de verschillende beslissingsgebieden wordt achterhaald welke informatie nodig is om in elke beslissingsgebied tot een keuze te kunnen komen. Dit wordt voornamelijk gerealiseerd via interviews en het analyseren van de aanwezige informatiebronnen. Hiermee worden de informatiebehoeftes van de architect in kaart gebracht. De informatiebehoeftes worden geanalyseerd om te onderzoeken wat de verschillende soorten informatiebehoeftes zijn en met welke informatie de meeste winst (tijd, geld, kwaliteit) behaald wordt.

Fase 3: Een brug slaan naar de huidige ondersteuning
Met het model van de beslissingsgebieden en de optimale volgorde en het overzicht van de belangrijkste informatiebehoeftes van een architect, wordt een brug geslagen naar de huidige ondersteuning. Er wordt door middel van de literatuur en de gevonden bevindingen gekeken hoe de FWB meer ondersteuning kan bieden bij het opzetten van een architectuur. In de literatuur wordt gekeken welke andere methodes van ondersteuning er zijn en of deze methodes overeenkomen met de bevindingen van het onderzoek.

Fase 4: Validatie
Als laatst zal het verbetervoorstel worden gevalideerd doormiddel van een voorbeeldimplementatie en een evaluatiecase. In deze validatie worden de bevindingen uit het onderzoek getoetst en meetbaar gemaakt.

1.8. Structuur van de scriptie

Hoofdstuk 2 geeft antwoord op de eerste onderzoeksvraag. De beslissingen uit de twee onderzochte projecten worden in dit hoofdstuk gepresenteerd en geanalyseerd en de afhankelijkheden worden verwerkt tot een generiek beslissingsmodel met een optimale volgorde.
Hoofdstuk 1.1 geeft antwoord op de tweede onderzoeksvraag. De verschillende informatiebehoeftes worden gepresenteerd en met een analyse worden de belangrijkste informatiebehoeftes gepresenteerd.
Hoofdstuk 4 slaat een brug naar de FWB en beschrijft een advies om tot een betere ondersteuning voor de architect te komen. In dit hoofdstuk wordt ook de evaluatiecase besproken ter validatie.
Hoofdstuk 5 beschrijft de verschillende contributies voor GPR en de contributies op academisch gebied.
Hoofdstuk 6 beschrijft een evaluatie van het gehele onderzoek.

2. Beslissingsproces van de architect

2.1. Inleiding

In dit hoofdstuk wordt het beslissingsproces expliciet gemaakt en wordt er een antwoord worden gevonden op de eerste onderzoeksvraag:

> *Hoe kan in zo min mogelijk stappen zo veel mogelijk beslist worden in de architecturen informatievoorziening en infrastructuur?*

Bij het opzetten van een architectuur denkt de architect na over een aantal knelpunten waarover beslist moet worden, zogenaamde beslissingsgebieden. Deze beslissingsgebieden worden in dit hoofdstuk in kaart gebracht, omdat dit de gebieden zijn waar een architect in ondersteund kan worden. [Kruchten 04] heeft onderzoek gedaan naar de eigenschappen van beslissingen die een architect maakt bij het opzetten van een architectuur. Hierbij geeft hij aan dat de categorieën/beslissingsgebieden waar een beslissing in kan vallen niet in de literatuur beschreven, maar sterk afhankelijk is van de rol van de architect en de werkwijze van de organisatie.

[Babar 05 b] geeft een aantal manieren om beslissingsgebieden te achterhalen. In dit onderzoek is gebruik gemaakt van individuele interviews, protocolanalyse en bestudering van bestaande architecturen. In dit onderzoek is een protocolanalyse gedaan aan de hand van de methode MArch [MArch 05] die gehanteerd wordt bij het opzetten van een architectuur. Deze methode beschrijft de werkwijze van een architect en de globale indeling voor de architectuur binnen GPR. Voor de bestudering van bestaande architecturen is gekozen voor de architectuurframeworks van twee recente representatieve projecten binnen GPR, de projecten ODZ en BeFlex.
Er is gekozen voor twee projecten, omdat de planning niet meer toe laat en omdat het resultaat van de bestudering gebruikt wordt als spiegel voor de architecten. Hierdoor kan een architect aangeven of de gevonden beslissingen afwijken met de overige projecten binnen GPR. Per framework wordt een lijst met genomen beslissingen opgesteld en per beslissing wordt aangegeven waar de beslissing betrekking op heeft en om wat voor soort beslissing het gaat. Met deze informatie worden de beslissingen ondergebracht in zogenaamde beslissingsgebieden. Deze beslissingsgebieden en de afhankelijkheden tussen de gebieden worden weergegeven in een conceptueel model. Het maken van een conceptueel model heeft een aantal doelen:

1. Een architect kan aan de hand van een lange lijst met beslissingen niet aangeven of de lijst compleet is. Een conceptueel model met beslissingsgebieden is voor een architect gemakkelijker te valideren.
2. De beslissingen in de twee verschillende projecten kunnen beter met elkaar vergeleken worden als ze naar een conceptueel niveau worden getild. Een lijst met beslissingen is erg projectspecifiek, maar elke beslissing kan wel in een bepaalde categorie worden ingedeeld.
3. Een conceptueel model kan gebruikt worden om één generiek beslissingsmodel te maken, dat overeenkomt met alle softwareontwikkelingsprojecten binnen GPR.
4. De informatiebehoeftes kunnen gemakkelijker achterhaald worden door te kijken naar beslissingsgebieden in plaats van losse projectafhankelijke beslissingen.

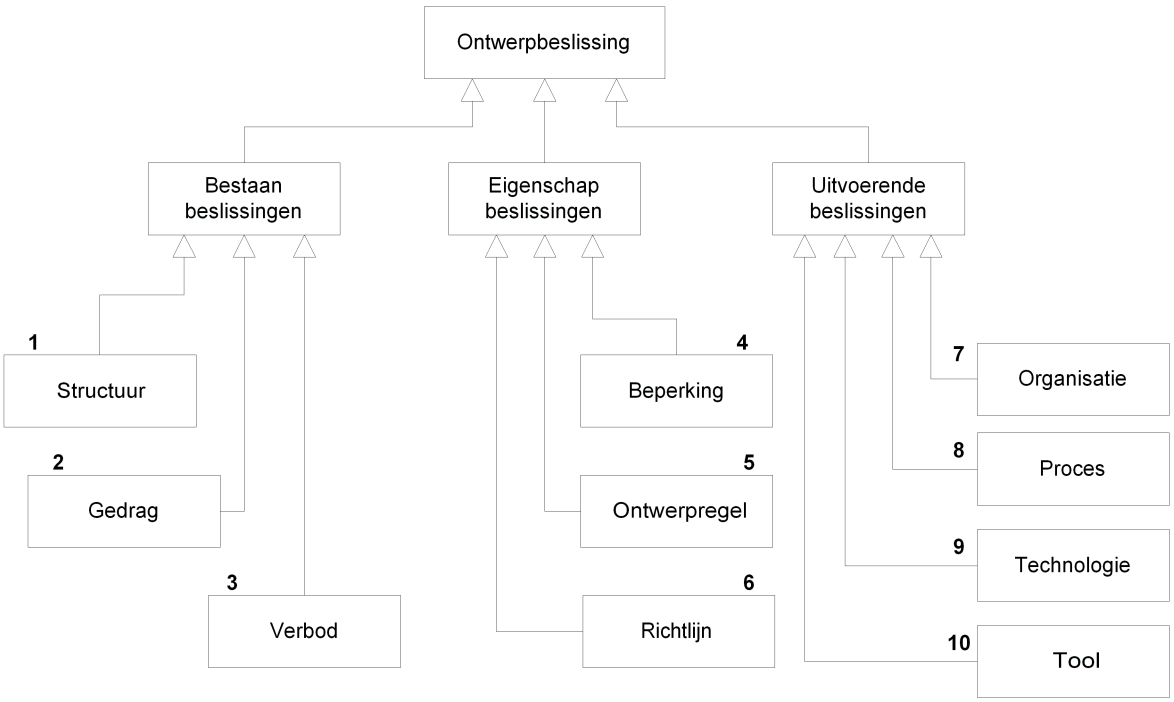
Bij elk van de beide projecten wordt de betrokken architect gebruikt voor de validatie. Voor de keuze van de twee projecten is gekozen voor de meest recente projecten binnen GPR. Het grote voordeel hiervan is dat de betrokken architecten het beslissingsproces in het project nog goed kunnen herinneren, waardoor de validatie van de bevindingen betrouwbaarder verloopt dat een project van een paar jaar geleden. De beslissingsmodellen van beide projecten worden verenigd in één algemeen beslissingsmodel. Aan de hand van de gevonden en gevalideerde afhankelijkheden tussen de beslissingsgebieden, kan een optimale volgorde in het nemen van beslissingen bepaald worden. Deze volgorde wordt verwerkt in het generieke model dat vervolgens wordt gevalideerd met een derde architect die kan aangeven of het model overeenkomt met het beslissingsproces van de door hem uitgevoerde projecten in het verleden.

Ook wordt er gekeken naar wat voor **soort** beslissingen genomen er worden door de architect. Met de resultaten hiervan kan geanalyseerd worden of de huidige ondersteuning is gericht op de juiste beslissingssoorten en waar de verbeterde ondersteuning op gericht moet zijn. [Kruchten 04] geeft een overzicht van de soorten beslissingen

die kunnen voorkomen. Hij geeft hierbij aan dat er veel onderzoek is gedaan naar de gevolgen van beslissingen, maar niet naar de eigenschappen van de beslissingen zelf. In dit onderzoek is daarom gekozen om zijn onderzoek te gebruiken als referentiemodel voor het soort beslissing (Figuur 6). Elk soort beslissing is genummerd en komt terug in de lijst met gevonden beslissingen per framework.

Beslissingssoort	ID	Omschrijving
Structuur	1	Bepaalt de verdeling van subsystemen, modules, lagen en componenten
Gedrag	2	Bepaalt de interactie tussen elementen, connectoren
Verbod	3	Beslissingen die dingen verbieden. Bijv. het systeem gebruikt geen MySQL als database.
Beperking	4	Een beperking voor het systeem. Bijv. er mogen geen open-source componenten gebruikt worden.
Ontwerpregel	5	Een regel waaraan voldaan moet worden. Bijv. alle domeinspecifieke classes moeten in één laag.
Richtlijn	6	Een richtlijn die de architect helpt bij het ontwerpen. De methode RUP bevat bijvoorbeeld een aantal richtlijnen voor de architectuur.
Organisatie	7	Beslissingen met impact op de organisatie. Bijv. alle componenten zullen intern ontwikkeld worden.
Proces	8	Beslissingen met impact op de te gebruiken methodiek. Bijv. we gaan DSDM gebruiken als methodiek. Of: alle veranderingen in de technische interface dienen door de Change Control Board goedgekeurd te worden.
Technologie	9	Belissingen met impact op de technische invulling van het systeem. Bijv. we gaan J2EE gebruiken voor de ontwikkeling.
Tool	10	De hulpmiddelen die bij de ontwikkeling gebruikt zullen worden. Bijv. de applicatie zal ontwikkeld worden in JBuilder 9.0

Tabel 1: Omschrijving van de verschillende soorten beslissingen [Kruchten 04]



Figuur 6: Indeling van de verschillende soorten beslissingen [Kruchten 04]

De beslissingen zijn in drie hoofdcategorieën verdeeld:

Bestaan beslissingen hebben betrekking op de samenstelling van componenten, subsystemen, het gedrag van componenten en de interactie binnen het systeem. *Eigenschap beslissingen* omschrijven de systeemkenmerken en hebben direct invloed hebben op de kwaliteit van het systeem. *Uitvoerende beslissingen* zijn niet direct te koppelen aan systeemcomponenten of kwaliteitsattributen, maar ze worden genomen vanuit de businessomgeving. Ze omschrijven hoe het systeem gebouwd gaat worden.

2.2. Beslissingen uit de projecten ODZ en BeFlex

Project ODZ

Introductie
Het project ODZ is gestart door de grootste klant van GPR, een zelfstandig besturingsorgaan dat in deze scriptie de naam ZBO zal dragen. ZBO is enkele jaren geleden ontstaan uit een fusie van vijf uitvoeringsorganisaties. Iedere uitvoeringsorganisatie beschikt over een eigen basisregistratiesysteem waarin de gegevens over werkgevers, werknemers en hun dienstverbanden zijn opgeslagen. ZBO voert onder meer sociale verzekeringswetten uit en streeft naar een volledige fusie waarbij de informatiesystemen volledig geïntegreerd zullen zijn. De fusie heeft het noodzakelijk gemaakt om van één systeem gebruik te gaan maken. Het systeem ODZ is het informatiesysteem dat moet zorgen voor het ontsluiten van informatie, die nu in de basisregistratiesystemen van de verschillende uitvoeringsorganisaties is opgeslagen.

Beslissingsmodel
Het architectuurframework van het project ODZ is verdeeld in de domeinen functionele architectuur, softwarearchitectuur, runtimearchitectuur en infrastructuurarchitectuur. Elke van deze domeinen is geanalyseerd op beslissingen. Deze beslissingen komen voor in de omschrijvende tekst, zoals:

4. Er mogen geen aanpassingen worden gedaan op de basisregistraties
6. ODZ moet benaderbaar zijn via webservices
22. Het inlezen van gegevens gebeurt via tijdelijke tabellen

En er zijn beslissingen gevonden in de architectuurmodellen, zoals:

25. Mutatiesignalen worden doorgegeven aan de abonnementservice
29. De gegevensbenaderingen verloopt via ODP.NET en SQL
39. De webserver communiceert via http met de client

De totale lijst met gevonden beslissingen kan gevonden worden in bijlage B. Per beslissing is aan de hand van Figuur 6 aangegeven om wat voor soort beslissing het gaat. Om uiteindelijk tot beslissingsgebieden te komen, is elke beslissing ingedeeld in een globaal beslissingsgebied. Omdat [Kruchten 04] aan heeft gegeven dat een lijst met beslissingsgebieden niet in de literatuur omschreven is en sterk afhankelijk is van de organisatie en de rol van de architect, is dit gedaan aan hand van eigen constateringen, de context van de beslissing en validatie met architecten. Alle gevonden beslissingsgebieden zijn gedestilleerd uit de lijst met beslissingen (bijlage B) en verwerkt tot een conceptueel model (Bijlage C). De afhankelijkheden in dit model, tussen de verschillende beslissingsgebieden, zijn naar eigen ervaring gelegd en vervolgens gevalideerd met de betrokken architect van het project. Het domein waaronder de verschillende beslissingen vallen is afgeleid van de plek waar ze in het architectuurframework gevonden zijn. Deze mapping tussen domein en beslissingsgebied kan daarom afwijken van de ideaalsituatie in GPR die omschreven wordt door de architectuuraanpak MArch.

Project BeFlex

Introductie
BeFlex is een applicatie welke de loongegevens van de uitzendbureaus bewerkt en archiveert. De loongegevens van uitzendkrachten worden per week aangeleverd, omdat werknemers per week loon ontvangen. Deze gegevens worden gebruikt door andere belanghebbende binnen en buiten het ZBO. Integratie speelt in dit project dus een belangrijke rol.

Beslissingsmodel
De manier waarop het project BeFlex is geanalyseerd komt overeen met het hiervoor beschreven project, ODZ.
Bijlage D toont de lijst met beslissingen die gevonden zijn in het architectuurframework van het project BeFlex. Ook voor dit project is een model opgesteld met de verschillende beslissingsgebieden (Bijlage E). Dit model is gevalideerd met de betrokken architect.
Veel beslissingsgebieden uit project BeFlex komen overeen met de gebieden uit project ODZ. Voor een omschrijving van deze gebieden wordt daarom ook verwezen naar het voorgaande hoofdstuk. Het verschil in de projecten zit in de invulling van de verschillende gebieden. In dit hoofdstuk draait het niet om de invulling van de gevonden gebieden en daarom wordt daar niet verder op ingegaan. In vergelijking met het project ODZ uit hoofdstuk 2.2 zijn er naast de overeenkomsten een aantal verschillen te achterhalen.

§ *Hergebruik van componenten*
In het project BeFlex is meer expliciete informatie te vinden over het hergebruik van componenten. Op dit moment zijn bestaande componenten moeilijk te verwerken in de architectuur, omdat de architect geen feedback krijgt van het ontwikkelteam. De architect is daarom ook niet op de hoogte van veel gebruikte kant-en-klare componenten. Het hergebruik van componenten hangt sterk af van de applicatieverwerkingsomgeving en het is daarom nuttig om deze beslissingen expliciet te overwegen in elke architectuur.

§ *Gebruik van bestaande softwarepakketten*
In BeFlex is expliciet nagedacht over het gebruik van bestaande softwarepakketten voor bepaalde functionaliteit. Deze beslissing beïnvloedt de technische interface en communicatie tussen componenten en subsystemen.

2.3. Analyse en bevindingen

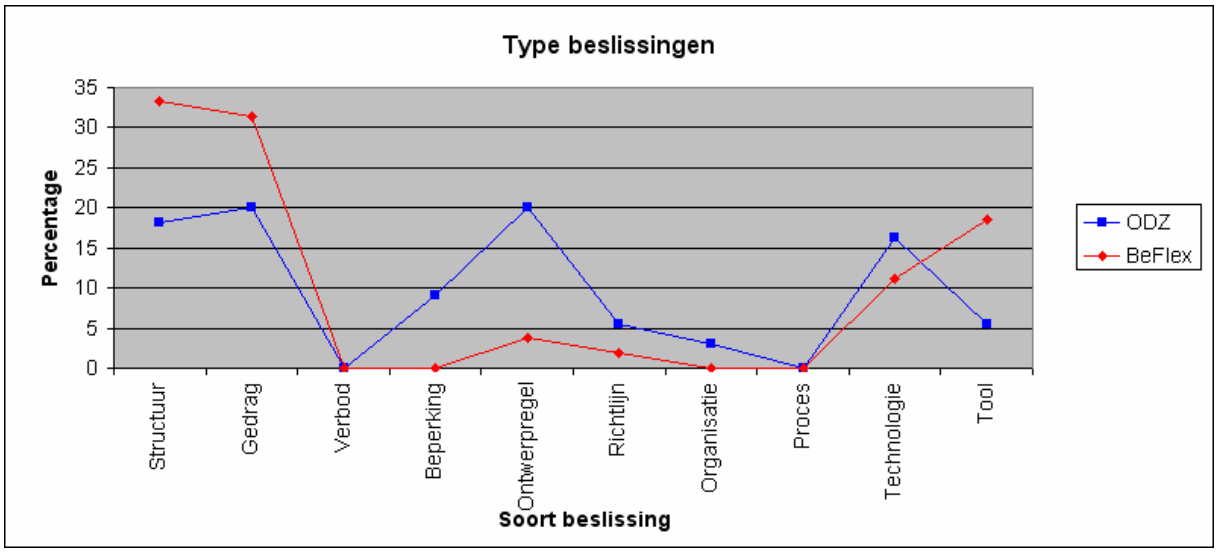
Volgorde in beslissingen
Bij elke beslissing is aangegeven in welk architectuurdomein (functioneel, software, runtime, infrastructuur) de beslissing gevonden is. Het valt op dat sommige beslissingsgebieden in beide projecten ergens anders gevonden zijn. Er wordt niet expliciet rekening gehouden met een volgorde van beslissingen, waardoor er een kans is dat beslissingen inefficiënt worden genomen, inconsistent zijn of waardoor de weergave ervan (de modellen/omschrijving in het framework) niet éénduidig is. Een voorbeeld van inefficiëntie is dat de architect moet nadenken over de webtechnologie of de manier van gegevensbenadering, terwijl deze beslissingen voor een groot deel bepaald worden door de applicatieverwerkingsomgeving. Inconsistentie vindt plaats als een architect eerst kiest voor de webtechnologie JavaServerPages en vervolgens voor een MS .NET applicatieverwerkingsomgeving. Deze twee beslissingen zijn niet compatibel met elkaar, wat leidt tot een inconsistente architectuur. Door aan de architect kenbaar te maken welke beslissingen de meeste impact hebben, kan hij daar rekening mee houden als de volgorde in beslissingen nog open ligt. Want hierbij moet namelijk worden opgemerkt dat de volgorde van beslissingen sterk afhankelijk is van de eisen en wensen van de klant. Er kan geen vaste volgorde worden aangehouden die in elk project ingezet kan worden. Als de klant C# eist, dan zal de architect als eerst het beslissingsgebied *programmeertaal* doorlopen. Ook al kan er in elk project niet dezelfde volgorde worden aangehouden kunnen de beslissingengebieden wel gerangschikt worden op de hoeveelheid impact die ze hebben door te kijken naar de afhankelijkheden in de beslissingsmodellen uit bijlage C en E :

Het beleid van de klant geeft aan waar een architectuur aan dient te voldoen. Dit beleid kan ingaan op allerlei beslissingsgebieden in de architectuur en wordt daarom als eerst in overweging genomen door de architect. Na het bepalen van opgelegde beperkingen in de architectuur aan de hand van het beleid van de klant, begint een architect met de **functionele verdeling van het systeem (1)**, waarna hij verder gaat met de verdeling van modules. Deze verdeling van modules wordt beïnvloed door het gebruik van standaard componenten en softwarepakketten. Het gebruik van standaard componenten en softwarepakketten wordt weer beïnvloed door de applicatieverwerkingsomgeving, het type database en de manier van gegevensbenadering (technische invullingen). De **combinatie van technologieën (2)** moet daarom als tweede beslist worden. Als de technische invulling van het systeem bepaald is wil de architect weten welke **bestaande componenten (3) en softwarepakketten (4)** ingezet kunnen worden, omdat deze twee beslissingsgebieden invloed hebben op de verdeling van modules. Nadat de architect heeft bepaald of hij bestaande componenten of softwarepakketten gaat gebruiken, begint hij met de **verdeling van modules (5)**. Uit de verdeling van modules volgt de **verdeling van componenten (6)** en als laatst de keuze voor **hardware (7)**.

Ruimte in het nemen van beslissingen
Aan de afhankelijkheden binnen beide beslissingsmodellen is te zien dat er veel samenhang is tussen de verschillende beslissingsmodellen. Binnen GPR wordt in de meeste projecten gekozen voor een MS .NET verwerkingsomgeving, wat inhoudt dat een architect zeer weinig ruimte overhoudt in het kiezen van de technologieën. Het beslissingsmodel geeft aan dat de verwerkingsomgeving bepalend is voor de webtechnologie, het type database, gegevensnadering, hergebruik van componenten, gebruik van bestaande softwarepakketten, besturingssysteem, programmeertaal, ontwikkeltool en het gebruik van systeemsoftware. Deze beslissingsgebieden zijn daarom minder interessant om te onderzoeken. De architecten hebben aangegeven dat de **verdeling van modules** en de **communicatie tussen modules** de lastigste beslissingsgebieden zijn. In deze gebieden is de meeste ruimte voor alternatieven en is de architect de meeste tijd kwijt om tot een beslissing te komen. Als een architect in deze beslissingsgebieden betere ondersteuning krijgt, dan levert dat al snel veel tijdswinst op. In hoofdstuk 4 (verbeterde ondersteuning) zal deze analyse worden meegenomen.

Het soort beslissingen

Zoals beschreven is voor beide projecten per beslissing aangegeven om wat voor soort beslissing het gaat. Deze soorten zijn gedestilleerd uit de twee lijsten met gevonden beslissingen (bijlage B en C) van de projecten ODZ en BeFlex. Figuur 7 toont de percentages van de verschillende soorten beslissingen en geeft het karakter van het beslissingsproces van beide projecten aan.



Figuur 7: Soorten beslissingen in de projecten ODZ en BeFlex

Opvallend is dat in beide projecten weinig aandacht is besteed aan de gebieden proces, organisatie, richtlijn en verbod. Het merendeel van de beslissingen valt in de gebieden structuur, gedrag, technologie en tools. Deze indicatie van het soort beslissingen dat wordt genomen helpt uiteindelijk bij de doelstelling van het onderzoek, het verbeteren van de ondersteuning bij het maken van beslissingen. Bij het opstellen van een verbeterde ondersteuning in hoofdstuk 4 zullen deze bevindingen worden meegewogen.

2.4. Ontwikkeld beslissingsmodel

Uit de interviews met architecten is gebleken dat er geen expliciete volgorde wordt aangehouden bij het nemen van beslissingen. Het risico hiervan is dat er veel tijd wordt geïnvesteerd in het maken van beslissingen die misschien automatisch genomen kunnen worden door met een andere beslissing te beginnen. Het is gewenst om zo veel mogelijk beslissen in zo min mogelijk stappen te maken. Aan de hand van de twee bestudeerde frameworks en de twee gemaakte conceptuele modellen kan gekeken worden welke volgorde van het nemen van beslissingen het efficiënts is. Om dit te bereiken wordt er begonnen met de beslissingen die de meeste impact hebben op andere beslissingen. Dit kan worden afgeleid van de afhankelijkheden die zijn aangegeven in de modellen van de twee frameworks (Bijlage C en Bijlage D). In beide figuren is bijvoorbeeld te zien dat de applicatieverwerkingsomgeving veel invloed heeft op andere beslissingsgebieden en dat de verdeling van modules weer afhankelijk is van het gebruik van componenten en softwarepakketten. Deze zijn om hun beurt weer afhankelijk van de technische invulling van het systeem. Dit betekent dan de verdeling van modules indirect ook afhankelijk is van de technische invulling van het systeem. Figuur 8 combineert alle afhankelijkheden uit Bijlage C en Bijlage D waarbij alle afhankelijkheden van boven naar beneden geordend zijn. Op die manier is te zien welke beslissingsgebieden genomen moeten worden voordat er een beslissing kan worden genomen in het volgende gebied. Het model toont aan wat als eerste beslist moet worden. Een architect binnen GPR begint altijd bij het onderscheiden van de verschillende functionaliteiten. Als hij bij het punt aankomt van de verdeling van modules, dan moet er eerst een beslissing worden genomen over de technische invulling van het systeem. Het is het efficiëntst om in het technische gedeelte te beginnen bij de applicatieverwerkingsomgeving, omdat die de meeste invloed heeft op de rest van de beslissingsgebieden. Een aantal beslissingsgebieden en afhankelijkheden zijn in het model als optioneel aangegeven:

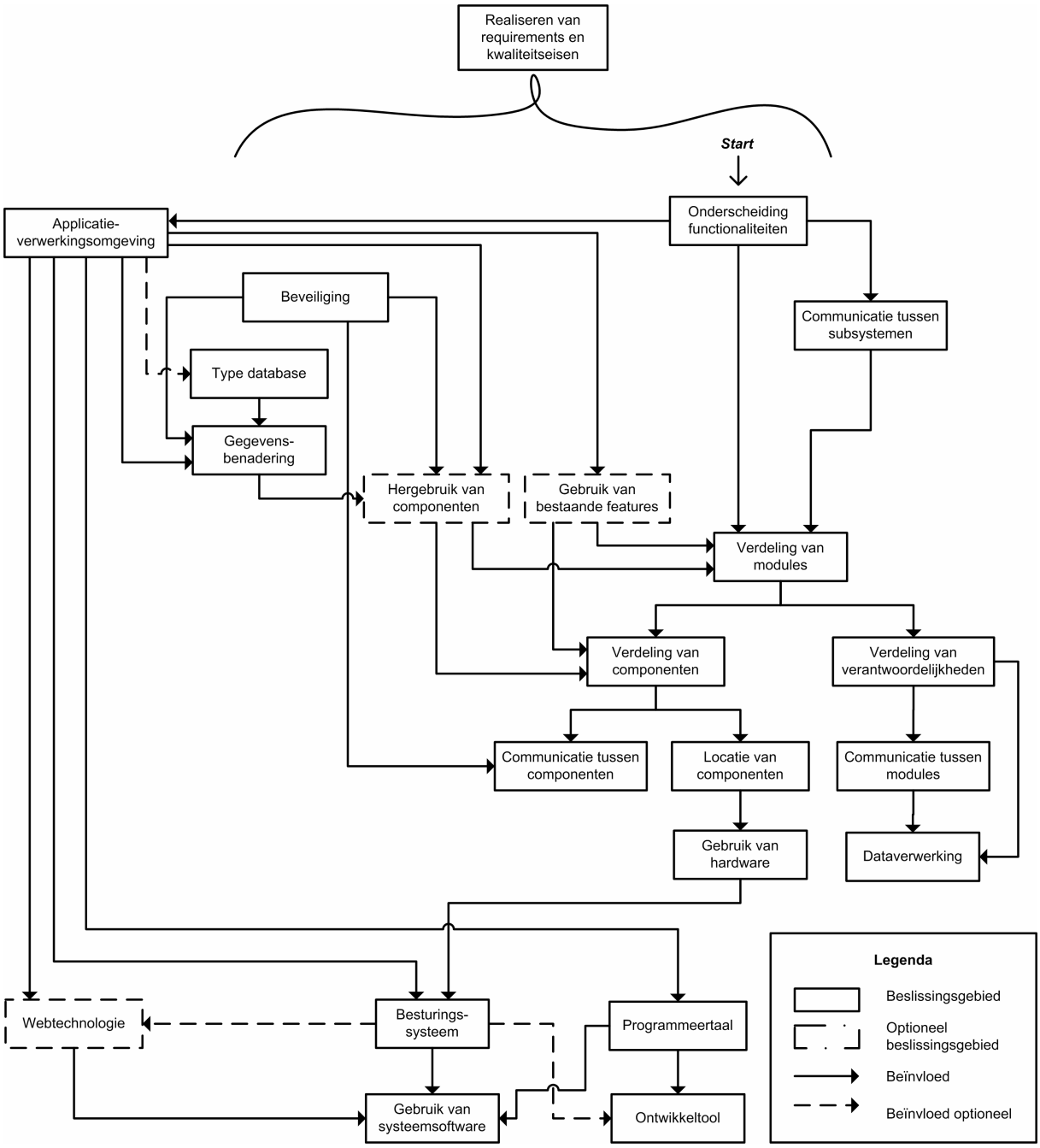
§ *Type database en applicatieverwerkingsomgeving*

In het geval van een Microsoft omgeving is er vaak een afhankelijkheid tussen het type database en de applicatieverwerkingsomgeving. Als er wordt gekozen voor MS .NET, dan kiest de architect ook vaak voor MS SQL Server. De combinatie van deze twee heeft zich bij de klanten van GPR bewezen en wordt daarom veel gebruikt.

§ *Bestaande componenten en softwarepakketten*
Het gebruik van bestaande componenten en softwarepakketten hangt sterk af van de eisen van de klant, het type project en de applicatieverwerkingsomgeving. De aanwezige componenten/softwarepakketten hoeven daarom niet altijd te 'passen' binnen het systeem.

§ *Webtechnologie*
Het kiezen van een webtechnologie is alleen van toepassing als het om een webapplicatie gaat.

§ *Ontwikkeltool en besturingssysteem*
In het geval van bijvoorbeeld de programmeertaal JAVA kan er gekozen worden voor verschillende ontwikkeltools. Bij het programmeren in JAVA op een UNIX computer wordt een andere ontwikkeltool gebruikt dan bij het programmeren in JAVA op een Windows computer. De programmeertaal C# gaat echter altijd samen met een versie van MS Visual Studio.



Figuur 8: Ontwikkeld beslissingsmodel

Uit het beslissingsproces van project ODZ en project BeFlex en de interviews met de architecten blijkt dat aan de hand van de applicatieverwerkingsomgeving de gehele technische invulling bepaald wordt. Er is echter geen absolute volgorde te definiëren, omdat dit per project anders is. Afhankelijk van de referentiearchitectuur of wensen van de klant, wordt bepaald welke keuze als eerste gemaakt moeten worden. Als de klant eist dat er C# gebruikt wordt in een webapplicatie, dan wordt dat als eerst besloten waarna de architect vervolgens de rest van de technische invulling van het systeem bepaalt. Het bovenstaande beslissingsmodel maakt alle afhankelijkheden duidelijk en zet de beslissingen met de meeste impact bovenaan, maar het model schrijft geen absolute volgorde voor. Afhankelijk van de wensen van de klant kan pas bepaald worden met welke beslissingen begonnen moet worden.

2.5. Validatie

Het generieke beslissingsmodel is gevalideerd met een architect die niet betrokken is geweest bij de validatie van de eerste twee beslissingsmodellen. Er is gekeken of het model compleet en correct is en of het aansluit op de belevingswereld van een architect binnen GPR. Het was voor de architect vrijwel meteen duidelijk wat het model inhield. De beslissingsgebieden kwamen sterk overeen met zijn gedachtegang over het beslissingsproces. Het model is op een aantal punten veranderd na de validatie met de architect. Het model is grofweg op te delen in twee delen:

- Het opdelen van de functionaliteiten en systeemdelen
- Het bepalen van de technische invulling voor het systeem

In het model vóór de validatie stonden deze twee delen voornamelijk los van elkaar. Hetgeen dat de delen met elkaar verbond waren de beslissingsgebieden *hergebruik van componenten* en *gebruik van bestaande features*. Deze twee beslissingsgebieden hebben zowel invloed op de opdeling van het systeem als de technische invulling van het systeem. Het is daarom logisch dat het bepalen van bestaande componenten en features vrij hoog in de beslissingsboom staan. De architect heeft echter aangegeven dat de functionele verdeling van het systeem wel invloed heeft op de technische invulling van het systeem. Als de functionele verdeling erg veel nadruk legt op de user interface, dan wordt er snel gekozen voor de programmeertaal met veel grafische mogelijkheden zoals JAVA of een dotNET variant. Als de functionaliteit meer gericht is op batchverwerking zonder user interface, dan wordt er eerder gekozen voor een programmeertaal met veel efficiëntie in de gecompileerde code zoals C++. In het model is daarom een afhankelijkheid aangegeven tussen het beslissingsgebied *onderscheiding functionaliteiten* en *applicatieverwerkingsomgeving.* De ‘start’ van het beslissingsproces is daarom ook gezet bij de onderscheiding van functionaliteiten. Het beslissingsgebied *bepalen van de architectuurgebieden* heeft in de modellen van de projecten ODZ en BeFlex niet veel afhankelijkheden en de architecten houden zo goed als altijd de indeling aan zoals MArch die beschrijft. In het generieke beslissingsmodel in dit beslissingsgebied daarom weggelaten. Verder heeft de architect aangegeven dat de keuze voor bepaalde hardware invloed kan hebben op de keuze voor het besturingssysteem. In het oude beslissingsmodel was deze afhankelijkheid niet aanwezig. Alle hiervoor beschreven veranderingen die uit de validatie met de architect zijn gekomen, zijn verwerkt in het bovenstaande beslissingsmodel.

2.6. Conclusie

Het doel van dit hoofdstuk was om een antwoord te krijgen op de eerste onderzoeksvraag met de volgende deelvragen:

1.1 Welke beslissingen worden genomen in de informatievoorzieningarchitectuur en de infrastructuurarchitectuur?
1.2 In welke groepering/volgorde zijn deze beslissingen te verdelen?
1.3 Welke afhankelijkheden zijn te vinden tussen de beslissingen?
1.4 Welke beslissingen hebben de meeste impact en moeten als eerste genomen worden?

1.1 Welke beslissingen worden genomen in de informatievoorzieningarchitectuur en de infrastructuurarchitectuur?

Zoals beschreven staat kan een lijst met beslissingsgebieden niet uit de literatuur gehaald worden. Daarom is er onderzoek gedaan naar het beslissingsproces van de architecten binnen de organisatie van GPR. In de oriënterende interviews in de eerste fase van het onderzoek is ingegaan op de onderwerpen waarover een architect moet beslissen bij het opstellen van een architectuur en wat voor volgorde hij hierin aanhoudt. De vragen op dit gebied leverde zeer weinig informatie op. Om deze reden is er gekozen om de aanwezigen architecturen binnen GPR te analyseren op beslissingen. Het resultaat van deze analyse is het generieke beslissingsmodel in Figuur 8. Dit model geeft de beslissingsgebieden aan die een architect moet doorlopen bij het opzetten van een informatievoorzieningarchitectuur en een infrastructuurarchitectuur.

1.2 In welke groepering/volgorde zijn deze beslissingen te verdelen?

Het groeperen van beslissingen, om zo de beslissingen tot een conceptueel niveau te tillen, is een creatief proces en daarom erg afhankelijk van de interpretatie van de auteur. Voor het doel van dit onderzoek maakt dat echter

niet veel verschil. Het doel van dit hoofdstuk was om een beslissingsmodel te creëren, dat aansluit op de belevingswereld van de architecten binnen GPR. Alle beslissingsmodellen zijn uitgebreid gevalideerd met diverse architecten. Uit het onderzoek van dit hoofdstuk is gebleken dat er geen vaste volgorde in beslissingen te definiëren is, omdat deze erg afhankelijk is van de wensen en eisen van de klant. Een aantal keuzes liggen van te voren al vast, omdat de klant bijvoorbeeld richtlijnen voorschrijft in de referentiearchitectuur. Daarnaast wordt de volgorde deels bepaald door de afwegingen die een architect moet maken.

1.3 Welke afhankelijkheden zijn te vinden tussen de beslissingen?

Omdat de beslissingsgebieden doormiddel van eigen inbreng zijn opgesteld, zijn ook de afhankelijkheden tussen deze gebieden aan de hand van eigen ervaring en de oriënterende gesprekken met de architect uit de beginfase van het onderzoek in kaart gebracht. Het model van deze eerste opzet is gevalideerd met de betrokken architect van het project. Het model is niet meteen in zijn geheel aan de architect getoond, maar er is per architectuurdomein ingegaan op de afhankelijkheden. Op die manier heeft de architect bij elke afhankelijkheid de mogelijkheid om de correctheid te controleren. Het resultaat van deze validatie van de afhankelijkheden is verwerkt en weergegeven in Figuur 8.

1.4 Welke beslissingen hebben de meeste impact en moeten als eerste genomen worden?

Zoals in deelvraag 1.2 is beschreven, is er geen vaste generieke volgorde in beslissingen te definiëren. Er zijn wel een aantal beslissingsgebieden omschreven die zeer veel impact hebben op de overige beslissingen. Een vaste volgorde wordt hier niet aangegeven, omdat deze sterk afhangt van het project en de wensen/eisen van de klant. Door de beslissingen met de meeste impact als eerst te maken, kan het aantal mogelijkheden van de overige beslissingen beperkt worden. De architect kan op die manier tijd besparen en een betere afweging maken, omdat hij uit minder alternatieven hoeft te kiezen. Dit maakt de bevindingen betreffende de afhankelijkheden/impact van de beslissingen bruikbaar.

Het ontwikkelde beslissingsmodel dient voor de komende fases in het onderzoek twee doelen. Het wordt gebruikt om te achterhalen welke informatie een architect nodig heeft bij het maken van beslissingen. Hiervoor moet uiteraard bekend zijn met welke beslissingsgebieden een architect te maken krijgt, het beslissingsmodel. En het beslissingsmodel wordt gebruikt om de uiteindelijke ondersteuning te laten aansluiten op de belevingswereld van een architect. Welke informatie moet als eerst achterhaald worden (via de afhankelijkheden/volgorde uit het model) en hoe sluit dit aan op de beslissingen die een architect moet nemen? Dit moet voor een architect glashelder zijn als hij de ondersteuning gebruikt bij het opzetten van een architectuur.

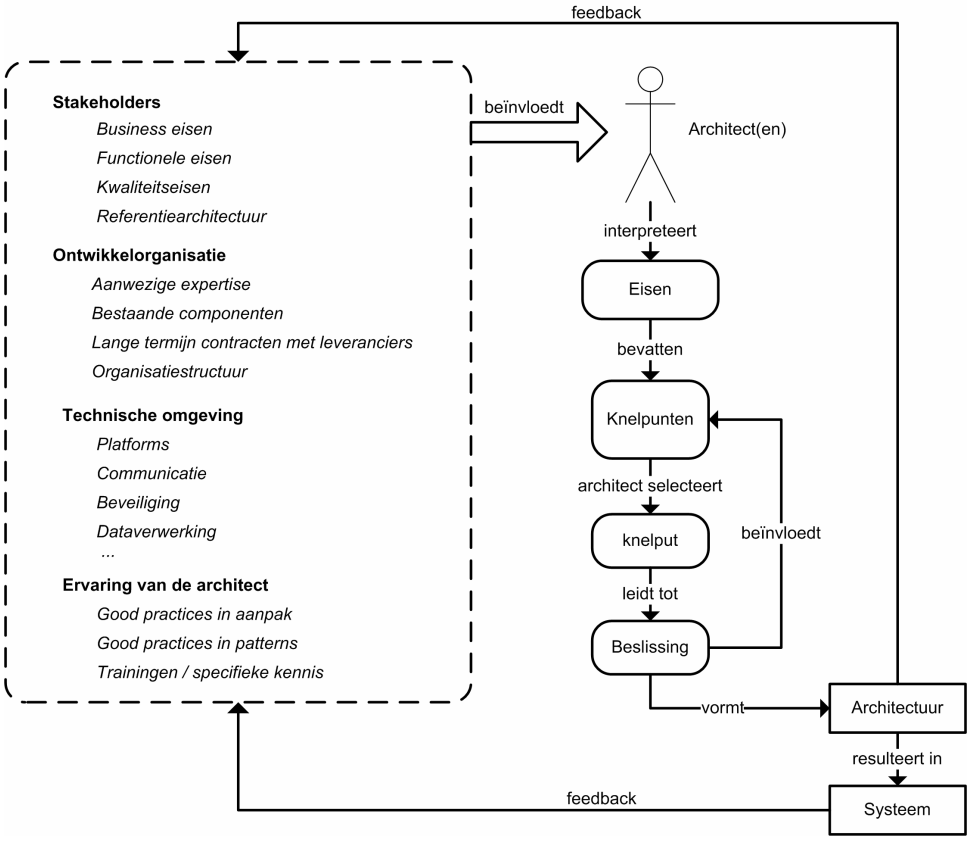
3. Informatiebehoeftes van de architect

3.1. Inleiding

In dit hoofdstuk wordt een antwoord gegeven op de tweede onderzoeksvraag:

> *In welke beslissingsgebieden kan met welke informatieondersteuning de meeste winst behaald worden?*

Er is in het voorgaande hoofdstuk in kaart gebracht welke beslissingsgebieden binnen GPR aan bod komen bij het opzetten van een softwarearchitectuur. **Welke informatie heeft een architect nu nodig om in elk beslissingsgebied tot een keuze te kunnen komen?** Deze vraag moet beantwoord worden, om te weten te komen hoe een architect het beste ondersteund kan worden bij het nemen van beslissingen. Informatie acquisitie is het proces waarbij deze vraag centraal staat, het vergaren van de juiste informatie. Informatie acquisitie is één van de belangrijkste en problematische processen bij het ondersteunen van de architect [Fujihara 97]. Het expliciet maken van dit proces is mede lastig omdat experts niet goed duidelijk kunnen maken wanneer ze welke informatie waar vandaan halen. De meest voorkomende techniek voor het expliciet maken van dit proces is het houden van interviews. [Money 95] geeft aan dat experts hun kennis en informatievergaring het beste duidelijk kunnen maken als ze concrete voorbeelden voorgelegd krijgen die werkelijk kunnen voorkomen in een project. Omdat een informatiebehoefte ontstaat uit een beslissingsgebied waarin de architect tot een keuze moet komen, wordt het beslissingsmodel uit hoofdstuk 2.4 gebruikt. Elk beslissingsgebied wordt met concrete voorbeelden besproken met een architect en per beslissingsgebied wordt ingegaan op de informatiebehoeftes die een architect heeft om tot een keuze te kunnen komen en hoe de informatie op dit moment binnen GPR vergaard wordt. Omdat een architect zelf moeilijk kan aangeven wat voor informatie hij gebruikt, wordt er een referentiemodel met informatiebehoeftes gebruikt. [Bass 03, p.11] omschrijft de aspecten die een architect kunnen beïnvloeden bij het opzetten van een architectuur. Een uitwerking van deze aspecten is te zien in Figuur 9.

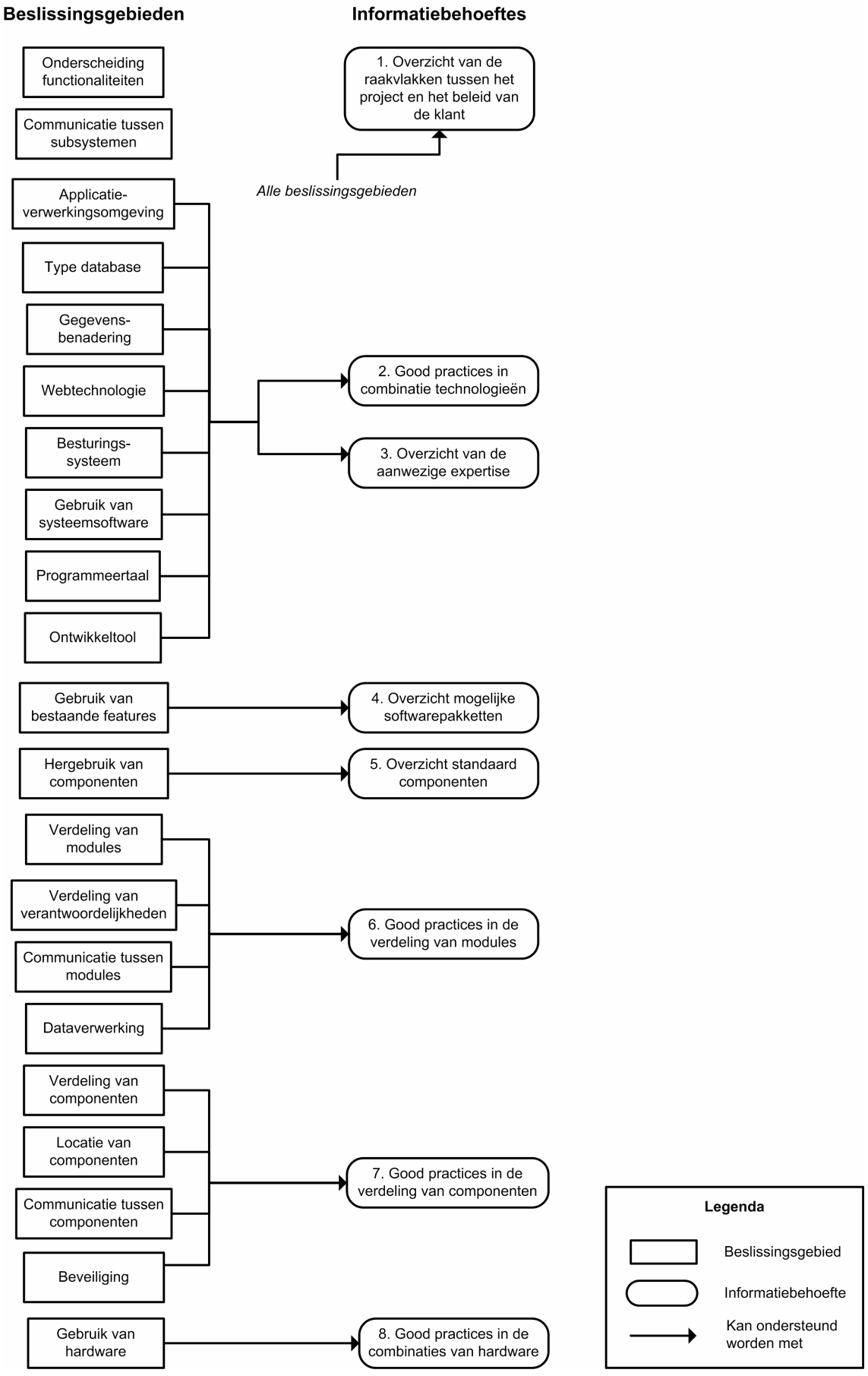


Figuur 9: Aspecten die een architect kunnen beïnvloeden [Bass 03]

3.2. Informatiebronnen, behoeftes en knelpunten

Aan de hand van gehouden interviews, het ontwikkelde beslissingsmodel uit hoofdstuk 2 en [Bass 03] (Figuur 9) is een aantal informatiebehoeftes naar boven gekomen. Per beslissingsgebied is samen met de architect bekeken welke informatiebehoeftes hij nodig heeft om tot een keuze te kunnen komen. Het diagram uit [Bass 03] is hierbij gebruikt om de architect te wijzen op informatiebehoeftes waar hij zelf niet aan had gedacht. Elk beslissingsgebied is besproken met een voorbeeld, om het voor de architect gemakkelijk te maken om de informatiebehoeftes kenbaar te maken. Bij het beslissingsgebied *type database* is bijvoorbeeld een case geschetst waarin een klant met een zeer grote hoeveelheid data moet werken, veel overzichten maakt en zijn gegevens regelmatig op een backup wil zetten. Met een dergelijke case wordt een beslissingsgebied concreter voor de architect. De gevonden informatiebehoeftes voor de beslissingsgebieden zijn weergegeven in Figuur 10. Een aantal beslissingsgebieden hebben dezelfde informatiebehoeftes en worden daarmee gebundeld. Een verklaring hiervoor is dat deze beslissingen vaak een logisch gevolg zijn van elkaar en dat de architect daarom dezelfde informatiebehoeftes heeft bij de beslissingsgebieden.

De volgorde van informatiebehoeftes is tot stand gekomen aan de hand van de afhankelijkheden en optimale volgorde van beslissingen in het ontwikkelde beslissingsmodel uit hoofdstuk 2. Deze volgorde staat omschreven in de analyse uit subhoofdstuk 2.3 onder het kopje “volgorde in beslissingen”.



Figuur 10: Relatie tussen de beslissingsgebieden en de informatiebehoeftes

Onderstaande opsomming geeft aan in welke volgorde de gevonden informatiebehoeftes aan bod komen.

1. Overzicht van de raakvlakken tussen het project en het beleid van de klant
2. Good practices in de combinaties van technologieën
3. Overzicht van de aanwezige expertise binnen GPR
4. Overzicht van de mogelijke softwarepakketten
5. Overzicht van de mogelijke standaard componenten voor hergebruik
6. Good practices in de verdeling van modules
7. Good practices in de verdeling van componenten
8. Good practices in de combinaties van hardware

In bijlage F is per informatiebehoefte een korte omschrijving met motivatie gegeven. Elke omschrijving van een informatiebehoefte is voorzien van drie punten:

Welke beslissingsgebieden zijn erbij betrokken?
Als er in de ondersteuning wordt ingegaan op een bepaalde informatiebehoefte, dan moet het voor de architect duidelijk zijn wat hij aan de informatie heeft. In welke beslissingsgebieden wordt de architect ondersteund met de informatie. Daarom is de relatie tussen de informatiebehoeftes en de beslissingsgebieden zeer belangrijk.

Waar haalt de architect nu de informatie vandaan?
Om te kunnen analyseren welke informatie de meeste winst (tijd en kwaliteit) kan opleveren voor de architect, is het noodzakelijk om te achterhalen hoe de informatie op dit moment achterhaald wordt. Waar gaat de meeste tijd in zitten en welke informatie is het minst toegankelijk.

Welke informatie achterhaald moet worden bij de architect om ondersteuning te kunnen bieden in de informatiebehoeftes ("afhankelijkheden / criteria")?
Met dit punt kan later in het onderzoek een brug worden geslagen naar een verbetering in de intakefase van de huidige ondersteuning. De dikgedrukte woorden in dit kopje geven de onderwerpen aan die in de intakefase aangehaald worden om de juiste informatie van een architect te achterhalen, zodat er op het juiste moment ondersteuning geboden kan worden in de informatiebehoeftes van de architect.

In de beschrijving van de informatiebehoeftes zijn twee referenties gebruikt, welke worden meegenomen in de conclusie van dit hoofdstuk. Kwaliteitseisen kunnen verdeeld worden in twee groepen, "runtime" kwaliteitseisen en "designtime" kwaliteitseisen [Clements 06]. Bij de *good practices in de verdeling van modules* gaat het voornamelijk om de statische "designtime" kwaliteitseisen, zoals onderhoudbaarheid, bouwbaarheid en time-to-market. De dynamische "runtime" kwaliteitseisen, zoals prestatie en integratie, spelen een belangrijke rol bij de *good practices in de verdeling van componenten*. Het achterhalen van deze kwaliteitseisen kan niet door simpelweg een kwaliteitsattribuut zoals Beveiliging als belangrijk aan te vinken. Kwaliteitsattributen worden pas concreet in een context. Voor het concreet maken van kwaliteitseisen en deze te beschrijven in een context, kan gebruik gemaakt worden van zogenaamde **kwaliteitsattribuut scenario's** [Bass 03]. Op die manier kunnen kwaliteitseisen duidelijk omschreven worden, gekoppeld worden aan een kwaliteitsattribuut en gemakkelijk gebruikt worden om good practices te kunnen hergebruiken.

3.3. Analyse en bevindingen

Uit de gevonden informatiebehoeftes is te ontdekken dat er vier verschillende soorten informatiebehoeftes onderscheiden kunnen worden:

Type 1: Informatie over good practices en mogelijke alternatieven per beslissingsgebied
(afgeleid van informatiebehoefte 2, 4, 5, 6, 7, 8)
Als de architect veel vrijheid heeft in het nemen van beslissingen, dan moet hij weten wat de mogelijke keuzes zijn en wat de gevolgen per keuze zijn. Een architect is bij het opzetten van een architectuur in een "green-field" situatie de meeste tijd kwijt met het bedenken van alternatieven [Kruchten 04]. Met een overzicht van alle mogelijke oplossingen, gevolgen per oplossing en good practice kan de architect veel tijd besparen en een overtuigende afweging (trade-off) maken.

Type 2: Informatie over het beleid van de klant

(afgeleid van informatiebehoefte 1)

Als de architect te maken heeft met een klant die een uitgewerkt beleid gebruikt voor het werken met architecturen, dan heeft de architect niet veel aan mogelijke alternatieven per beslissingen of good practices. De klant schrijft in zo'n geval voor waar een architect zich aan moet houden en er is weinig ruimte voor een afweging tussen verschillende oplossingen (de afweging). Op zo'n moment wil de architect weten op welke vlakken het project in aanraking is met het beleid van de klant. Op welke punten liggen de beperkingen van de architect en welke keuzes liggen van te voren al vast?

Type 3: Informatie over wat er wanneer beslist moet worden (aandachtspunten)

(afgeleid van de huidige ondersteuning binnen GPR)

Een derde vorm van informatie is het bieden van aandachtspunten voor een architect. Deze vorm van informatie is niet door de architect aangegeven als zeer ondersteunend, maar het maakt het mogelijk om veel voorkomende fouten te voorkomen en om een aantal punten voor te dragen waar een architect niet aan zou hebben gedacht. Het is opvallend dat deze vorm van informatie niet expliciet is genoemd door de architecten, maar dat in de huidige ondersteuning daar wel de nadruk op ligt.

Type 4: Overzicht van aanwezige expertise

(afgeleid van informatiebehoefte 3)

Elke afdeling binnen GPR heeft één of meerdere experts op een bepaald gebied. Als een architect meer wil weten of een bepaald onderwerp, dan weet hij uit ervaring enkele contactpersonen op de verschillende afdelingen. Elke architect heeft een eigen kenniskring met experts en de bijbehorende kennisgebieden. Deze informatie is niet vastgelegd, terwijl veel architect wel baat zouden hebben bij dergelijke lijst.

Met welk type informatiebehoefte valt nu de meeste winst te behalen? Type 1 is gericht op het bieden van oplossingen en is van toepassing als een architect veel vrijheid heeft in het nemen van beslissingen. Binnen GPR worden bijna alle projecten voor ZBO gedaan, de grootste klant. Deze klant heeft een zeer uitgebreide referentiearchitectuur, waarin beschreven staat waar een architectuur aan moet voldoen. De architect heeft daarom in de meeste gevallen niet veel aan mogelijke oplossingen, alternatieven en good practices (type 1). De taken van de architect bestaan voornamelijk uit het zorgen voor een goede aansluiting tussen het te bouwen systeem en het beleid van de klant. In de meeste projecten wil een architect weten welke beslissingsgebieden al door de klant bepaald zijn en welke beslissingsgebieden hierdoor beperkt zijn.

Een lijst met aandachtspunten (type 3) is nuttig voor een architect, maar moet wel aansluiten op de belevingswereld van een architect. Als de architect één lange lijst krijgt met aandachtspunten, adviezen en valkuilen, dan zal de architect een afkeer vormen tegen de bulk aan informatie. Als de architect gepaste informatie krijgt geordend per beslissingsgebied, dan krijgt de informatie meer herkenning bij de architect en daarmee meer nut en draagvlak.

De volgende genummerde lijst geeft de te behalen winst (tijd en kwaliteit) aan van de verschillende type informatiebehoeftes (van veel winst naar weinig winst):

1. *Informatietype 2: Het beleid van de klant*

   De meeste projecten worden gedaan voor de klanten met een eigen beleid. De klant is tevreden als het systeem aansluit op hun beleid. Als de architect ondersteund wordt in de aansluiting tussen het project en het beleid van de klant, dan heeft dat directe gevolgen voor de klanttevredenheid. Daarnaast kan er aan de kant van de architect veel tijd bespaart worden, doordat het zoeken in de referentiearchitect voor een groot deel vermeden kan worden.

2. *Informatietype 1: Good practices en mogelijke alternatieven*

   Als GPR een nieuw product op de markt zet of een project aanneemt van een klant zonder eigen referentiearchitectuur, dan kunnen bewezen oplossingen grote impact hebben op de kwaliteit van het systeem. In dergelijke "green-field" situatie heeft de architect alle vrijheid in het nemen van beslissingen. Deze vrijheid heeft als risico dat een architect kiest voor een oplossing, waarvan de gevolgen niet te voorspellen zijn. Good practices in de vorm van bewezen oplossingen geven de architect meer zekerheid over de gevolgen van beslissingen en geven daarmee ook meer zekerheid over de kwaliteit van het systeem.

3. *Informatietype 3: Aandachtspunten*
   Aandachtspunten kunnen veelvoorkomende fouten voorkomen, maar uit de interviews blijkt dat de meeste architecten weinig behoefte hebben aan de aandachtspunten/stappen die nu worden gegeven door de huidige ondersteuning. De punten die nu in het document met ondersteunde informatie gegenereerd worden zijn al bekend bij de ervaren architecten en hebben voor hun weinig meerwaarde. Dit hoeft uiteraard niet te betekenen dat de informatie niet belangrijk is. Toch heeft deze vorm van informatie een lage prioriteit omdat de veel voorkomende fouten op dit moment worden voorkomen, door de minder ervaren architecten te laten beoordelen door de ervaren architecten. Voor de leercurve (en daarmee de kwaliteit van de beslissingen) is het zelfs beter om de beginnende architect eerst de fouten te laten maken, waarna ze gecorrigeerd worden door een ervaren architect.

4. *Informatietype 4: Aanwezige expertise*
   Ondanks dat de aanwezige expertise niet is vastgelegd in een overzichtelijk lijst, komen architecten wel aan de juiste expertise als dat nodig is. In dit proces kan wel wat tijdswinst worden geboekt, maar het heeft geen hoge prioriteit.

3.4. Conclusie

Het doel van dit hoofdstuk was om een antwoord te krijgen op de tweede onderzoeksvraag met de volgende deelvragen:

2.1 Op basis van welke informatie en kennis maakt de architect elke beslissing?
2.2 Welke informatiebronnen zijn aanwezig?
2.3 Wat zijn de informatiebehoeftes van een architect?

2.1 Op basis van welke informatie en kennis maakt de architect elke beslissing?
Er is in kaart gebracht wat de informatiebehoeftes zijn en welke informatie nodig is om de architect te kunnen voorzien in zijn informatiebehoeftes. Aan de hand van de verschillende beslissingsgebieden waar een architect mee te maken krijgt, is achterhaald welke informatie een architect gebruikt om tot beslissingen te kunnen komen. Opvallend is dat het beleid van de klant in de meeste gevallen een groot deel van de beslissingen bepaald. De grootste klant van GPR, ZBO, heeft zo'n uitgebreide referentiearchitectuur, dat de architect bijna geen ruimte meer heeft om af te wijken in de beslissingen.

2.2 Welke informatiebronnen zijn aanwezig?
Tijdens de verschillende interviews met de architecten is uitgebreid ingegaan op het informatie acquisitieproces tijdens het opzetten van een architectuur. Welke informatiebronnen worden wanneer geraadpleegd? Uit deze interviews is ontdekt dat zeer veel informatie wel bekend is bij de ervaren architecten, maar niet expliciet is vastgelegd. Dit maakt het zeer lastig om zicht te krijgen in het beslissingsproces van de architect en om de kwaliteit van de architecturen te kunnen verbeteren. De belangrijkste impliciete informatiebronnen die tijdens de interviews naar boven zijn gekomen, zijn overzichten van de aanwezige expertise binnen GPR, overzichten van de veelgebruikte componenten in de ontwikkelteams en overzichten van de veelgebruikte softwarepakketten binnen GPR.

2.3 Wat zijn de informatiebehoeftes van een architect?
Omdat het beleid van een klant met een referentiearchitectuur zeer bepalend is voor de beslissingen die een architect moet maken, heeft de architect veel behoefte aan een overzicht van de raakvlakken tussen het project en de referentiearchitectuur van de klant. Als de architect aan een project werkt voor een klant zonder referentiearchitectuur, dan heeft hij de meeste behoefte aan good practices. Er is geconstateerd dat deze good practices op verschillende abstractieniveaus en op verschillende momenten in het beslissingsproces voorkomen. Er zijn good practices in de verdeling van modules, maar bijvoorbeeld ook op het gebied van combinaties in hardware. Deze good practices worden op dit moment niet nauwkeurig vastgelegd, omdat de ondersteuning hiervoor (de FWB) niet goed aansluit op het beslissingsproces en de informatiebehoeftes van de architect en daarom onvoldoende gebruikt wordt. Naast good practices mist de architect op dit moment een overzicht van de verschillende experts binnen GPR, de aanwezige componenten en de veel gebruikte software pakketten die zich binnen GPR bewezen hebben. In de lijst met informatiebehoeftes is onderscheid gemaakt tussen de verschillende soorten informatie en er is bepaald welke informatiebehoeftes de meeste winst kunnen opleveren.

4. Verbeterde ondersteuning in de informatie acquisitie

4.1. Inleiding

In de voorgaande twee hoofdstukken is onderzocht hoe het beslissingsproces verloopt en wat de informatiebehoeftes zijn van een architect binnen GPR. In dit hoofdstuk zullen de voorgaande twee hoofdstukken worden gebruikt om tot een verbeterde aansluiting te komen tussen de architect en de huidige ondersteuning binnen GPR, de FrameWorkBench (FWB). Na een korte terugblik op de huidige intakefase, wordt in dit hoofdstuk een antwoord gevonden op de derde doelstelling:

> *Het verbeteren van de koppeling tussen het beslissingsproces en de informatiebehoeftes van een architect en de huidige ondersteuning, zodat de FWB beter ingezet kan worden bij het ondersteunen van de architect.*

4.2. Beoordeling huidige ondersteuning

Aansluiting tussen de huidige ondersteuning en de informatiebehoeftes

Tijdens de probleemfase van het onderzoek zijn twee cases in de FWB ingevoerd en zijn de gegenereerde richtlijnen van beide cases met elkaar vergeleken. Ondanks het feit dat de cases volledig verschillend waren op de onderwerpen uit de vragenlijst [Bijlage A], waren er nauwelijks verschillen te ontdekken in de gegenereerde richtlijnen van beide cases. De impact van de vragenlijst is in de huidige ondersteuning minimaal. In de analyse van de informatiebehoeftes uit hoofdstuk 0 is de informatiebehoefte van een architect uit elkaar getrokken en is er onderscheid gemaakt tussen verschillende types informatie. Door te kijken naar deze verschillende type informatie is te verklaren waarom de vragenlijst zo weinig invloed heeft. Vragen over systeemkenmerken, technologie voorkeuren, kwaliteitseisen kunnen gebruikt worden voor het bepalen welke oplossingen, good practices en alternatieven (type 1) gebruikt kunnen worden. De richtlijnen die door de FWB gegenereerd worden, gaan echter in op de stappen die een architect moet nemen op de verschillende momenten (type 3). Omdat de vragenlijst en de uiteindelijke richtlijnen allebei ingaan op verschillende informatie types, sluiten deze twee niet goed op elkaar aan. De architect krijgt vervolgens niet de informatie die hij verwacht en beoordeelt de FWB als onbruikbaar. Voorbeelden uit de uitgevoerde cases uit hoofdstuk 1.4.1 (probleemanalyse) geven aan dat het antwoord op een vraag niet aansluit op de gegenereerde richtlijnen:

- ***Antwoord in de vragenlijst***
  In de vragenlijst wordt aangegeven dat beschikbaarheid zeer belangrijk is
  Gegenereerde richtlijnen
  Bepaal op welk platform de applicatie services beschikbaar gesteld moeten worden. Maak deze overweging op basis van betrouwbaarheid, beschikbaarheid, schaalbaarheid benodigde capaciteit, kosten of voorkeur van de klant.
  Toelichting
  Er is aangegeven dat beschikbaarheid zeer belangrijk is. De architect verwacht dan ook ondersteunende informatie waarin duidelijk wordt hoe deze beschikbaarheid bereikt kan worden (informatietype 1). Daarentegen bevat de informatie niet meer dan een aandachtspunt (informatietype 3) waarin staat dat de architect een overweging moet maken, terwijl de architect dat juist in de vragenlijst heeft aangegeven.

- ***Antwoord in de vragenlijst***
  In de vragenlijst wordt aangegeven dat J2EE wordt gebruikt als ontwikkelomgeving
  Gegenereerde richtlijnen
  Bepaal of de .NET Web Richtlijnen dan wel de J2EE Richtlijnen opgevolgd moeten gaan worden.
  Toelichting

In de vragenlijst wordt een bepaalde oplossingsrichting aangegeven (informatietype 1), namelijk J2EE. De gegenereerde informatie geeft echter een aandachtspunt weer (informatietype 3) dat niet meer van toepassing is als de architect al heeft aangegeven dat er voor J2EE is gekozen.

De voorbeelden laten zien dat de antwoorden op de vragen vaak niet goed overeenkomen met de richtlijnen die gegenereerd worden. Aan de hand van de vragen kan een architect de beperkingen in het project aangegeven, zodat bepaalde keuzes worden uitgesloten. De architect verwacht dan ook dat deze aangegeven beperkingen terug te zien zijn in de gegenereerde informatie en dat bepaalde oplossingen uitgesloten of juist voorgesteld worden (informatietype 1). Voor dit onderzoek is belangrijk dat er geconstateerd is dat doel van de huidige intakefase niet duidelijk is voor de architect en dat de verschillende type informatie door elkaar worden gebruikt.

Categorisering en automatisch beslissingen maken

Het beslissingmodel uit hoofdstuk 2.4 geeft aan welke beslissingsgebieden altijd aan bod komen bij het opzetten van een softwarearchitectuur. De huidige FWB is niet gespitst op deze beslissingsgebieden, waardoor de architect moeilijk een verband kan leggen tussen de intakefase en zijn eigen belevingswereld. Er wordt niet duidelijk aangegeven waarom een bepaalde vraag wordt gesteld en welke architectuurbeslissingen met het antwoord worden beïnvloed.

Bepaalde beslissingen kunnen automatisch gemaakt worden, doordat ze afhankelijk zijn van eerder genomen beslissingen. Op dit moment speelt de FWB daar niet op in. Als er bijvoorbeeld gekozen wordt voor MS .NET, dan kan er aan de hand van de afhankelijkheden een uitspraak worden gedaan over het type database, de webtechnologie, de programmeertaal, de gegevensbenadering in de datalaag, de ontwikkeltool, het gebruik van systeemsoftware en het te gebruiken besturingssysteem. Dit gebeurt nu niet.

4.3. Succesfactoren

Aan de hand van de bevindingen uit hoofdstuk 2 en 3 kunnen zes succesfactoren worden opgesteld voor de intakefase van een architectuurondersteuning. Deze succesfactoren zijn gebaseerd op de belangrijkste bevindingen uit het onderzoek:

Uit de probleemanalyse van het onderzoek is gebleken dat:

- de aansluiting tussen de huidige ondersteuning en het beslissingsproces van de architect niet goed is en één van de hoofdoorzaken is waarom de ondersteuning onvoldoende draagvlak heeft bij de architecten (succesfactor 1).
- de intakefase niet goed inspeelt op de antwoorden van de architect, waardoor de meeste architecten hun twijfels hebben bij de relevantie van de vragen (succesfactor 2).

Uit de analyse van de beslissingsgebieden, hoofdstuk 2, is gebleken dat:

- de nadruk ligt op de structuur en het gedrag van het systeem volgens de beslissingssoorten van [Kruchten 04] (succesfactor 3).
- De lastigste beslissingsgebieden uit het ontwikkelde beslissingsmodel zijn “Verdeling van modules” en “Communicatie tussen modules” (succesfactor 4)

Uit de analyse van de informatiebehoeftes, hoofdstuk 3, is gebleken dat:

- de architect de meeste winst heeft aan informatie over het beleid van de klant (succesfactor 5).
- dat good practices op verschillende abstractieniveaus voorkomen in het beslissingsproces (succesfactor 6).

Succesfactoren:

1. In de probleemfase van het onderzoek is geconstateerd dat de huidige tooling voor het ondersteunen van de architect (de FWB) niet veel gebruikt wordt omdat de intakefase niet goed aansluit op het beslissingsproces en de informatiebehoeftes van de architect. De vragen die gesteld worden moeten aansluiten op de belevingswereld van de architect. Dit houdt in dat het duidelijk moet zijn waarom bepaalde vragen gesteld worden, op welke beslissingen ze betrekking hebben en hoe het de architect gaat helpen bij het nemen van beslissingen. Door dit duidelijk te maken, begrijpt de architect beter de relevantie van de vragen en de ondersteuning. Dit is misschien een open deur, maar zeker wel noodzakelijk om tot een succesvolle ondersteuning voor de architect te komen. Daarom is dit geconstateerde probleem uit de

probleemfase hier opgenomen als succesfactor. De resultaten uit hoofdstuk 2 geven aan met welke beslissingsgebieden een architect te maken krijgt en hoofdstuk 3 heeft aangetoond hoe een architect in deze beslissingsgebieden ondersteund kan worden (informatiebehoeftes). Door deze resultaten te combineren kan er een intakefase worden voorgesteld die aansluit op de onderwerpen waar een architect over moet nadenken en de informatie biedt die de architect op dat moment nodig heeft.

2. De vragenlijst dient dynamisch te zijn. Dit houdt in dat er moet worden ingespeeld op de antwoorden van de architect. Als de architect aangeeft dat het systeem gebouwd wordt in de webtechnologie ASP.NET, dan moet er geen vraag meer worden gesteld over de applicatieverwerkingsomgeving (dat is namelijk automatisch MS .NET). Aan de hand van afhankelijkheden kunnen technische beslissingen voor een groot deel automatisch worden gemaakt. Ook dit punt is al vroeg in het onderzoek geconstateerd en is misschien een open deur. Net als succesfactor 1 is ook dit punt noodzakelijk om tot een succesvolle ondersteuning te komen en daarom opgenomen als succesfactor.
3. Figuur 7 uit hoofdstuk 2.3 heeft aangetoond dat de architecten zich het meeste concentreren op de beslissingen over de structuur en het gedrag van het systeem. Beslissingen over het proces en de methodes en organisatie komen niet veel terug in de frameworks en worden ergens anders genomen. Ongeacht of dit goed is of niet, is het een constatering uit de praktijk waar de ondersteuning rekening mee moet houden.
4. De architecten hebben aangegeven dat de beslissingsgebieden “verdeling van modules” en “communicatie tussen modules” het lastigst zijn. De good practices op dit gebied kunnen bij deze beslissingen veel ondersteuning bieden.
5. Uit de analyse van hoofdstuk 3 is gebleken dat de architect de meeste baat heeft als hij inzicht krijgt in het beleid van de klant. Het beleid van de klant heeft vaak grote impact op de mogelijke keuzes in de te nemen architectuurbeslissingen. Om een architect effectief te kunnen ondersteunen in de beslissingen, moet dit beleid van de klant worden meegenomen in de ondersteunende informatie. Als de klant een eigen beleid heeft, dan levert een overzicht van de raakvlakken tussen het project en het beleid van de klant de meeste ondersteuning. Waar zitten de beperkingen voor de architect en wat schrijft het beleid van de klant (vaak in de vorm van een referentiearchitectuur) voor?
6. Good practices komen voor op verschillende abstractieniveaus in de architectuur (modules, componenten, implementatie, technologie, hardware). Deze onderscheiding moet ook verwerkt worden in de ondersteuning, zodat het voor een architect duidelijk is wanneer en waar hij de good practices kan inzetten.

4.4. Verbeteradvies

Succesfactor 1: Aansluiting op de belevingswereld van de architect

De volgorde van vragen die gesteld worden moet voor de architect logisch zijn en het moet duidelijk zijn waar de vragen betrekking op hebben. Figuur 8 toont de beslissingsgebieden die bij het opzetten van een architectuur aan bod komen. Deze gebieden kunnen gebruikt worden als herkenningspunten voor de architect. In hoofdstuk 3 is achterhaald welke informatiebehoeftes nodig zijn in de verschillende beslissingsgebieden en welke volgorde in het aanbieden van informatiebehoeftes het meest efficiënt is. Deze volgorde van informatiebehoeftes uit hoofdstuk 3 en de beslissingsgebieden uit hoofdstuk 2 zijn verwerkt tot een stappenplan voor de intakefase. In deze intakefase is het belangrijk dat het voor de architect zichtbaar is welke beslissingsgebieden betrokken zijn bij elke vraag. De onderstaande tabel geeft een aantal stappen aan die in de intakefase doorlopen moeten worden. Door in de stappen uit tabel 2 de betrokken informatiebehoeftes en beslissingsgebieden visueel weer te geven, is het voor een architect op elke moment duidelijk waar hij zich bevindt in het beslissingsproces en waar hij in ondersteund wordt. Elke stap in het stappenplan dekt één informatiebehoefte. De betrokken beslissingsgebieden per stap zijn rechtstreeks herleid uit Figuur 10 waarin de relatie tussen de informatiebehoeftes en de beslissingsgebieden is uitgelegd. De kolom met de onderwerpen van de vragen is herleid uit de kopjes “Afhankelijkheden / criteria” uit hoofdstuk 3. In de volgende succesfactoren zal dit stappenplan verder worden toegelicht.

Stap	Informatiebehoefte	Gedekte beslissingsgebieden	Onderwerp vragen
1	1. Overzicht van de raakvlakken tussen het project en het beleid van de klant	• Alle	Wie is de klant?
2	2. Good practices in	• Applicatie-	Wat zijn de opgelegde technische

	de combinatie van technologieën	verwerkingsomgeving • Type database • Gegevensbenadering • Webtechnologie • Besturingssysteem • Gebruik van systeemsoftware • Programmeertaal • Ontwikkeltool	beperkingen naast de beperkingen die eventueel door de klant zijn opgelegd?
3	3. Overzicht van de aanwezige expertise	• Applicatie-verwerkingsomgeving • Type database • Gegevensbenadering • Webtechnologie • Besturingssysteem • Gebruik van systeemsoftware • Programmeertaal • Ontwikkeltool	De lijst met expertise hangt af van de gekozen technische oplossingen. Dit kan rechtstreeks achterhaald aan de hand van de antwoorden uit de vorige stap.
4	4. Overzicht van de mogelijke softwarepakketten binnen GPR	• Gebruik van softwarepakketten	Wat zijn de globale functionaliteiten van het te bouwen systeem?
5	5. Overzicht van de mogelijke standaard componenten voor hergebruik	• Hergebruik van componenten	Wat zijn de specifieke taken van het te bouwen systeem?
6	6. Good practices in de verdeling van modules	• Verdeling van modules • Verdeling van verantwoordelijkheden • Communicatie tussen modules • Dataverwerking	Welke statische kwaliteitseisen zijn gewenst? Op welke systeemprocessen moet deze eisen betrekking hebben?
7	7. Good practices in de verdeling van componenten	• Verdeling van componenten • Communicatie tussen componenten • Locatie van componenten	Welke dynamische kwaliteitseisen zijn gewenst? Op welke systeemprocessen moet deze eisen betrekking hebben?
8	8. Good practices in de combinaties van hardware	• Gebruik van hardware	Welke kwaliteitseisen zijn gewenst? Op welke systeemprocessen moet deze eisen betrekking hebben?

Tabel 2: Omschrijving van de verschillende stappen in de intakefase

In stap 3 van tabel 2 wordt ingegaan op een overzicht van de aanwezige expertise binnen GPR. [Babar 05 b] geeft aan dat kennis op twee manier vast gelegd kan worden:

1. het vastleggen en centraliseren van de inhoudelijke kennis, impliciete kennis expliciet maken
2. het ondersteunen van kennisdeling door vast te leggen wie wat weet

Uit een eerder interview met een projectleider binnen GPR is gebleken dat het inhoudelijk bijhouden van expertise niet te onderhouden is. Er is ooit een poging gedaan, maar deze faalde omdat de technologie te hard ging. De informatie bij de bron was altijd recenter dan de informatie uit het kennissysteem. In dit onderzoek wordt daarom gekozen om de expertise vast te leggen in de vorm van verwijzingen naar personen. Deze verwijzingen zijn gemakkelijker te onderhouden en blijven langer 'houdbaar'. Tevens wordt één van de belangrijkste faalpunten uit kennissystemen voorkomen. [King 02] heeft uitgebreid onderzoek gedaan naar de meest voorkomende oorzaken waarom een kennissysteem faalde in de ondersteuning. Op de derde plek staat het verkeerd interpreteren

van informatie of het overnemen van verouderde informatie vanwege het ontbreken van persoonlijk contact [1]. Dit risico wordt voorkomen door te verwijzen naar de juiste personen voor het vergaren van expertise. Een kennissysteem met verwijzingen naar personen behoort tot de zogenaamde "Knowledge Map Systems (KMS)"[Houari 04]. Houari heeft onderzoek gedaan naar de verschillende soorten kennissystemen en geeft daarbij aan dat KMS gericht is op het verzamelen en vasthouden van individuele expertise. Het systeem bestaat uit een gegevensverzameling van personen en hun specialisme, waarin gebruiker kunnen zoeken en aanpassingen kunnen verrichten. Belangrijk is dat Houari constateert dat een dergelijk systeem het delen van impliciete kennis stimuleert. Het bijhouden van expertise in de vorm van verwijzingen naar personen heeft ook enkele nadelen. [Zhang 06] heeft de organisatorische factoren onderzocht van kennisdeling. Zhang geeft hierbij aan dat het delen van expertise niet altijd gewenst is bij iedereen. Iemand met veel expertise wil niet altijd het aanspreekpunt van de organisatie worden. Daarnaast kan het voorkomen dat het management het kennissysteem gaat zien als een controlemiddel voor de kwaliteit van werknemers. Iemand die niet als expert geregistreerd staat, maar juist veel globale inzichten/vaardigheden heeft, kan soms onterecht als minder waardevol worden gezien. In succesfactor 2 wordt verder ingegaan op de manier waarop expertise wordt gekoppeld aan het beslissingsproces van de architect.

Succesfactor 2: De vragenlijst moet dynamisch zijn

Het dynamisch maken van de vragenlijst wordt mogelijk gemaakt door meer in te spelen op de antwoorden van de architect. Als het beleid van de klant bepalend is, dan wordt er niet uitgebreid in gegaan op de technische vragen. Bij het vragen naar technische beperkingen, in stap 2, moet de architect niet een lange lijst krijgen met vragen die op een statische manier alle technische gebieden aflopen, maar de architect moet de vrijheid krijgen om in een totaal overzicht zelf aan te kunnen geven waar de beperkingen liggen. In stap 4 wordt ingegaan op de bestaande softwarepakketten die ingezet kunnen worden. Deze lijst moet worden gebaseerd op de technische keuzes die al gemaakt zijn in stap 2. Op die manier krijgt de architect een dynamische lijst met softwarepakketten die werkelijk van toepassing zijn en wordt er voorkomen dat er bestaande softwarepakketten worden voorgesteld die niet kunnen integreren in bijvoorbeeld een .NET omgeving (als de architect hiervoor gekozen heeft). Hetzelfde geldt voor het gebruik van standaard componenten, die in stap 5 achterhaald worden. Ook hier moet de architect een lijst krijgen met componenten die nog van toepassing zijn en eventueel ingezet kunnen worden. Alle stappen moeten met elkaar samenwerken en voorkomen dat de architect vragen/mogelijkheden krijgt voorgeschoteld die niet meer relevant zijn. Het moeilijkst is dat voor de stappen 6, 7, en 8 die ingaan op het gebruik van good practices. Hoe kan er nu voor gezorgd worden dat een architect alleen de good practices krijgt voorgeschoteld die werkelijk van toepassing kunnen zijn in het systeem?

In [Babar 05 a] wordt een tool gepresenteerd voor het werken met good practices. In deze paper gebruikt Babar een zogenaamde Utility Tree voor het bladeren naar good practices. Deze boomstructuur is gerangschikt naar kwaliteitsattributen. Onder elk kwaliteitsattribuut staan de good practices opgesomd die van toepassing kunnen zijn. Deze staan omschreven aan de hand van hun scenario's, zodat een architect in één oogslag kan zien waar de good practice betrekking op heeft. [Babar 05 a] gaat echter niet in op het aanbieden van good practices tijdens het beslissingsproces van de architect. In het "toekomstig werk" beschrijft Babar dat het projectgericht toepassen van good practices in de toekomst verder wordt uitgewerkt. Omdat dit onderzoek juist gericht is op de intakefase van het beslissingsproces van een architect, wordt er een eigen voorstel gemaakt voor het projectgericht aanbieden van good practices. In de succesfactoren 5 en 6 wordt verder ingegaan op de vorm van good practices. In deze succesfactor wordt bepaald hoe de good practices gekoppeld kunnen worden aan de antwoorden van de architect. De volgende eigenschappen worden gebruikt:

Betrokken proces(sen)	Communicatie tussen lagen	
Betrokken klant	ZBO	
Criteria	**Technisch beslissingsgebied**	**Criteria**
	(Bijvoorbeeld) Applicatieverwerkingsomgeving	J2EE
	(Bijvoorbeeld) Type database	Oracle

§ ***Betrokken processen.*** Een kwaliteitsattribuut, zoals prestatie, kan op meerdere manier bereikt en daarmee ingezet worden (in de GUI, de database, bij het uitvoeren van bepaalde berekeningen of bijvoorbeeld in het uitwisselen van gegevens tussen componenten). Een belangrijke eigenschap van een patroon is daarom het

[1] De eerste twee oorzaken uit [King 02] zijn: 1) het niet kunnen bieden van voordelen op strategisch niveau en 2) het niet verkrijgen van medewerking van het topmanagement. Deze punten vallen buiten dit onderzoek.

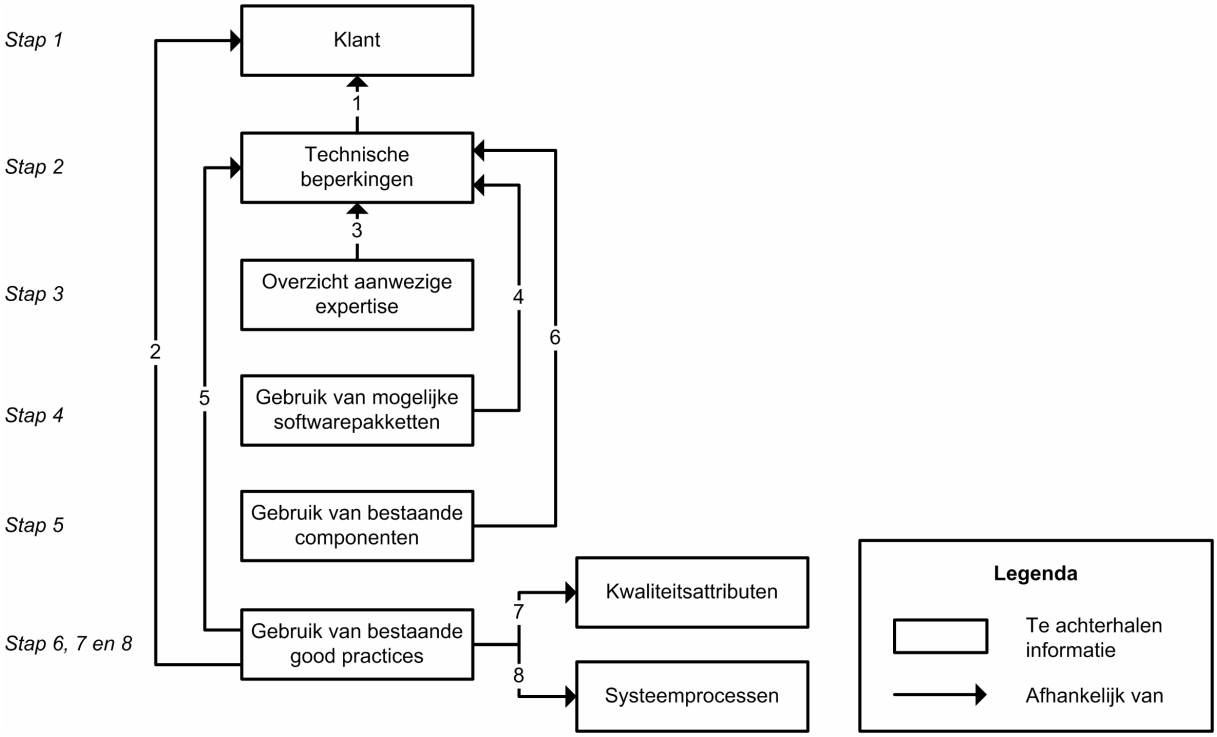
systeemproces waarop het patroon betrekking heeft. Met dit gegeven kan beter bepaald worden of een architect het patroon kan gebruiken of niet.

§ ***Betrokken klant.*** Uit de interviews met betrekking tot de informatiebehoeftes van een architect (hoofdstuk 3) is naar boven gekomen dat het beleid van de klant ook invloed heeft op de verdeling van modules. ZBO schrijft bijvoorbeeld voor dat er in softwaresystemen standaard een 3-laagsarchitectuur gebruikt moet worden. Als de architect in de eerste stap de betrokken klant selecteert, dan kan met deze eigenschap de patronen worden geselecteerd waar de klant veel waarde aan hecht.

§ ***Criteria.*** Uit de interviews met betrekking tot de beslissingsgebieden (hoofdstuk 2) is naar boven gekomen dat het gebruik van patronen (verdeling van modules) beïnvloed wordt door de technische keuzes. Sommige patronen gaan bijvoorbeeld in op een compositie van MS .NET modules en vereisen daarom een MS .NET omgeving. Andere patronen gaan bijvoorbeeld in op de mogelijkheden van Orcale en stellen daarom eisen aan het type database dat gebruikt wordt.

Met deze extra velden en de gerelateerde kwaliteitsattributen van het patroon, kan gecontroleerd worden of het patroon in aanmerking komt om ingezet te worden in het te bouwen systeem.

Figuur 11 geeft schematisch de afhankelijkheden weer tussen de verschillende stappen uit de intakefase. In tabel 4 worden de afhankelijkheden verder toegelicht.



Figuur 11: Afhankelijkheden van de stappen in de intakefase

Relatie	Omschrijving
1	Het beleid van de klant (zoals een referentiearchitectuur) schrijft bepaalde richtlijnen voor over de technische invulling van het systeem.
2	In het beleid van de klant worden in sommige gevallen ook richtlijnen voorgeschreven over het gebruik van bepaalde good-practices. Bijvoorbeeld een 3-laagsarchitectuur.
3	Aanwezige expertise is direct gekoppeld aan de technische invulling van het systeem. Als er wordt gekozen voor een Oracle database dan wil de architect ook weten wie binnen het databaseteam de expert is op het gebied van Oracle en niet op het gebied van MS SQL Server bijvoorbeeld.
4	Bepaalde softwarepakketten kunnen bijvoorbeeld alleen opereren in een J2EE omgeving, omdat ze daarin gebouwd zijn.
5	In de voorgaande omschrijving over het relateren van good-practices is aangegeven dat een good practice afhankelijk kan zijn van technische specificaties.
6	Componenten zijn in een bepaalde omgeving gebouwd (bijv. J2EE of .NET) en zijn daarmee afhankelijk van de technische invulling van het systeem. Een ander voorbeeld is een component dat de

	communicatie met een Oracle database afhandeld. In dat geval heeft het type database veel invloed op de vraag of het component ingezet kan worden of niet.
7	Een good practice (patroon) realiseert één of meerdere kwaliteitsattributen. Een architect maakt een afweging tussen de verschillende kwaliteitsattributen en geeft daarmee automatisch aan welke patronen nuttig kunnen zijn en welke niet.
8	Een kwaliteitsattribuut kan via een patroon op verschillende gebieden worden gerealiseerd (database, GUI, etc). Om deze gebieden te adresseren is een patroon gekoppeld aan een systeemproces. Zo wordt de context en mogelijke toepassing van het patroon specifieker omschreven en kan de architect gemakkelijker aangeven of het patroon ingezet kan worden of niet. Door de architect aan te laten geven welke systeemprocessen van toepassing zijn, kan de context van een patroon vergeleken worden met de context van het te bouwen systeem.

Tabel 4: Omschrijving van de relaties uit Figuur 11

De omschreven afhankelijkheden tussen stappen geven aan dat er ingespeeld kan worden op de antwoorden van de architect, maar niet hoe dit gerealiseerd kan worden en of het wel haalbaar is. Om de afhankelijkheden tussen de verschillende stappen concreter te maken, is er een onderscheid gemaakt tussen de verschillende objecten uit het stappenplan. De relatie tussen de objecten kan het concreets worden weergegeven door gebruik te maken van een datamodel. Figuur 12 toont een strokendiagram met de eigenschappen en relaties van alle objecten. Met dit strokendiagram is het mogelijk om de afhankelijkheden tussen de verschillende stappen (Figuur 11) te realiseren. Bij elke antwoord van een architect is het een kwestie van een query uitvoeren op de database om de mogelijke antwoorden voor de volgende vraag te achterhalen.

Voorbeeld 1:

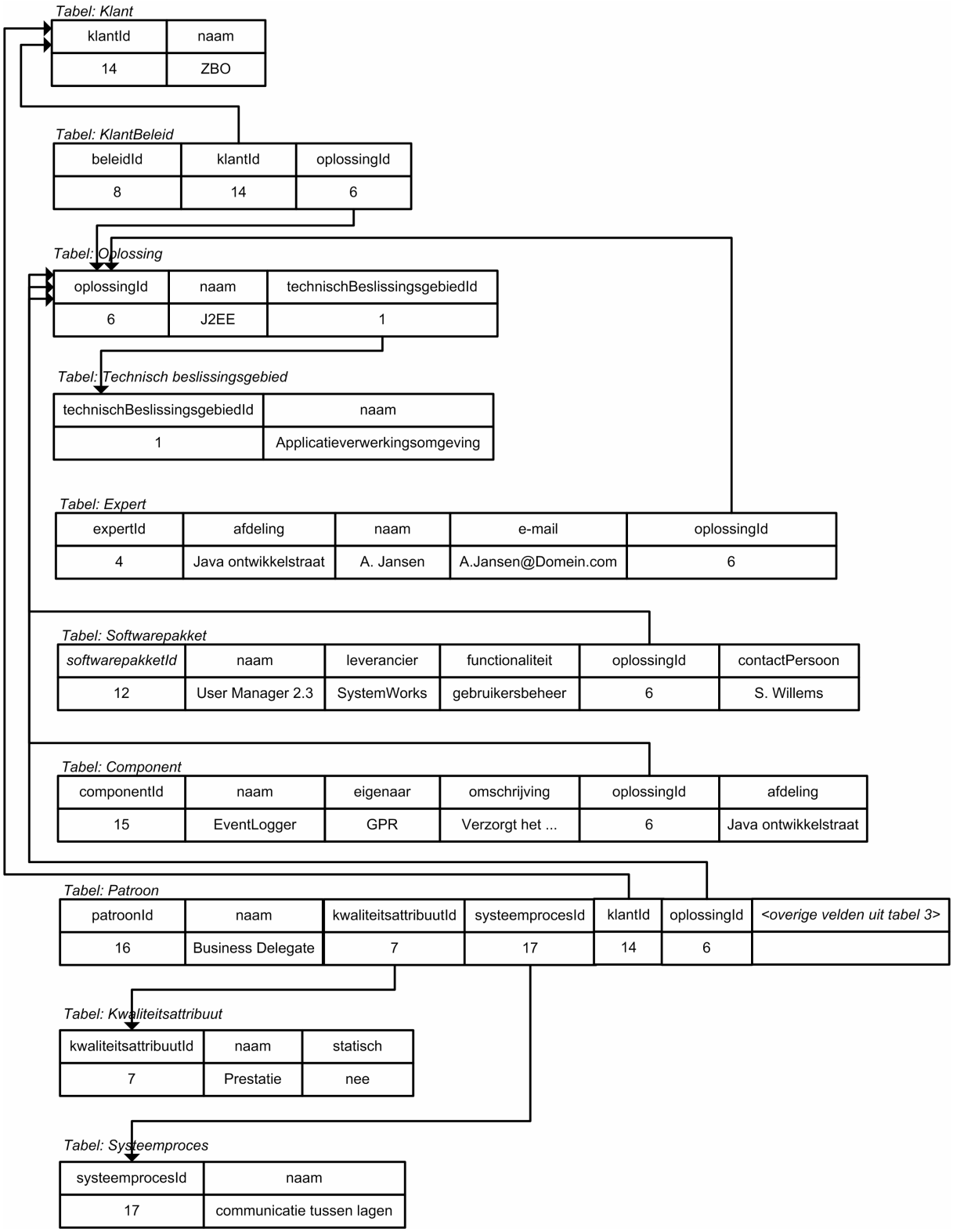
De architect selecteert in stap 2 dat er gekozen is voor een Oracle database (oplossingsId 8) en de programmeertaal C# (oplossingsId 12). Om in stap 4 vervolgens een lijst te genereren met mogelijke softwarepakketten die ingezet kunnen worden kan de volgende query in de kennisdatabase worden uitgevoerd:

```
SELECT functionaliteit FROM SoftwarePakket WHERE oplossingId = 8 OR oplossingId = 12;
```

Voorbeeld 2:

De architect selecteert in stap 1 dat de klant ZBO (klantId 1) betrokken is bij het project. In stap 2 selecteert hij dat er gekozen is voor een Oracle database (oplossingsId 8) en de programmeertaal C# (oplossingsId 12). In stap 6 selecteert hij dat vervangbaarheid (kwaliteitsattribuutId 4) erg belangrijk is. Om nu in stap 6 de systeemprocessen van de mogelijke good practices op te sommen, waaruit de architect een selectie kan maken, kan de volgende query in de kennisdatabase worden uitgevoerd:

```
SELECT naam FROM SysteemProces, Patroon WHERE Patroon.systeemprocesId = SysteemProces.systeemprocesId AND Patroon.kwaliteitsattribuutId = 4 AND (Patroon.klantId = 1 OR Patroon.oplossingId = 8 OR Patroon.oplossingId = 12);
```



Figuur 12: Datamodel (strokendiagram) van de relaties tussen de verschillende objecten in de intakefase

Succesfactor 3: Het soort beslissingen is voornamelijk gericht op de structuur en het gedrag van het systeem
en
Succesfactor 4: De beslissingsgebieden "verdeling van modules" en "communicatie tussen modules" zijn het lastigst.

Deze succesfactoren geven aan dat de architecten zich voornamelijk bezig houden met de structuur en het gedrag van het systeem en dat daarbinnen de verdeling en communicatie tussen modules het lastigst is. Omdat deze succesfactoren allebei aangeven ingaan op de structuur en het gedrag van het systeem zullen de factoren samen worden besproken. Uit hoofdstuk 3 is gebleken dat good practices hierin veel ondersteuning kunnen bieden. Wat zijn de bewezen oplossingen uit het verleden en wat voor gevolgen hebben deze oplossingen gehad? Het feit of een bepaalde oplossing werkt, in dit geval een modulaire verdeling van het systeem, is een zeer belangrijk leermoment voor de architect. Het resultaat van een bepaalde oplossing is belangrijke informatie om te delen. Zo kan aan de ene kant voorkomen worden dat andere architect dezelfde fouten maken en aan de andere kant kan een bewezen succesvolle oplossing worden meegewogen in een volgend project. De ervaring van een architect bestaat voor een groot deel uit het weten welke oplossingen mogelijk zijn en, nog veel belangrijker, wat de gevolgen zijn per oplossing. Die twee gegevens bepalen voor een groot deel de kwaliteit van de afweging die een architect maakt tussen de verschillende alternatieven. Door deze gegevens, de mogelijke oplossingen en de gevolgen, te delen met andere architecten, kan een architect gebruik maken van alle opgedane ervaringen in de organisatie en zo een betere afweging maken tussen alternatieven.

Welke informatie moet in de intakefase nu achterhaald worden om te kunnen controleren of een bestaande good practices ingezet kan worden? Omdat deze succesfactoren ingaan op de structuur en het gedrag van het systeem en de beslissingsgebieden "verdeling van modules" en "communicatie tussen modules" wordt een good practice gezien als een patroon van modules (pattern). Een patroon biedt een bewezen oplossing voor een bepaald probleem en gaat in op één of meerdere kwaliteitsattributen. Het probleem van kwaliteitsattributen is dat ze lastig hergebruikt kunnen worden. Het attribuut prestatie of beveiliging zegt niks op zich, maar krijgt pas een betekenis in een bepaalde context. Het feit dat een kwaliteitsattribuut alleen een betekenis heeft in een bepaalde context, maakt het lastig om de realisatie van een kwaliteitsattribuut (een patroon) te hergebruiken. Er moet gecontroleerd worden of de context van de bewezen oplossing overeenkomt met de context van het te bouwen systeem. Een veelgebruikt hulpmiddel om kwaliteitsattributen concreet te maken en de context te beschrijven zijn kwaliteitsattribuut scenario's [Bass 03 p.71]. Kwaliteitsattribuut scenario's beschrijven een systeeminteractie en leggen daarbij het verband tussen een oplossing, een kwaliteitsattribuut en de context. Er wordt beschreven welk proces het probleem vormt, wie het proces veroorzaakt, in welke toestand het proces optreedt en hoe het systeem dient te reageren om het probleem te verhelpen. Doordat hiermee de good practice concreet wordt gemaakt, is deze gemakkelijker in te zetten in andere projecten. Het oplossingsgedeelte van de kwaliteitsattribuut scenario's (de reactie van het systeem) wordt in [Bass 03] niet verder besproken, hoewel dit juist een zeer belangrijk onderdeel is voor de architect. Het gebruik van scenario's helpt bij het vastleggen en concretiseren van good practices, maar is niet voldoende. Bij maken van een beslissing over een probleem heeft de architect vaak een aantal alternatieven. Tussen deze alternatieven wordt een afweging (trade-off) gemaakt waarbij wordt gekeken welke oplossing het beste aansluit op de wensen van de klant. De architect wil in dit proces weten wat de **gevolgen** zijn van alle oplossingen. De keuze voor een oplossing heeft gevolgen op lang termijn en de architect wil voorkomen dat bepaalde consequenties op de kwaliteit van het systeem in de afweging over het hoofd worden gezien. Door de gevolgen tevens gestructureerd vast te leggen, kan er een feedbackcirkel gecreëerd worden waarin de gevolgen van elk patroon telkens beter beschreven worden naar mate de patronen meer gebruikt worden.

[Woods 05] stelt een manier voor om scenario's meer te voorzien van extra informatie in de vorm van een advies waarin beschreven staat wat een architect dient te weten en waar op gelet moet worden, zogenaamde perspectives. Het grote nadeel van [Woods 05] is dat dit advies geen formele notatie heeft en dat het niet gebruikt kan worden voor automatische assistentie bij het maken van beslissingen. Woods geeft aan dat het toepassen van perspectives nog altijd erg afhankelijk is van de ervaring van de architect. [Babar 04] gaat verder in op het formeel vastleggen van good practices. Hij heeft na onderzoek aangetoond dat de huidige methodes voor het bewaren van architectuuroplossingen de relatie tussen scenario's, patronen en kwaliteitsattributen onvoldoende ondersteunen. Hierbij geeft hij aan dat dit de hoofdoorzaak is dat patronen niet veel hergebruikt worden door architecten. Volgens Babar is er wel veel onderzoek gedaan naar deze relaties, maar wordt er nooit ingegaan op het systematisch opvangen en documenteren van deze relaties. [Bass 02] heeft bijvoorbeeld via interviews alle veelvoorkomende scenario's omtrent *bruikbaarheid* in kaart gebracht en geclassificeerd. Doormiddel van interviews is gevraagd welke scenario's belangrijk zijn bij het werken aan de bruikbaarheid van het systeem. Elke van deze scenario's is gekoppeld aan een categorie en een patroon waarmee de bruikbaarheid in de context van het scena-

rio bereikt kan worden. Bass beschrijft in zijn onderzoek echter niet een manier om de relatie tussen de scenario's en oplossingen (patronen) te beheren en iteratief te verbeteren.

Babar schrijft in zijn onderzoek de eigenschappen van een patroon en gebruikt hij scenario's als onderdeel van een patroon (zie tabel 3).

Patroonnaam: Business Delegate	**Patroontype:** Architectuurpatroon	
Omschrijving	Dit patroon vermindert de koppeling tussen de verschillende architectuurlagen, door één toegangspoort te maken voor het aanroepen van de service in de laag. Ook wordt caching hierin ondersteund, om de prestaties te verbeteren.	
Context	Een client wordt geconfronteerd met de complexiteit in het omgaan met gedistribueerde componenten.	
Probleemomschrijving	De presentatielaag communiceert direct met de business services. Deze directe interactie maakt het systeem kwetsbaar voor veranderingen in de business services.	
Voorgestelde oplossing	Verminder de koppeling tussen de presentatielaag en de business services. De Business Delegate verbergt de onderliggende implementatiedetails van de business service.	
Factoren die mogelijk een probleem kunnen vormen	De presentatielaag heeft directe toegang nodig tot een business service.	
Beschikbare tactics	Delegate Proxy, Delegate Adapter	
Betrokken kwaliteitsattributen	**Positief**	**Negatief**
	Beheerbaarheid, prestatie, aanpasbaarheid	Begrijpbaarheid (indirecte communicatie), Helderheid (er komt een extra laag in de architectuur).
Betrokken abstracte scenario's	S1: De componenten in de presentatielaag zullen niet worden blootgesteld aan de implementatiedetails van de business service.	
	S2: Het systeem biedt een caching mechanisme om de aanvragen naar de business service te versnellen.	
	S3: Veranderingen in de business service implementatie zal geen impact hebben op de componenten in andere lagen die gebruik maken van de business service.	
Voorbeelden van gebruik	E-commerce portalen, Car-dealer systeem NEDIS, MIP	

Tabel 3: De eigenschappen van een patroon [Babar 04]

Het verschil tussen de bovenstaande patroonbeschrijving en de kwaliteitsattribuut scenario's uit [Bass 03] is dat de patronen de neveneffecten van de oplossing beter beschrijven.

Succesfactor 5: Rekening houden met het beleid van de klant

In de eerste stap van tabel 2 wordt gecontroleerd of het beleid van de klant bekend is in de kennisdatabase van GPR. Als dat het geval is, dan heeft dat grote invloed op de beslissingen van de architect. In elke beslissingsgebied waar een architect mee te maken krijgt (hoofdstuk 2.4, Figuur 8) kan het beleid van de klant bepalend zijn, afhankelijk van hoe uitgebreid het beleid van de klant is. Om een architect hierin te ondersteunen, kan het beleid van alle bekende klanten worden verwerkt in een kennisdatabase (zie Figuur 12 voor een weergave van deze database). Per beslissingsgebied uit hoofdstuk 2.4 kan worden aangegeven wat het beleid van de desbetreffende klant voorschrijft. Heel concreet gezien kan er een tabel worden gemaakt waarin een klant wordt gekoppeld met een oplossing in elke beslissingsgebied. Stap 2 kan dan automatisch worden ingevuld, omdat de technische beperkingen bepaald kunnen worden aan de hand van de informatie in de kennisdatabase. Als de klant bijvoorbeeld de programmeertaal C# voorschrijft, dan kan dat automatisch worden ingevuld als technische beperking. De output van stap 2 zijn een aantal technische beperkingen, of die nu door de architect zijn aangegeven of door het beleid van de klant maakt niet uit.

Voorbeeld:
De klant is ZBO en het beleid van deze klant is ingevoerd in de kennisdatabase. Dit houdt in dat er per beslissingsgebied is aangegeven wat het beleid van de klant voorschrijft. In stap 1 selecteert de architect deze klant. In stap 2 wordt automatisch bepaald dat de programmeertaal C# moet worden gebruikt, omdat dit in de kennisdata-

base staat omschreven bij de klant ZBO. In stap 2 kunnen nu één of meerdere “good practice in de combinatie van technologieën” geleverd worden die aansluiten op de technische keuze C#, bijvoorbeeld:

Programmeertaal: **C#** ß **(technische beperking vanuit de klant)**

Good practice die hierbij past:

Applicatieverwerkingsomgeving:	MS .NET
Type database:	MS SQL Server
Gegevensbenadering:	ADO.NET
Webtechnologie:	ASP.NET
Besturingssysteem:	Windows 2003 Server
Systeemsoftware:	IIS 6, SQL Server 2000, SOAP Runtime, Iexplorer 6.0, .NET framework 1.1
Programmeertaal:	C#
Ontwikkeltool:	MS Visual Studio 2003

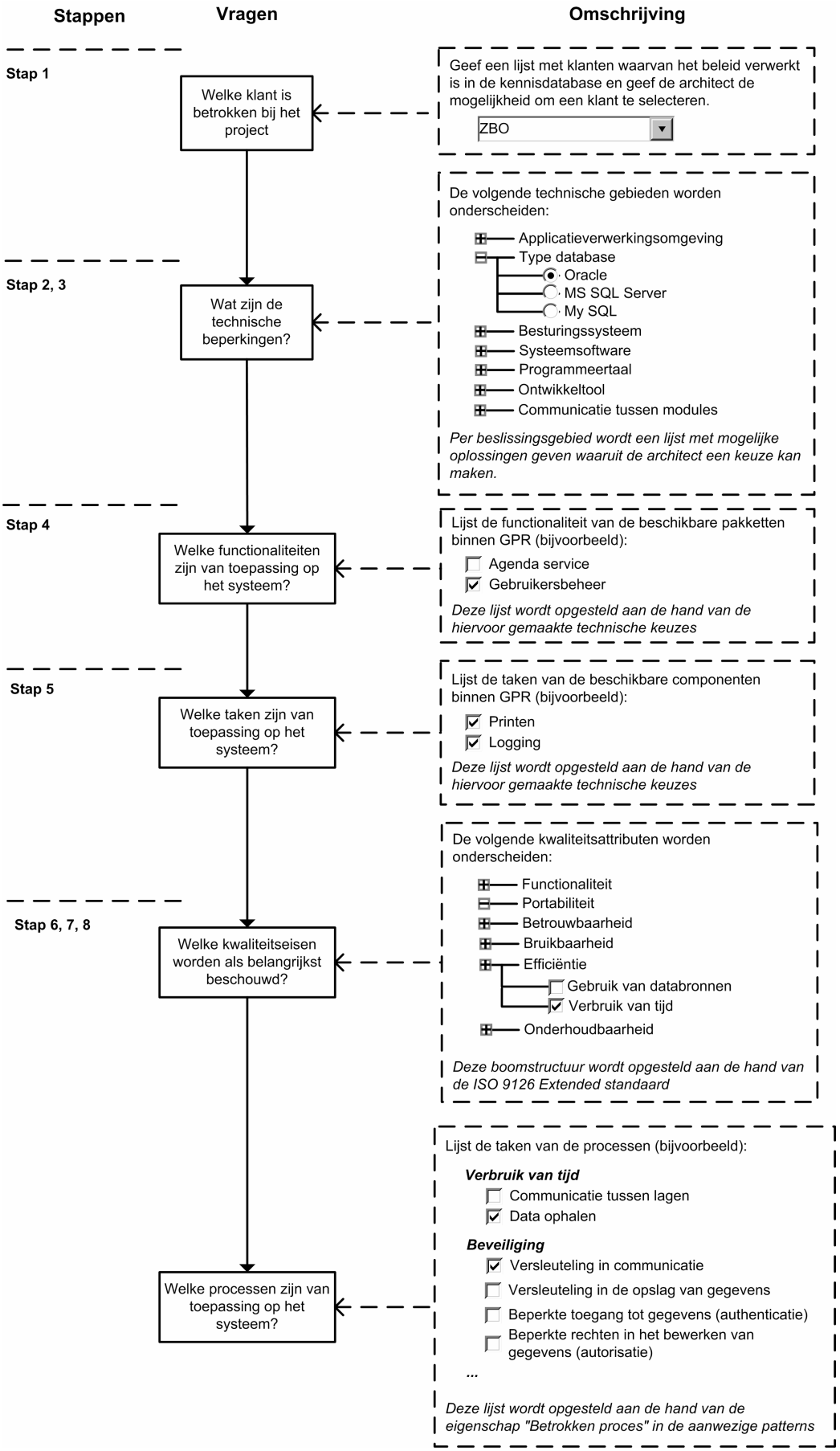
Succesfactor 6: Good practices inzetten op verschillende abstractieniveaus

Zoals in hoofdstuk 3 (informatiebehoeftes) beschreven staat, kan er een onderscheid gemaakt worden tussen dynamische en statische kwaliteitsattributen. Deze twee soorten komen respectievelijk overeen met de good practices in de verdeling modules (designtime) en de verdeling van componenten (runtime). Door deze scheiding ook te verwerken in de ondersteuning wordt het onderscheid tussen de verschillende good practices voor de architect nog duidelijker. In de ondersteunende informatie kan apart worden ingegaan op de verschillende soorten kwaliteitsattributen (dynamische kwaliteitsattributen zoals prestaties, beveiliging en beschikbaarheid en statische kwaliteitsattributen zoals onderhoudbaarheid, bouwbaarheid en time-to-market). De volgende twee succesfactoren gaan dieper is op het hergebruik van good practices en het gebruik van deze kwaliteitsattributen.

4.5. Totaaloverzicht succesfactoren

Alle succesfactoren die hier beschreven staan kunnen verwerkt worden in één implementatie van een verbeterde ondersteuning. Om een beeld te vormen van deze implementatie zijn de stappen uit tabel 2 (succesfactor 1) vertaald naar een mogelijke grafische weergave (Figuur 13). Elke stap is afhankelijk van de voorgaande stap en per stap is in het figuur toegelicht hoe de grafische weergave in die stap tot stand moet komen. Kort samengevat komen alle succesfactoren terug in de implementatie of in het gegenereerde informatiedocument (bijlag G geeft een voorbeeld van dit document):

Succesfactor 1 (aansluiting op het beslissingsproces): Het stappenplan uit tabel 2 is gebaseerd op de bevindingen uit het onderzoek. De voorbeeldimplementatie is weer gebaseerd op het stappenplan en sluit daarmee aan op het beslissingsproces van de architect. Daarnaast wordt in het informatiedocument bij elke stap duidelijk vermeld welke beslissingen en informatiebehoeftes bij de stap betrokken zijn.
Succesfactor 2 (dynamische vragenlijst): De mogelijke antwoorden bij vraag in de implementatie wordt de kennisdatabase geraadpleegd. Het datamodel uit Figuur 12 heeft laten zien hoe deze raadplegingen gerealiseerd kunnen worden.
Succesfactor 3 en 4 (nadruk op structuur en gedrag): De good practices in het informatiedocument zijn gericht op bewezen patronen. Patronen hebben betrekking op de structuur en het gedrag van het systeem en helpen de architect bij het maken van deze beslissingen.
Succesfactor 5 (beleid van de klant): Het datamodel uit Figuur 12 heeft laten zien hoe het beleid van de klant in de database verwerkt kan worden. Als in stap 1 bijvoorbeeld de klant ZBO wordt geselecteerd, dan kan in stap 2 gebruikt worden om alle technische oplossingen automatisch te selecteren die door de klant worden voorgeschreven. Ook de good practices die gegenereerd worden aan de hand van de antwoorden van de architect, kunnen gekoppeld zijn aan een klant (als het beleid van de klant bijvoorbeeld een 3-lagen structuur voorschrijft).
Succesfactor 6 (good practices op verschillende abstractieniveaus): Na het aangeven van de juiste kwaliteitsattributen en systeemprocessen in stap 6, 7 en 8, wordt het informatiedocument gegenereerd. In dit document wordt een duidelijk onderscheid gemaakt tussen de verschillende good practices. Hiervoor is gebruik gemaakt van de nivaus *runtime*, *designtime* en *hardware niveau*.



Figuur 13: Voorbeeldimplementatie

4.6. Evaluatie verbeteradvies

In dit hoofdstuk wordt het voorgestelde verbeteradvies geëvalueerd met Figuur 13 en een fictieve case. Er zal omschreven worden hoe de bevindingen uit het onderzoek een rol spelen bij het doorlopen van de voorbeeldimplementatie met de fictieve case. De uitwerking van de uitgevoerde evaluatie is terug te lezen in bijlage G. Er is gekozen voor een case waarbij alle stappen uit het stappenplan aan bod komen en waarbij de architect over een aantal lastige knelpunten moet beslissen. De case omschrijft een project voor een datawarehouse waarbij alle gegevens van de filialen van de klant samenkomen op één punt. Op deze centrale gegevensverzameling, die wel honderden gigabytes groot kan worden, wil de klant een aantal analyses uitvoeren op zo de statistieken/trends van de verschillende filialen te bekijken. De architect moet een oplossing vinden om een overzicht uit een zeer grote gegevensverzameling binnen 2 minuten bij de eindgebruiker te krijgen. Een dergelijke oplossing bepaald voor een groot gedeelte de kwaliteit van het systeem, omdat het een knelpunt vormt voor de wensen van de klant. De case omschrijft verder nog een aantal eisen van de klant die in een echt project kunnen voorkomen, zoals beveiliging, de keuze voor de database (Oracle) en applicatieverwerkingsomgeving (J2EE). Met deze case kan gecontroleerd worden of de bevindingen uit het onderzoek bruikbaar zijn geweest en of het verbeteradvies kan voldoen aan de informatiebehoeftes van de architect.

De fictieve case is doorlopen met een architect, waarbij de voorbeeldimplementatie als leidraad is gebruikt. De architect heeft aangegeven of de voorbeeldimplementatie aansluit op zijn belevingswereld en of het gegenereerde informatiedocument (zie kopje “Mogelijk gegenereerde informatie voor de architect” in bijlag G) een meerwaarde heeft bij het maken van beslissingen.

4.6.1. Evaluatie met architect

Tijdens de evaluatie met de architect zijn hoofdstuk 4.3 en 4.4 geheel doorgenomen. De succesfactoren zijn door de architect bevestigd en als essentieel gekenmerkt. Tevens heeft hij aangegeven dat deze punten te weinig terugkomen in de huidige ondersteuning. Bij het stappenplan heeft de architect aangegeven dat het veel ondersteuning kan bieden bij het opzetten van applicatie architecturen. Hierbij heeft hij aangemerkt vóór het toepassen van het stappenplan eerst de verschillende subsystemen worden onderscheiden. Het kan zo zijn dat een systeem verdeeld wordt in een front-end en een back-end gedeelte. De front-end kan draaien in een dotNet omgeving, terwijl de back-end draait in een J2EE omgeving. In dat geval kan het stappenplan niet ingezet worden voor het gehele systeem. Het stappenplan kan wel worden toegepast op elke subsysteem, maar de kwaliteitsattributen/richtlijnen die op het gehele systeem gericht zijn, worden dan niet meegenomen. Er moet daarom worden opgemerkt dat er een fase is vóór het stappenplan, de verdeling van subsystemen. Dit is logisch te verklaren door Figuur 10, de relatie tussen de beslissingsgebieden en informatiebehoeftes, uit hoofdstuk 3 erbij te betrekken. Aan de beslissingsgebieden v*erdeling van functionaliteiten* en *communicatie tussen subsystemen* zijn geen informatiebehoeftes gekoppeld omdat deze gebieden zeer afhankelijk zijn van de functionele eisen en de organisatie van de klant. Deze komen daarom ook niet terug in het stappenplan, maar worden wel als eerste beslist.

De integratie en communicatie met andere systemen zijn vaak onderbelicht in projecten, terwijl deze aspecten vaak grote beperkingen in de keuzes kunnen betekenen. Op het gebied van integratie zijn ook verschillende oplossingen te onderscheiden, zoals het component InfoMessaging of de standaard XML. De architect heeft daarom aangegeven om ook het beslissingsgebied *communicatie tussen modules* mee te nemen in de technische beperkingen uit stap 2 (te zien in de voorbeeldimplementatie in bijlage G).

De good practices uit het voorbeeld advies zijn door de architect voorgesteld aan de hand van de kwaliteitsattributen en systeemprocessen. De manier om good practices verder te specificeren via het koppelen van systeemprocessen sloot goed aan op het beeld wat de architect heeft van de huidige good practices binnen GPR.

4.6.2. Evaluatie van de bevindingen uit het onderzoek

Of de bevindingen uit het onderzoek, het beslissingsmodel en de informatiebehoeftes, compleet zijn is zeer lastig te zeggen. Elk softwareontwikkelingsproject is anders en een softwarearchitectuur kan zeer specifieke en uiteenlopende beslissingen bevatten. De compleetheid van de informatiebehoeftes is gewaarborgd, door deze te achterhalen via het generieke beslissingsmodel. In het beslissingsmodel is getracht een zo compleet mogelijk model te maken van de beslissingen die gemaakt worden bij het opzetten van een architectuur. De informatiebehoeftes die hierop gebaseerd zijn hebben dezelfde mate van compleetheid, omdat elk beslissingsgebied betrokken is geweest bij het achterhalen van de informatiebehoeftes.

De bruikbaarheid van de resultaten kan worden aangetoond met het voorbeeld informatiedocument dat vanuit de fictieve case gegenereerd is (zie einde bijlage G). Dit document kan helpen bij het verbeteren van de beslissingen, het voorkomen van valkuilen en het verbeteren van de kwaliteit van het systeem. Als de architect kiest voor een bewezen oplossing uit de kennisdatabase, dan is hij zich bewust van de positieve en negatieve gevolgen van de oplossing. Op die manier kan hij goed inschatten wat voor gevolgen de oplossing heeft op de wensen van de klant en de kwaliteit van het systeem, met een betere afweging tot gevolg. In de analyse van de informatiebehoeftes is aangetoond dat de architect op dit moment geen overzicht hebben van de expertise binnen GPR en de aanwezige componenten/softwarepakketten in de ontwikkelteams. Dit kan leiden tot veel tijdverlies of een onvolledige afweging omdat de architect niet alle specificaties kan achterhalen van een bepaalde technische oplossing. Met de gegenereerde informatie kan de architect snel toegang krijgen tot de expertise binnen GPR die van toepassing is op het project en krijgt hij een overzicht van de componenten/softwarepakketten die aansluiten op de systeemkenmerken van het project. Samengevat kan de gegenereerde informatie uit de fictieve case de kwaliteit van het systeem verbeteren en tijd besparen. Omdat dit het doel is van een ondersteuning in het beslissingsproces kan gesteld worden dat de gevonden informatiebehoeftes uit het onderzoek correct zijn, ze voldoen aan het algemene doel van een verbeterde ondersteuning.

Hoewel de informatie die gegenereerd wordt voor de architect bruikbaar kan zijn om de kwaliteit van de ondersteuning en het beslissingsproces te verbeteren, hoeft dit nog niet het doel van het onderzoek te vervullen. Het doel was om een verbetering in de ondersteuning voor te stellen, zodat de ondersteuning zou **aansluiten op het beslissingsproces en de informatiebehoeftes van de architecten binnen GPR**. Op die manier zou de ondersteuning meer draagvlak krijgen en wordt automatisch de feedbackloop voor architectuurkennis in werking gezet.

De correctheid van onderzoeksresultaten kan in dit onderzoek niet worden aangetoond met de literatuur. In de beginfase van het onderzoek is aangetoond dat veel oplossingen uit de literatuur binnen GPR niet hebben gewerkt, omdat ze niet goed aansloten bij de architecten. In dit onderzoek is daarom erg veel aandacht besteed aan het achterhalen van de beslissingen en informatiebehoeften van de architecten. Hier heeft de nadruk op gelegen in het onderzoek. De bevindingen zijn in hoofdstuk 4.4 verwerkt tot een oplossing en op die manier gevalideerd in dit hoofdstuk. De correctheid van de bevindingen zijn in deze evaluatie bevestigd door één ervaren architect. De oplossing in het onderzoek zou sterker geëvalueerd kunnen worden door meer architecten te betrekken bij de evaluatie.

4.7. Conclusie

Vanuit de analyses uit hoofdstuk 1.4, 2 en 3 zijn zes succesfactoren opgesteld die aangeven waar in de ondersteuning rekening mee gehouden moet worden om deze aan te laten sluiten op het beslissingsproces en de informatiebehoeftes van een architect. Per succesfactor is aangegeven hoe deze te realiseren is in een nieuwe situatie. De gevonden beslissingsgebieden en informatiebehoeftes zijn gebruikt als leidraad in het stappenplan. De afhankelijkheden tussen de verschillende beslissingen uit het ontwikkelde beslissingsmodel uit hoofdstuk 2.4 zijn verwerkt tot schematisch model dat de afhankelijkheden tussen de verschillende stappen laat zien. Het schematische model is concreet gemaakt met een datamodel. Dit datamodel laat zien hoe de intakefase dynamisch gemaakt kan worden, zodat er wordt ingespeeld op de keuzes van de architect. Voor de realisatie van de ondersteuning in good practices is een template voorgesteld. In dit template zijn een aantal velden verwerkt waarmee good practices in de intakefase gekoppeld kunnen worden aan de situatie van de architect. In het hoofdstuk is een voorbeeld gegeven van een mogelijke implementatie van de intakefase. Ter evaluatie van de voorgestelde oplossing is een fictieve case toegepast en doorlopen met een architect. De architect stond zeer positief tegenover de oplossing en heeft een aantal nuttige feedbackpunten gegeven, welke zijn verwerkt in de oplossing.

5. Conclusie

5.1. Contributies

Contributies op academisch gebied
De bevindingen uit het onderzoek, zoals de gevonden beslissingsgebieden, afhankelijkheden, informatiebehoeftes, succesfactoren en oplossing zijn onderzocht in de organisatie van GPR, maar zijn ook algemeen inzetbaar. De beslissingsgebieden geven de onderwerpen aan die aan bod komen bij het ontwerpen van een systeem. Deze beslissingsgebieden komen in elk systeem voor, of deze nu door een architect bepaald worden of door een ontwerper/ontwikkelaar. De informatiebehoeftes zijn deels achterhaald via interviews met diverse architecten en deels met behulp van de literatuur. Uit de literatuur is een referentiemodel gebruikt en zijn verschillende bronnen gebruikt die onderzoek hebben gedaan naar het ondersteunen van de architect in de informatie die nodig is bij het maken van beslissingen. Het feit dat de bevindingen in het onderzoek gebaseerd zijn op de huidige literatuur én de constateringen in een grote organisatie, heeft veel waarde op het academische gebied. Er is verder gewerkt op bestaand werk, en de hierop gebaseerde oplossingen zijn getoetst in de praktijk.

Het ondersteunen van een architect bij het nemen van beslissingen in een zeer recent onderzoeksgebied wat nog lang niet is uitgekristalliseerd. In dit onderzoek is aangetoond dat er nog veel verbeteringen mogelijk zijn in de huidige architectuurondersteuning binnen een grote organisatie als GPR. De opgestelde succesfactoren om een oplossing voor ondersteuning draagvlak te geven bij de bij de architecten en aan te laten sluiten op de beslissingen en informatiebehoeftes van een architect kunnen hulp bieden bij soortgelijke onderzoeken. Elke organisatie heeft vaak net iets andere invulling van de rol van architect, maar dit onderzoek binnen GPR kan altijd gebruikt worden als reflectie voor andere onderzoeken.

Contributies voor GPR
In Figuur 4 uit hoofdstuk 1.4 is aangetoond hoe het wegvallen van architectuurondersteuning kan leiden tot kosten in tijd en geld. In dit onderzoek is een verbetervoorstel opgezet om de huidige ondersteuning beter aan te laten sluiten bij de beslissingen en informatiebehoeftes van een architect. Hiermee kan meer draagvlak worden gecreëerd bij de architecten, zodat de feedbackcirkel van architectuurkennis in werking gesteld kan worden en beslissingen en oplossingen iteratief verbeterd kunnen worden. In het verbetervoorstel is concreet ingegaan op een mogelijke oplossing en een implementatie voor de database en de presentatielaag. Hiermee kan GPR de oplossing rechtstreeks gebruiken in de praktijk. Eén van de doelen van GPR was om zicht te krijgen in het beslissingsproces en de informatiebehoeftes van de architect. Deze onderwerpen zijn uitgebreid in het onderzoek aan bod gekomen. Op deze bevindingen is de oplossing gebouwd, waarbij de nadruk heeft gelegen op een goede aansluiting met de architecten. Dit is het grote verschil tussen de oplossing in dit onderzoek en de oplossingen die in het verleden binnen GPR zijn gelanceerd. In de eerste fase van dit onderzoek is de fout gemaakt om gelijk te werken aan een oplossing, zonder eerst onderzoek te doen naar wát de oplossing zou moeten bieden. Deze fout is dankzij de intensieve begeleiding binnen GPR op tijd voorkomen, zodat er nu een oplossing is ontwikkeld die gebaseerd is op uitgebreid onderzoek binnen de organisatie van GPR.

5.2. Discussie

Het in kaart brengen van de beslissingsgebieden en informatiebehoeftes is veel gebeurd door middel van interviews en bestaande documentatie. Het opstellen van een architectuur is een creatief proces en er kan nooit gegarandeerd worden dat de beslissingsgebieden en informatiebehoeftes overeenkomen met alle architecten en softwareontwikkelingsprojecten binnen de Sector Public van GPR. In dit onderzoek zijn zo veel mogelijk verschillende architect geïnterviewd, maar bij het onderwerp beslissingen en informatiebehoeftes heeft iedereen vaak een ander beeld, waardoor het onderwerp altijd ter discussie kan worden gesteld. Verder is in dit onderzoek gekozen om twee architectuurframeworks te analyseren en ook hierbij geldt dat het niet te garanderen is dat de resultaten hiervan overeenkomen met alle projecten binnen de Sector Public.

5.3. Toekomstig werk

In het onderzoek zijn de verschillende beslissingsgebieden en informatiebehoeftes van de architect in kaart gebracht. Hierbij is voornamelijk ingegaan op de relatie tussen beslissingsgebieden en informatiebehoeftes. In de toekomst kan meer worden ingezoomd op de verschillende keuzes binnen elk beslissingsgebied. Zo kunnen de gevolgen van bepaalde keuzes (op andere keuzes) beter in kaart worden gebracht en kan de architect beter ondersteund worden in de afweging die hij moet maken tussen verschillende alternatieven. [Regli 00] heeft onderzoek gedaan naar de meest voorkomende methodes voor het ondersteunen van ontwerpopties en rationale en maakt daarbij een onderscheid tussen een process-oriented aanpak en een feature-oriented aanpak. In dit onderzoek bij GPR is geconstateerd dat er binnen GPR een process-oriented aanpak wordt toegepast. Deze aanpak wordt gekenmerkt door issues, opties en argumenten tijdens het beslissingsproces. De feature-oriented aanpak gaat meer in op het hergebruik van architectuurelementen in de vorm van product-lines. Volgens [Regli 00] en [Kruchten 04] zijn de afgelopen paar jaren twee methodes succesvol geweest in de proces-orieted aanpak, namelijk Issue-Based Information Sysetm (IBIS)[Lubars 91] en Questions, Options, and Criteria (QOC) [MacLean 91]. IBIS wordt gebruikt voor de weergave/formele notatie van ontwerpopties en rationale. Hierbij wordt elke issue vooraf gegaan aan een vraag. Afhankelijk van het antwoord op de vraag wordt de volgende positie bepaald in het beslissingsproces. Deze volgende positie leidt weer tot andere issues en vragen en wordt bepaald aan de hand van criteriafactoren. IBIS kan in de toekomst gebruikt worden om de afhankelijkheden in de intakefase op een dieper niveau uit te werken. Het verkennen en in kaart brengen van alle mogelijke opties per beslissingsgebied kan gerealiseerd worden met QOC. Deze methode gaat in op een zogenaamde Decision Space Analysis. Hierbij wordt een bepaalde case gebruikt en wordt er gekeken welke concrete beslissingen gemaakt zijn en wat de alternatieven waren. In de relatie tussen deze concrete beslissingen en alternatieven wordt ook gebruik gemaakt van vragen, opties en criteria.

6. Onderzoeksevaluatie

Tijdens de probleemfase van het onderzoek is uitgebreid ingegaan op de rol van de FrameWorkBench en wat de voordelen en nadelen zijn van een ondersteuning in het beslissingsproces van de architect. Omdat het onderwerp van het onderzoek zeer breed is en het probleem niet meteen helder was, heeft het zeer veel tijd gekost om het doel van het onderzoek scherp te krijgen. Dit heeft tot gevolg gehad dat het onderzoek een paar keer veranderd is van richting. De literatuurstudie van het onderzoek was gericht op het verbeteren van de FrameWorkBench door bepaalde beslissingen voor de architect te maken. Dit zou gerealiseerd worden door het architectuurframeworks-jabloon, wat door de FWB gegenereerd wordt, meer inhoud te geven. Welke punten uit een architectuur zijn nog variabel (variabilities) en welke punten zijn generiek en kunnen van te voren ingevuld worden (commonalities). Door deze oplossingsrichting te volgen is de literatuurstudie geëindigd in een bibliografie over variabilities, commonalities en product lines. De 14 papers die hierbij gebruikt waren gingen in op de verschillende fases bij het opzetten van een product line en op het genereren van architectuursjablonen door onderzoek te doen naar de generieke delen van architecturen. Na een aantal voortgangsbesprekingen bleek dit toch niet de juiste richting te zijn. Het was wel een correcte oplossingsrichting, maar het onderzoek diende zich eerst te richten op de informatiebehoeftes van een architect. Veel ontwikkelde oplossingen binnen GPR uit het verleden bleken niet bruikbaar te zijn, omdat deze niet goed aansloten op het beslissingsproces en de informatiebehoeftes van de architect. Ontwikkelde tools werden vaak door de architect als onbruikbaar en overhead gezien voor het beslissingsproces. Vanaf dat punt heeft het onderzoek zich gericht op de intakefase van de huidige ondersteuning. Wat wil een architect precies voor informatie voorgeschoteld krijgen en welke vragen moeten in de intakefase gesteld worden om de architect te voorzien van deze informatie. Om deze nieuwe onderzoeksrichting scherp te krijgen was het noodzakelijk om de rol van de FrameWorkBench scherp te krijgen en te verantwoorden hoe een dergelijke ondersteuning kan bijdragen aan een beter beslissingsproces. Om dit te kunnen bereiken is veel verkennend onderzoek gedaan in de literatuur, waarvan weinig papers terug zijn gekomen in het onderzoek. De 18 papers over kennissystemen en architectuurbeslissingen hebben bijgedragen aan een algemene beeldvorming van het omgaan met kennis en het gebruik van informatie bij het nemen van beslissingen. Uiteindelijk heeft deze beeldvorming geholpen bij het gronden van de geconstateerde problemen in de organisatie van GPR in de probleemanalyse. De gelezen papers zijn alleen samengevat en lang niet allemaal gebruikt in het onderzoek. Enkele papers over kennissystemen zijn gebruikt voor de stap *aanbieden van expertise* in het verbeteradvies. Bijlage H geeft een overzicht van alle literatuur die wel gelezen is, maar niet is teruggekomen in het onderzoek.

6.1. Onderzoeksreflectie

Het doel van dit onderzoek was om een verbeterde ondersteuning voor te stellen die goed aansluit op het beslissingsproces en de informatiebehoeftes van de architecten binnen GPR. Na de probleemanalyse is het onderzoek verdeeld in drie fases: het achterhalen van beslissingen, het achterhalen van informatiebehoeftes en het opstellen van een verbeteradvies. Op deze drie fases zal kort worden gereflecteerd.

Het achterhalen van de beslissingen
Tijdens het achterhalen van de beslissingen die door de architecten gemaakt worden is niet veel gebruik gemaakt van de literatuur, omdat de literatuur geen generiek beslissingsproces met concrete beslissingsgebieden beschrijft die altijd doorlopen worden bij het opzetten van een softwarearchitectuur. Dit heeft geen negatieve gevolgen voor het onderzoek gehad, omdat het doel van het onderzoek gericht was op het expliciet maken van het beslissingsproces binnen GPR. Door gebruik te maken van aanwezige architectuurframeworks en interviews met architecten, kan er met meer zekerheid worden gezegd dat de resultaten goed aansluiten op het beslissingsproces van de architecten binnen GPR.

Het afleggen van interviews met de diverse architecten, om zo de beslissingen bij het opzetten van een architectuur te kunnen achterhalen, bleek niet te werken. Een architect is niet in staat om expliciet aan te geven hoe het beslissingsproces in een project verloopt. Dit heeft wel bevestigd dat de architect geen expliciete structuur aanhouden bij het nemen van architectuurbeslissingen. Ook de literatuur beschrijft geen generiek beslissingsproces met beslissingsgebieden die bij het opzetten van een software architectuur doorlopen worden. Er is daarom gekozen om bestaande architectuurframework te analyseren om zo tot een beslissingsmodel te kunnen komen.

Het ontwikkelen van het beslissingsmodel uit hoofdstuk 2 was een creatief proces waarbij eigen inbreng en interpretatie een belangrijke rol speelde. Dit heeft tot gevolg gehad dat er bij de analyse veel eigen inbreng en interpretatie betrokken werd. Dit maakt de gevonden beslissingen en bevindingen minder onafhankelijk. Voor een sterkere validatie moet het proces van beslissingen achterhalen nog een keer door een andere onderzoeker uitgevoerd worden. Op die manier kan de onafhankelijkheid en betrouwbaarheid van deze aanpak werkelijk getoetst worden. Helaas heeft de planning van dit onderzoek het niet toe gelaten om beide architectuurframeworks nog een keer door een andere onderzoeker te laten analyseren.

Achterhalen informatiebehoeftes
Aan de hand van het beslissingsmodel en een interview met een architect is een lijst met acht informatiebehoeftes opgesteld. De invulling van deze lijst is voor een groot deel bepaald door de antwoorden van de architect. De lijst zou sterker gemaakt kunnen worden door meer architecten te interviewen. Toch heeft deze lijst een grote waarde voor het onderzoek gehad, omdat het doel van het onderzoek was om een ondersteuning voor te stellen die aansluit bij de informatiebehoeftes van de architect. Het is daarom voor het onderzoek niet noodzakelijk dat de lijst met gevonden informatiebehoeftes allesomvattend is, maar dat de lijst wel aansluit op de belevingswereld van de architecten.

Opzetten verbeteradvies
Op basis van de bevindingen in de gevonden beslissingen en informatiebehoeftes is het verbeteradvies opgesteld. Het was lastig om het verbeteradvies te evalueren met de architecten. Het succes van de voorgestelde oplossing kan pas bewezen worden als de nieuwe ondersteuning geïmplementeerd is en gebruikt wordt door de architecten bij het nemen van beslissingen. Een architect wil bijvoorbeeld good practices voorgeschoteld krijgen, waar hij zelf niet aan gedacht zou hebben. Dit is echter zeer lastig te toetsen, omdat de voorgestelde oplossing niet voorzien is van een gevulde kennisdatabase. Het was wel mogelijk om het doel van het onderzoek te valideren, door te controleren of de nieuwe ondersteuning aansluit op het beslissingsproces en de informatiebehoeftes van de architecten binnen GPR. In het verbeteradvies is een voorbeeldimplementatie opgesteld. Op die manier was het mogelijk om de voorgestelde oplossing in concrete vorm met de architect te bespreken en om de verschillende stappen in de ondersteuning te doorlopen met een fictieve case. Hiermee is aangetoond dat de oplossing aansluit op de belevingswereld van de architect en daarmee ook aansluit op het doel van het onderzoek.

6.2. Leerpunten

De onstabiele richting van het onderzoek heeft een aantal oorzaken gehad:

1. De opdracht was zeer breed omschreven
In de opdrachtomschrijving staat de FrameWorkBench centraal. Alle stappen uit de FWB zijn beschreven en bij elke stap is aangegeven welke verbeteringen er eventueel mogelijk zijn. Omdat de FWB betrekking heeft op een zeer breed gebied (van eisen achterhalen tot aan het genereren van een architectuurdocument), was het mogelijk om allerlei oplossingen en doelstellingen te verzinnen. Tijdens de literatuurstudie van het onderzoek is gekozen om de FWB bruikbaarder te maken door de architectuuroplossingen beter te laten genereren door de FWB. Dit bleek achteraf niet de gewenste richting te zijn. Het was beter geweest om in de eerste fase van het onderzoek alle mogelijke oplossingsrichtingen en doelstellingen van het onderzoek op te sommen, om vervolgens in samenwerking met de begeleiders tot één duidelijke richting te komen. Daarbij moet wel gezegd worden dat het achteraf pas duidelijk is hoeveel oplossingsrichtingen er mogelijk waren en wat precies de positie is van het huidige onderzoek in het proces van het opzetten van een architectuur.

2. De problemen waren niet concreet aan te wijzen in de praktijk
Tijdens de probleemfase van het onderzoek heeft het erg veel moeite gekost en de vraag te beantwoorden waarom het onderzoek nu wordt uitgevoerd. Het werkt toch ook zonder de architect te ondersteunen? Het eerste plan van aanpak van het onderzoek is opgezet, voordat de probleemanalyse volledig was uitgevoerd. Het was beter geweest om eerst het probleem scherp te krijgen, om daarna pas bezig te gaan zijn met oplossingen en doelstellingen.

3. Het doel van het onderzoek was niet meteen scherp
Het doel van het onderzoek is in de loop van het onderzoek steeds duidelijker geworden. Ook de opdrachtgever had niet vanaf de eerste dag een scherp doel van het onderzoek. De stelling “een betere ondersteuning voor de architect” is wel een doel, maar veel te globaal om een onderzoek op te kunnen bouwen. Dit had beter gekund door net als in de eerste oorzaak langer stil te staan bij de mogelijke onderzoeksrichtingen, de aanwijsbare problemen en het plan van aanpak.

4. Het onderwerp was zeer complex en abstract
Dit is een oorzaak die niet voorkomen had kunnen worden. Achteraf kan gezegd worden wat precies de plaats van het onderzoek is in het proces van het opzetten van een architectuur. Vóór het onderzoek was dit beeld nog zeer onduidelijk en abstract en was het lastig om altijd op dezelfde lijn te zitten met de begeleiders en architecten. Het opdoen van kennis over alle aspecten en processen omtrent architectuur, heeft een zeer grote waarde gehad voor het leerproces.

5. Het onderzoek ging in op veel impliciete informatie
Impliciete kennis is lastig te achterhalen en te valideren. Het heeft geen zin om een interview te plannen met een architecten en te vragen naar de beslissingen die hij neemt en de informatie die hij daarbij nodig heeft. Ook een architect moet een bestaande case gaan bestuderen om te achterhalen welke beslissingen genomen zijn/worden. Het feit dat er in het onderzoek veel impliciete kennis gebruikt is, maakte het lastig om een overtuigende validatie uit te voeren en om iedereen te overtuigen van de resultaten uit het onderzoek.

Referenties

[Babar 04] M. Babar, 'Scenarios, Quality Attributes, and Patterns: Capturing and Using their Synergistic Relationships for Product Line Architectures', 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004

[Babar 05 a] M. Babar, I. Gorton en R. Jeffery, 'Capturing and Using Software Architecture Knowledge for Architecture-Based Software Development', The 5th International Conference on Quality Software (QSIC) 2005, Melbourne

[Babar 05 b] M. Babar, I. Gorton, R. Jeffery, 'Toward a Framework for Capturing and Using Architecture Design Knowledge', University of New South Wales, Australia, 2005

[Bass 02] L. Bass, B. John, 'Linking Usability to Software Architecture Patterns Through General Scenarios', Journal of Systems and Software, p. 187-197, 2002

[Bass 03] L. Bass, P. Clements en R. Kazman, 'Software Architecture in Practice', Second Edition, Addison Wesley, 2003

[Tyree 05] J. Tyree en A. Akerman, 'Architecture Decisions: Demystifying Architecture', IEEE Software, 2005

[Clements 06] P. Clements, 'View-Oriented Representation of Software Architecture', Presentation, Carnegie Mellon University, SEI, 2006
http://www.cs.up.ac.za/download.php/ISS2006/Clements/ISS2006_Clements.pdf

[Fujihara 97] H. Fujihara, D. Simmons, N. Ellis en R. Shannon, 'Knowledge Conceptualization Tool', IEEE Computer Society, 1997

[Houari 04] N. Houari en B. Far, 'Application of Intelligent Agent Technology for Knowledge Management Integration', 3rd IEEE International Conference on Cognitive Informatics (ICCI'04), 2004

[IEEE 90] 'IEEE Standard Glossary of Software Engineering Terminology', IEEE Std. 610.12, 1990

[IEEE 92] 'IEEE Standard for Software Quality Metrics Methodology', IEEE Std. 1061, 1992

[IEEE 00] 'IEEE Recommended practice for architecture description.', IEEE Std. 1471, 2000

[ISO 91] ISO/IEC TR 9126 (1991). International Organization for Standardization, Geneva. An international standard for quality factors.

[Jansen 05] A. Jansen en J. Bosch, 'Software Architecture as a Set of Architectural Design Decisions', 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005

[Kruchten 04] P. Kruchten, 'An Ontology of Architectural Design Decisions in Software-Intensive Systems', University of British Columbia, 2004

[MacLean 91] A. MacLean, R. Young, V. Bellotti and T. Moran, 'Questions, Options, and Criteria: Elements of Design Space Analysis', Human-Computer Interaction, 6 (3-4). 201-251, 1991

[Lubars 91] M. Lubars, 'Representing Design Dependencies in an Issue-Based Style', IEEE Software, vol. 8, no. 4, pp. 81-89, Jul/Aug, 1991

[MArch 05] Getronics PinkRoccade, 'Methodische Aanpak Architectuur: Werken onder Architectuur', Whitepaper, versie 3.3, 2005. Beschikbaar via http://www.GetronicsPinkRoccade.nl/ > Services > Standaarden > Architectuur

[Money 95] W. Money en J. Harrald, 'The Application of Group Support Systems to Knowledge Acquisition for Disaster Response Planning', 28th Hawaii International Conference on Sys-

tem Sciences (HICSS'95), 1995

[Probst 98] G. Probst, 'Practical Knowledge Management: A Model that Works', Arthur D Little PRISM, Second Quarter 1998

[Regli 00] W. Regli, X. Hu, M. Atwood and W. Sun, 'A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval', Engineering with Computers, Vol. 16, No. 3 – 4, pp. 209-235, 2000

[Woods 05] E. Woods en N. Rozanski, 'Using Architectural Perspectives', 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005

[Zhang 06] J. Zhang, S. Faerman en A. Cresswell, 'The Effect of Organizational/Technological Factors and the Nature of Knowledge on Knowledge Sharing', p. 74a, Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 4, 2006

Bijlage A: Intakevragenlijst van de FWB

Type opdracht

1. Soort project (systeem onwikkeling, audittraject, insourcing, etc)
2. Doorlooptijd (in maanden)
3. Omvang in functiepunten (< 300, tussen 300 en 600, > 600)
4. Projecteisen (wel, matig, niet)
5. Time-to-market (belangrijk of niet)

Organisatie van de opdrachtgever

6. Welke organisatie / klant (ZBO, PinkRoccade, anders)
7. Organisatieonderdeel (gehele bedrijf, specifiek onderdeel)
8. Bedrijfsfunctie (marketing, verkoop/levering, personeelsbeheer, financieel beheer, anders)
9. Impact (primaire, secundaire bedrijfsproces)
10. Informatieoverdracht met derde partijen (ja, nee)
11. SuwiML (ja, nee)
12. Koppelingen met andere systemen (wel, matig, niet)

Proces aspecten

13. Breedte proces (bedrijfsproces, werkproces, processtap)
14. Toepassingsonderdelen (besturend, uitvoerend, beherend)
15. Type verwerking (gevalsverwerking, stapelverwerking)
16. User interface (ja, 50_50, nee)
17. ICT in het proces (adviserend, uitvoerend)
18. Bekendheid van de problematiek (wel, matig, niet)
19. Stabiliteit probleemstelling (wel, matig, niet)
20. Belang onderhoud (ja, nee)

Project

21. ISK diensttype (ontwikkelen, implementeren, infra, onderhouden, etc)
22. Deellevering (wel, matig, niet)
23. Prioritering (wel, matig, niet)

Type technologie

24. Voorkeur (ja/nee met rationale)
25. Voorkeur database (nee, Oracle, SQL server)
26. Voorkeur ontwikkeltool (Coolgen, WebSphere, MS dotNet, etc)
27. Voorkeur platform (nee, Windows, Unix, VMS)
28. Webtechnologie (ja/ nee, zo ja, Intranet, Internet, Extranet)

Type functionaliteit

29. Basisfunctionaliteit (printen, mailing, tekstverwerking, groupware)
30. Portabiliteit tussen HW en OS (ja, nee)
31. Muteren raadplegen (muteren en raadplegen, alleen raadplegen)
32. Nieuwste technologische ontwikkelingen vereist (ja, nee)
33. Aantrekkelijkheid userinterface (belangrijk of niet)

Stabiliteit van de toepassing

34. Correctheid toepassing (impact groot / klein)
35. Beschikbaarheid toepassing (wel / geen eisen)

Vereiste prestaties

36. Intensiteit gebruik systeem (transacties per minuut)
37. Verdeling transacties (gelijk, pieken)
38. Intensiteit gebruik database (verwerking aantal records per minuut)
39. Omvang gegevensverzameling (meer / minder dan 100 MB)
40. Actualiteitseisen (direct, binnen 1 uur, binnen 24 uur, na 24 uur)
41. Benodigde responsetijd
42. Beschikbaarheid gegevens (hoog, laag, geen eisen)

Organisatie en gebruikers

43. Aantal locaties (alle in de organisatie, beperkt aantal)
44. Gebruik van de gegevens (alleen intern, ook andere locaties)
45. Aantal gebruikers (1, <20, <100, <600, >600)
46. Aantal per type gebruiker (aantal raadplegers, aantal taakverwerkers, etc)
47. Aantal concurrent gebruikers (<10, <50, <300, >300)
48. Concurrent gebruikers per type (aantal raadplegers, aantal taakverwerkers, etc)
49. Aantal werkplekken (alle binnen de organisatie, beperkt aantal)
50. Heterogeniteit werkplek (veel varianten, alleen standaard pc's)
51. Type werkplek (vast, mobiel, thuis)
52. Verschillende platforms (ja, nee)
53. Wireless (ja, nee)

Andere systemen

54. Gebruik gegevens andere toepassingen (ja, nee)
55. (netwerk) koppelingen met derde (ja, nee)

Levering en exploitatie

56. Openstelling applicatie (kantooruren, buiten kantooruren, weekend)
57. Benutting openstellingtijden (soort verwerking)
58. Release snelheid (binnen uur, binnen nacht, geen eisen)
59. Release frequentie (1 maal per maand, per kwartaal, lagere frequentie)
60. Release omvang (alle werkplekken, beperkt)
61. Beheer werkplek (PinkRoccade, anders)
62. Verantwoorde voor beschikbaarheid applicatie (PinkRoccade, andere)

Bijlage B: Beslissingen in project ODZ

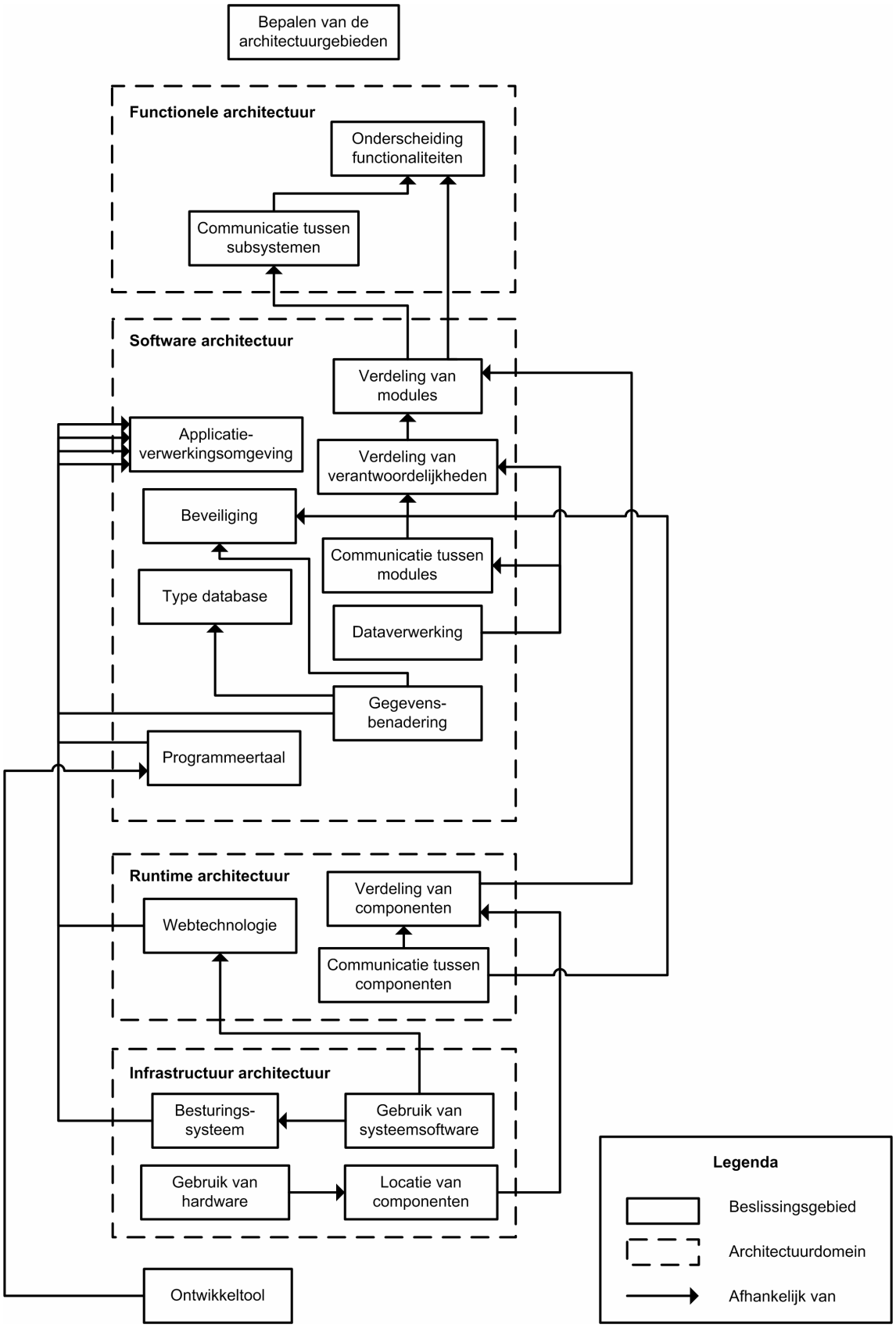
Architectuurframework release 1, versie 1.7

Nr.	Beslissing	Soort	Betrekking op / categorie
	Hoofdstuk Systeemkenmerken		
1.	Toegang tot ODZ wordt door InfoMessaging op applicatieniveau bepaald	2	Beveiliging
2.	Toegang tot individuele gegevens op gebruikersniveau wordt bepaald binnen de aanroepende systemen	2	Beveiliging
	Hoofdstuk Uitgangspunten		
3.	ODZ moet aansluiten bij de architectuurstandaarden van ZBO	5	Gehele systeem
4.	Er mogen geen aanpassingen worden gedaan op de basisregistraties	4	Dataverwerking
5.	Fysieke gegevensopslag vereist	5	Gegevensbenadering
	Hoofdstuk Principes		
6.	ODZ moet benaderbaar zijn via webservices	5	Communicatie tussen subsystemen
7.	Elke 24 uur moeten batches worden gedraaid.	5	Dataverwerking
8.	De informatie in de DB moet gecontroleerd worden op éénduidigheid en consistentie	5	Dataverwerking
	Hoofdstuk Rationales		
9.	Gebruik van InfoMessaging	9	Communicatie tussen modules
10.	Gebruik van MS.Net webservices	9	Applicatieverwerkingsomgeving Communicatie tussen modules
11.	Maximale ontkoppeling gewenst	5	Communicatie tussen modules
12.	Uitwisseling via bestanden mogelijk	6	Communicatie tussen componenten
13.	Verwerking aangeleverde mutatiegegevens binnen één nacht	4	Performance dataverwerking
14.	Gebruik van Oracle 9i (9.2.0.5)	10	Type database
	Contextmodel		
15.	Functionele verdeling van het systeem	1	Onderscheiding van functionaliteiten
16.	ODZ geeft signaal aan Centrale Mutatie Verwerking	2	Communicatie tussen subsystemen
17.	ZBO systemen raadplegen de gegevens van ODZ via webservices en gegevensdiensten	2	Gegevensbenadering Communicatie tussen subsystemen
18.	ODZ systeem verwerkt data en slaat het resultaat op in eigen database	2	Dataverwerking
	Functioneel gegevensmodel		
19.	Ontwikkeling van gegevensdiensten op basis van ZBOML	1	Verdeling van modules

	Functionele subsystemen		
20.	ODZ controleert zelf of er nieuwe mutatiebestanden klaar staan	2	Verdeling van verantwoordelijkheden
21.	Mutatiegegevens worden meerdere malen per dag verwerkt om uitvallen te voorkomen	2	Dataverwerking Realiseren van een kwaliteitseis (beschikbaarheid)
22.	Het inlezen van gegevens gebeurt via tijdelijke tabellen	5	Gegevensbenadering
23.	Database wordt benaderd via aparte modules (gegevensdiensten)	1	Verdeling van modules
24.	Er zijn 6 verschillende gegevensdiensten (zoeken op persoon, opvragen persoon, ..)	1	Verdeling van verantwoordelijkheden
25.	Bij het raadplegen worden mutaties eerst syntactisch gecontroleerd	2	Gegevensbenadering
	Softwarearchitectuur		
26.	3-laags architectuurmodel, om onderhoudbaarheid en flexibiliteit te bereiken	1	Verdeling van modules, realiseren van kwaliteitseisen (onderhoudbaarheid, flexibiliteit, vervangbaarheid)
27.	Mutatiesignalen worden doorgegeven aan de abonnementenservice	2	Verdeling van verantwoordelijkheden
28.	Gebruik van een Mutatie Syntactische Controle component	1	Verdeling van modules Verdeling van verantwoordelijkheden
29.	Gegevensbenadering met ODP.NET en SQL	9	Gegevensbenadering
30.	Gegevensopslag in een Oracle DB, gebruik van indexen en materialized views voor prestatie eis	6	Type database Realiseren van een kwaliteitseis (prestaties)
31.	Toekenning van componenten aan lagen	1	Verdeling van modules
32.	Gebruik van .NET voor de gegevensdiensten, om risico's m.b.t. de opleverdatum te vermijden	9	Applicatieverwerkingsomgeving Realiseren van kwaliteitseis (time-to-market)
33.	De webservice voor de gegevensdiensten worden in VB.NET ontwikkeld	9	Programmeertaal
34.	De verwerkingslogica wordt in VB.NET ontwikkeld	9	Programmeertaal
	Runtimearchitectuur		
35.	Elke gegevensdienst krijgt een eigen executable, zodat ze apart gereleased kunnen worden.	5	Verdeling van componenten Realiseren van kwaliteitseis (vervangbaarheid)
36.	Gebruik van een apart verwerkingscomponent, omdat mutatieberichten functioneel onderscheiden zijn.	1	Verdeling van componenten
37.	VB.NET wordt gebruikt voor het ontwikkelen van de verwerkingslaag	9	Programmeertaal
38.	In de gegevenslaag wordt gebruik gemaakt van ADO.NET	9	Gegevensbenadering
39.	De webserver communiceert via http met de client	2	Communicatie tussen componenten
40.	Het InfoMessaging systeem communiceert via BEA met de client	2	Communicatie tussen componenten
41.	ASP.Net wordt gebruikt in de webservice	9	Webtechnologie
42.	Generieke componenten worden niet gedeeld in dll's	4	Verdeling van componenten

43.	De gegevens en databasetabellen vormen het gemeenschappelijk component.	5	Verdeling van componenten
	Infrastructuur architectuur		
44.	Er wordt gebruik gemaakt van een testdatabase	6	Locatie van componenten
45.	De productieomgeving wordt geplaatst in het centrale rekencentrum van ZBO	7	Locatie van componenten
46.	De computersystemen worden geïmplementeerd op bestaande configuraties	4	Locatie van componenten
47.	Op de Intel applicatieserver draait op Windows 2003, dual CPU, SOAP/http, InfoMessaging 3.1, .NET omgeving	10	Gebruik van hardware OS Gebruik van systeemsoftware
48.	ODZ mutatie bestanden worden op een extra Windows 2003 server gezet	10	Gebruik van hardware OS
49.	De mutatie/controle functionaliteit wordt op een aparte server gezet, zodat de interactie met de applicatie niet vertraagd	5	Realiseren van een kwaliteitseis (prestatie)
50.	Gebruik van Oracle 9i release 2 op een HP-UX 11i computer. Deze combinatie heeft zich bewezen en is bekend terrein binnen GPR.	10	Gebruik van sofware Gebruik van hardware Realiseren van een kwaliteitseis (betrouwbaarheid)
51.	De databaseserver: een dubbele uitvoering van HP-UX 11i, dubbele VPAR van HP Superdome	10	Gebruik van hardware
52.	Diskopslag voor applicatieservers en databaseserver staat op het SAN (Storage Area Network). Dit voor de flexibiliteit van de dataopslag	5	Locatie van componenten Gebruik van hardware Realiseren van een kwaliteitseis (flexibiliteit, uitbreiding)
	Hoofdstuk methode, technieken en hulpmiddelen.		
53.	Gebruik van Windows 2000 en XP, Internet explorer 6.0, Microsoft Visual Studio .NET 2003, ODP.NET	10	OS Ontwikkeltool Gebruik van systeemsoftware

Bijlage C: Conceptueel beslissingsmodel van ODZ



Aan het model is af te lezen dat de architecten een top-down aanpak gebruiken. Er wordt begonnen vanuit eisen en een referentiearchitectuur[2], waarna de architect stap na stap detaillering aanbrengt en voor de juiste technische invulling kiest. Onderstaand worden alle beslissingsgebieden en afhankelijkheden uit het model kort beschreven. Een architect houdt in elke beslissingsgebied altijd rekening met de kwaliteitseisen en eisen van de klant. Dit verband is altijd aanwezig en sterk afhankelijk van het project en wordt daarom niet expliciet beschreven bij alle onderstaande punten.

§ *Bepalen van de architectuurgebieden*
Voordat een architect begint met het opstellen van een architectuur, maakt hij een beslissing over welke architectuurgebieden worden meegenomen bij het opzetten van een architectuur. Deze gebieden bepalen voor een groot gedeelte welke beslissingsgebieden aan bod komen. Als de architect bijvoorbeeld expliciet het gebied Infrastructuur meeneemt, zal er een beslissing gemaakt moeten over de hardware, etc. Als dit gebied niet wordt meegenomen in het framework, dan blijven deze beslissingen open.

§ *Onderscheiding functionaliteiten*
De verantwoordelijkheden binnen het systeem worden verdeeld in subsystemen. In deze fase worden de verschillende lagen van het systeem aangegeven op functioneel gebied en worden de verantwoordelijkheden van alle subsystemen beschreven.

§ *Communicatie tussen subsystemen*
Afhankelijk van de onderscheiding tussen functionaliteiten en subsystemen, wordt de communicatie tussen deze elementen bepaald. Het gaat hierbij om het soort informatie, zoals wijzigingssignalen of raadplegingen. Hierbij wordt niet ingegaan op technische interfaces of iets dergelijks.

§ *Verdeling van modules*
Afhankelijk van de verdeling van subsystemen en de communicatie tussen subsystemen, wordt het systeem gedetailleerder verdeeld in modules. Deze modules vormen de ‘containers’ voor de uiteindelijke broncode. Deze verdeling is nog niet afhankelijk van de technische invulling, maar is sterk afhankelijk van de kwaliteitseisen van de klant. Als vervangbaarheid zeer belangrijk wordt geacht, dan zal de architect bepaalde code moeten bundelen in modules of moeten kiezen voor een compositie waarbij bepaalde modules vervangen kunnen worden zonder dat het gehele systeem daarvan afhankelijk is.

§ *Verdeling van verantwoordelijkheden*
De verdeling van verantwoordelijkheden hangt samen met de verdeling van modules. Afhankelijk van de kwaliteitseisen wordt bepaald wat de taken zijn van de verschillende modules. Als het type database bijvoorbeeld gemakkelijk gewijzigd moet kunnen worden, dan worden alle taken omtrent dataraadpleging gebundeld in één module.

§ *Communicatie tussen modules*
Bij de communicatie tussen modules wordt ingegaan op het soort berichten dat onderling verstuurd wordt. Deze beslissingen hangen daarom sterk af van de verantwoordelijkheden/taken van de verschillende modules. Bij het aangeven van het soort bericht wordt ingegaan op de technische interfaces van de verschillende modules. De onderlinge communicatie hangt daarom ook sterk af van de omgeving van het systemen. Integratie met andere systemen wordt mede bepaald door de technische interfaces van de modules en de communicatieberichten.

§ *Applicatieverwerkingsomgeving*
Met de keuze voor applicatieverwerkingsomgeving (bijv. .NET of J2EE) begint de technische invulling van het systeem. Deze keuze hangt sterk af van referentiearchitectuur van de klant of van de aanwezige expertise binnen de ontwikkelorganisatie (GPR).

§ *Beveiliging*
De beveiliging is op te delen in een aantal gebieden:
3. Versleuteling in communicatie
4. Versleuteling in de opslag van gegevens
5. Toegang tot gegevens (authenticatie)
6. Rechten tot de bewerking van gegevens (authorisatie)
7. Verificatie

[2] De grootste klant van, ZBO, heeft een eigen referentiearchitectuur met principes en richtlijnen. Dit is echter niet bij elke klant van GPR het geval.

De keuze voor het beveiligingsniveau hangt af van het type applicatie (web of niet) en de gegevens waarmee een te bouwen applicatie gaat werken. Als deze gegevens vertrouwelijk zijn, dan zal er gekozen worden voor strikte beveiligingsmaatregelen. De integratie met andere systemen en de vorm van communicatie tussen deze systemen (internet, lokaal, draadloos) kan de graad van beveiliging tevens beïnvloeden.

§ *Type database*
Het type database (bijv. Oracle, MS SQL Server, MySQL) wordt vaak bepaald door de omvang van de gegevensverzameling (Aantal MB, aantal gebruikers, aantal transacties) van het te bouwen systeem of door de referentiearchitectuur van de klant. Soms wordt het type database beïnvloed door de applicatieverwerkingsomgeving. In een MS .NET omgeving wordt bijvoorbeeld vaak gekozen voor een MS SQL Server als database, omdat deze twee goed op elkaar aansluiten. In het project ODZ is gekozen voor de database Oracle. Deze database is niet afhankelijk van de applicatieverwerkingsomgeving.

§ *Dataverwerking*
De keuze voor dataverwerking is sterk afhankelijk van de eisen en de verdeling van verantwoordelijkheden van de modules. In dit beslissingsgebied bepaald een architect hoe en welke data door het systeem loopt en welke modules de data aanpassen.

§ *Gegevensbenadering*
De gegevensbenadering (bijv. ADO.NET, ODP.NET) wordt bepaald door het type database, de applicatieverwerkingsomgeving en de benodigde beveiliging. Met deze drie gegevens bepaald de architect hoe de datalaag voor het opslaan en raadplegen van informatie uit de database eruit gaan zien.

§ *Programmeertaal*
De programmeertaal wordt bepaald door de applicatieverwerkingsomgeving, de referentiearchitectuur van de klant of de aanwezige expertise binnen de ontwikkelorganisatie. Als er gekozen wordt voor de applicatieverwerkingsomgeving MS .NET, dan zijn er verschillende keuzes mogelijk als programmeertaal. In het project ODZ is er gekozen voor de programmeertaal C#, omdat de referentiearchitectuur van ZBO dat voorschrijft.

§ *Webtechnologie*
De webtechnologie (bijv. ASP.NET, PHP, JSP, Java Servlets) wordt bepaald door de applicatieverwerkingsomgeving of door de referentiearchitectuur van de klant. Vaak zijn de mogelijkheden voor webtechnologie zeer beperkt. In een MS .NET omgeving kan er bijvoorbeeld alleen gekozen worden voor ASP.NET.

§ *Verdeling van componenten*
De verdeling van componenten beschrijft waar elk stuk broncode (module) komt te draaien. De kwaliteitseisen van de klant, zoals flexibiliteit, vervangbaarheid en betrouwbaarheid, beïnvloeden de keuzes in dit beslissingsgebied sterk. In het project ODZ werd betrouwbaarheid zeer belangrijk geacht en daarom zijn bepaalde componenten op meerdere plekken gezet.

§ *Communicatie tussen componenten*
In dit beslissingsgebied wordt ingegaan op protocollen (bijv. HTTP, SSL) en het type berichten dat tussen componenten wordt uitgewisseld. Opvallend bij het project ODZ is dat hier weinig aandacht aan is besteed. De communicatie tussen verschillende componenten is niet goed expliciet gemaakt, waardoor het duidelijk is hoe componenten met elkaar communiceren.

§ *Besturingssysteem*
Op meerdere gebieden wordt een besturingssysteem bepaald (afhankelijk van het project). Voor ODZ waren dat:
8. Besturingssysteem voor de database server
9. Besturingssysteem voor de applicatie server
10. Besturingssysteem voor de ontwikkelcomputer
11. Besturingssysteem voor de applicatie

Het besturingssysteem kan bepaald worden door de ontwikkelomgeving. In het geval van een JAVA applicatie of een webapplicatie is er ruimte voor meerdere besturingssystemen. In dat geval bepaalt de klant of leverancier welk besturingssysteem wordt gebruikt.

§ *Gebruik van systeemsoftware*

Het gebruik van systeemsoftware wordt bepaald door de eisen van de klant, de besturingssystemen en in het geval van ODZ de webtechnologie. In een MS .NET omgeving is het bijvoorbeeld noodzakelijk van IIS als webserver wordt gebruikt

§ *Gebruik van hardware*
Het gebruik van hardware hangt sterk af van de contracten met de verschillende leveranciers en de bestaande infrastructuur van de klant. Binnen GPR zijn er bepaalde good practices in combinaties van verschillende types hardware. Deze combinaties worden bijna altijd aangehouden, mits de klant andere wensen heeft.

§ *Locatie van componenten*
De locatie van componenten hangt af van eventuele kwaliteitseisen, de keuze voor hardware en de verdeling van componenten.

§ *Ontwikkeltool*
De ontwikkeltool (bijv. JBuilder 9.0, MS Visual Studio 2005) wordt bepaald door de programmeertaal en de expertise binnen de ontwikkelorganisatie.

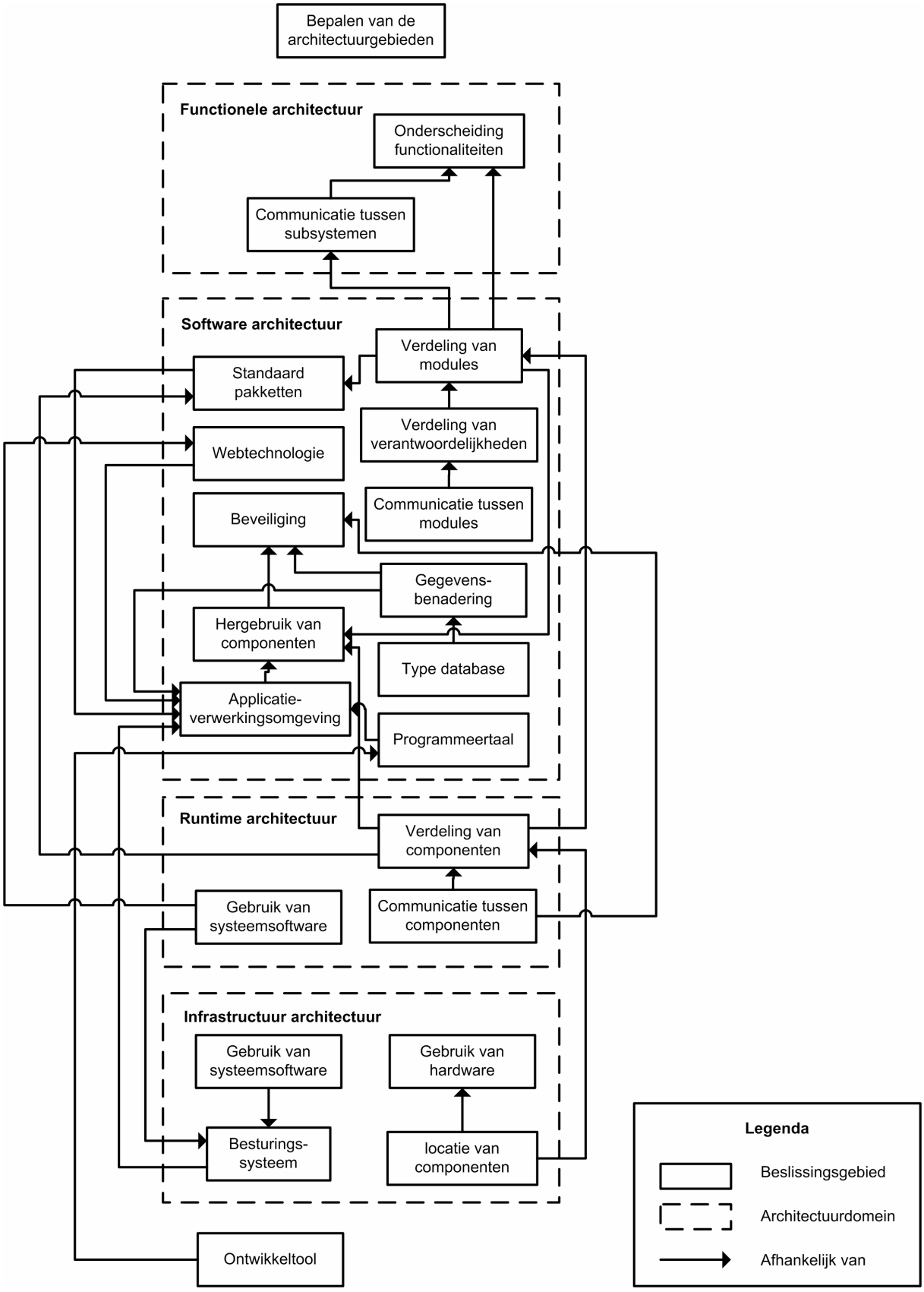
Bijlage D: Beslissingen in project BeFlex

Architectuurframework release 3, versie 1.2

Nr.	Beslissing	Soort	Betrekking op / categorie
	Functionele subsystemen		
1.	Koppelvlak 28 verzorgt de koppeling van de Belastingdienst via het generieke koppelvlak van SI naar BeFlex	1, 2	Communicatie tussen subsystemen
2.	Koppelvlak 59 zorgt voor de foutberichtafhandeling naar SI	1, 2	Communicatie tussen subsystemen
3.	Voor elk ontvangen weekaanlevering zal door BeFlex een bericht naar de polisadministratie worden gestuurd	2	Communicatie tussen subsystemen
4.	Functionele verdeling van het systeem	1	Onderscheiding van functionaliteiten
5.	De presentatielaag start de verwerkingsprocessen en stelt de Week loongegevens online beschikbaar	1, 2	Verdeling van verantwoordelijkheden
6.	Het invoersysteem zorg voor het ontvangen en archiveren van berichten	1, 2	Verdeling van verantwoordelijkheden
7.	Het verwerkingssysteem zorgt voor de voorraadcontrole en het valideren/opslaan van gegevens	1, 2	Verdeling van verantwoordelijkheden
8.	Het uitvoersysteem zorgt voor het verzenden van foutberichten	1, 2	Verdeling van verantwoordelijkheden
9.	Het beheersysteem zorgt voor het toekennen van rechten aan gebruikers	1, 2	Verdeling van verantwoordelijkheden
	Softwarearchitectuur		
10.	De BeFlex Webapplicatie zorgt voor de weergave van de inkijkgegevens in de Browser	1, 2	Verdeling van verantwoordelijkheden
11.	Via HTML zal IE 6.0 gebruikt worden	10	Communicatie tussen componenten Gebruik van systeemsoftware
12.	Het MQ IN component zorgt voor het verwerken van de inkomende berichten.	1, 2	Verdeling van verantwoordelijkheden
13.	MQ IN wordt met IBM WebSphere MQ gebruikt.	9	Verdeling van modules Webtechnologie
14.	MQ UIT vormt de interface tussen de afnemers en de verwerkingslogica	1, 2	Verdeling van verantwoordelijkheden
15.	Het Security component zorgt voor de autorisatie.	1, 2	Verdeling van verantwoordelijkheden
16.	Het Security component kan hergebruikt worden van het ODZ project	6	Beveiliging Hergebruik van bestaande componenten
17.	Een bericht voor BeFlex wordt gearchiveerd in XML formaat	5	Gegevensbenadering
18.	Het component Invoeren Bericht plakt de berichten van de belastingdienst weer aan elkaar.	1, 2	Verdeling van verantwoordelijkheden
19.	Het component Verwerken WA zorgt voor de controle op de binnengekomen berichten	1, 2	Verdeling van verantwoordelijkheden
20.	Het component Verwerken Batch zorgt voor het verwerken van de voorraadcontroles	1, 2	Verdeling van verantwoordelijkheden
21.	Het component Versturen Bericht zorgt voor het versturen van de foutberichten.	1, 2	Verdeling van verantwoordelijkheden

22.	Voor prestatie eisen worden indexen gebruikt in de Oracle database	5	Type database Realiseren van een kwaliteitseis (prestatie)
23.	Het ophalen van data gebeurt via SQL statements	9	Gegevensbenadering
24.	De verwerkingslaag wordt ontwikkeld in C# .NET	9	Applicatieverwerkingsomgeving Programmeertaal
25.	De applicatie is verdeeld in 4 lagen	1	Verdeling van modules
	Runtimearchitectuur		
26.	Client-Server structuur wordt gebruikt	1	Verdeling van componenten
27.	De verwerkingslaag draait op .NET en wordt ontwikkeld in C#	9	Applicatieverwerkingsomgeving Programmeertaal
28.	Internet Explorer wordt gebruikt in de presentatielaag	10	Gebruik van systeemsoftware
29.	Taakomschrijving per runtimcomponent	2	Verdeling van componenten Verdeling van verantwoordelijkheden
30.	Decompositie van de componenten	1	Communicatie tussen componenten
31.	IIS wordt gebruikt als software voor de webserver	10	Gebruik systeemsoftware
32.	De Oracle Data Provider (ODP) wordt gebruikt voor interactie met de DB	9	Gegevensbenadering
33.	ADO.NET wordt gebruikt voor de communicatie met Oracle Data Provider	9	Gegevensbenadering
	Infrastructuur architectuur		
34.	Client maakt gebruik van Intel desktop / Windows XP	10	Gebruik van hardware OS Gebruik van systeemsoftware
35.	Ontwikkelserver draait op HP-server / HP-UX11i of Intel Server / Windows	10	Gebruik van hardware OS Gebruik van systeemsoftware
36.	Ontwikkelclient draait op Intel Desktop / Windows XP	10	Gebruik van hardware OS Gebruik van systeemsoftware
37.	Web- en applicatieserver draait op Intel Desktop / Windows 2003	10	Gebruik van hardware OS Gebruik van systeemsoftware
	Hoofdstuk Technieken en tools.		
38.	Ontwikkelomgeving client bestaat uit Visual Studio .NET (C#) en ODP.NET 9.2.0.4	10	Ontwikkeltool Gebruik van systeemsoftware
39.	Voor de databaseserver wordt gebruik gemaakt van Oracle RDBMS 9i (9.2.0.5)	10	Gebruik van systeemsoftware
40.	Op de web- en applicatieserver draaien IIS 6, .NET framework 1.1, ODP.NET 9.2.0.4, Info-Messaging Client, BEA MessageQ en IBM WebSphere MQ	10	Gebruik van systeemsoftware

Bijlage E: Conceptueel beslissingsmodel BeFlex



Bijlage F: Informatiebehoeftes

1. Overzicht van de raakvlakken tussen het project en het beleid van de klant

Verklaring en koppeling aan beslissingsgebieden

Uit de interviews met architecten is naar boven gekomen dat veel van de bovenstaande informatiebehoeftes sterk beïnvloed worden door het beleid van de klant. Als de klant voorschrijft om de programmeertaal C# te gebruiken, dan is de architect beperkt in de vrijheid bij het nemen van beslissingen. De programmeertaal bepaalt de applicatieverwerkingsomgeving en verplicht de architect om een MS .NET omgeving te gebruiken. Deze applicatieverwerkingsomgeving beperkt vervolgens weer de keuze voor hergebruik van componenten, de gegevensbenadering, webtechnologie, besturingssysteem, etc. Als de klant een eigen beleid heeft wil de architect weten waar het beleid invloed heeft op het project en waar de beperkingen liggen in het nemen van beslissingen.

Informatievergaring op dit moment

Op dit moment wordt vaak de referentiearchitectuur van de belangrijkste klant geraadpleegd. Deze referentiearchitectuur beschrijft richtlijnen op allerlei architectuur domeinen (processen, organisatie, productbeschrijvingen, software, infrastructuur) en het kost de architect daarom veel tijd om uit te zoeken hoe een project kan gaan aansluiten op het beleid van de klant.

Afhankelijkheden / criteria

Om aan te kunnen geven waar het beleid van de klant het project raakt, dient achterhaald te worden om wat voor type project het gaat. Een systeemontwikkeling project heeft tenslotte andere richtlijnen dan een consultancy project. Aangezien de scope van dit onderzoek ligt op systeemontwikkeling projecten, is het voldoende om te weten **welke klant** bij het project betrokken is.

2. Good practices in de combinaties van technologieën

Verklaring en koppeling aan beslissingsgebieden

Uit het ontwikkelde beslissingsmodel is een sterke afhankelijkheid te ontdekken tussen de volgende beslissingsgebieden:

1. Applicatieverwerkingsomgeving
2. Webtechnologie
3. Type database
4. Gegevensbenadering
5. Besturingssysteem
6. Gebruik van systeemsoftware
7. Programmeertaal
8. Ontwikkeltool

Deze beslissingsgebieden hebben betrekking op de technische invulling van het systeem. Een architect wil bij het nemen van de technische beslissingen graag weten welke combinaties goed of slecht werken en welke beslissingen aan de hand van één beslissing automatisch bepaald kunnen worden. Uit de interviews met de architecten blijkt dat er zeer weinig ruimte is om uiteenlopende beslissingen te nemen in de bovenstaande beslissingsgebieden. Als een architect voor .NET of J2EE kiest als applicatieverwerkingsomgeving, dan heeft hij behoefte aan een advies waarin duidelijk wordt welke combinaties mogelijk zijn met de overige beslissingsgebieden.

Informatievergaring op dit moment

Een groot deel van de technische oplossingen wordt rechtstreeks uit de referentiearchitectuur van de klant gehaald. Zoals gezegd is hier niet veel ruimte in. Als de klant geen voorkeur heeft dan bepaald de architect zelf de combinatie van technologieën. Dit gebeurt naar eigen inzicht en algemene kennis van de architect. In bijna alle gevallen wordt dan gekozen voor .NET, ASP.NET, MS SQL Server, ADO.NET, Windows, IIS, IExplorer 6.0, C# en Visual Studio 2003. Heel veel standaard combinaties en voor de architect ‘logische’ keuzes.

Afhankelijkheden / criteria

Om een architect good practices in combinaties van technologieën aan te kunnen bieden, is het belangrijk om te weten of er in het project **beperkingen zijn aan de technische invulling** van het systeem (zoals een referentiearchitectuur of aanwezige expertise in de ontwikkelteams). Een architect moet kunnen aangeven welke technologie voor het project al vast ligt. Vervolgens kan worden aangegeven, aan de hand van de afhankelijkheden in

het ontwikkelde beslissingsmodel (Figuur 8), welke beslissingen geraakt worden en automatisch bepaald kunnen worden. In Figuur 8 is bijvoorbeeld te zien dat de webtechnologie afhankelijk is van de applicatieverwerkingsomgeving. De applicatieverwerkingsomgeving heeft vervolgens weer invloed op het type database, de gegevensbenadering, het besturingssysteem, het gebruik van systeemsoftware, de programmeertaal en de ontwikkeltool. Als de architect bijvoorbeeld aangeeft dat de klant graag wilt dat er ASP.NET wordt gebruikt als webtechnologie, dan kan er automatisch worden voorgesteld om te kiezen voor de combinatie .NET, ASP.NET, MS SQL Server, ADO.NET, Windows, IIS, IExplorer 6.0, C# en Visual Studio 2003.

3. Overzicht van de aanwezige expertise binnen GPR

Verklaring en koppeling aan beslissingsgebieden

Voordat een architect de technische invulling van het systeem kan bepalen, dient hij/zij op de hoogte te zijn van bepaalde expertise. Hierbij kan gedacht worden aan expertises als:

- Databases (prestaties, voor- en nadelen, nieuwe features)
- Beveiliging (mogelijke technologieën, risico's, vereisten, impact)
- Hardware (prestaties, combinaties van hardware, contracten met leveranciers, specificaties)

De benodigde informatie is sterk afhankelijk van het project en de technische issues waar een architect mee te maken krijgt. Het is daarom niet efficiënt om de kennis inhoudelijk vast te leggen en voor te schotelen aan de architect, maar wel om aan te geven waar bepaalde kennis te vinden. Op die manier kan de architect specifieke kennis opvragen als hij dat nodig heeft, vergt het onderhouden van de aanwezige kennis minder tijd, wordt miscommunicatie tussen de expert en kennisnemer voorkomen (er is persoonlijk contact) en heeft de kennisnemer de garantie dat de meest recente kennis wordt geraadpleegd (er is geen vertraging tussen de bevindingen van de expert en het vastleggen van de inhoudelijke kennis in een kennisbank).

Informatievergaring op dit moment

Elke architect heeft een sociaal netwerk en weet waar bepaalde expertise ligt. GPR heeft een aparte afdeling die zich bezig houdt met database ontwikkelingen. Overige specifieke technische kennis ligt verspreid over de verschillende afdelingen. Op dit moment is er geen lijst met personen/bronnen en de daarbij horende expertise. Uit de interviews met de architecten blijkt dat er veel behoefte is aan een dergelijk overzicht van aanwezige expertise. Naast de personen op de verschillende afdeling, maakt de architect veel gebruik van websites, forums, nieuwsgroepen of onderzoeksinstituten zoals Gartner.

Afhankelijkheden / criteria

Voor het aanbieden van de juiste expertise, is het nodig om te weten aan welke expertise de architect behoefte heeft. Met **welke onderwerpen** krijgt een architect te maken en waar zitten de **lastige punten**.

4. Overzicht van de mogelijke softwarepakketten

Verklaring en koppeling aan beslissingsgebieden

Nadat is bepaald hoe de functionaliteiten binnen het te bouwen systeem globaal verdeeld zijn, gaat de architect nadenken over de invulling van deze functionaliteiten. Deze stap begint met de communicatie tussen subsystemen en de verdeling van modules. Hierbij is het van belang om te weten welke bestaande softwarepakketten (van derde partijen of intern) ingezet kunnen worden in de architectuur. Zo kan een architect bepalen welke functionaliteit automatisch gebundeld en gedekt is, dit dient bekend te zijn voordat de functionele verdeling van het systeem gemaakt is.

Informatievergaring op dit moment

Op dit moment heeft elke architect impliciet kennis van een aantal beschikbare softwarepakketten die ingezet kunnen worden in een te bouwen systeem. Als bepaalde delen van de te bouwen functionaliteit gedekt kunnen worden door één van deze softwarepakketten, dan kan de architect deze verwerken in de architectuur (mits de voordelen groter zijn dan het zelf ontwikkelen van de functionaliteit).

Afhankelijkheden / criteria

Om een architect te ondersteunen bij het gebruiken van bestaande softwarepakketten, moeten de **globale functionaliteiten** van het systeem bekend zijn. Elke feature die ingezet kan worden dekt een bepaalde groep functionaliteiten (bijv. gebruikersbeheer of agenda/berichten service). Er moeten achterhaald worden of de functionaliteiten van het te bouwen systeem overeenkomen met een feature dat ingezet kan worden.

5. Overzicht van de mogelijke standaard componenten voor hergebruik

Verklaring en koppeling aan beslissingsgebieden

Bij de verdeling van modules maakt een architect een keuze in het gebruik van standaard componenten. Hij moet hierbij weten welke componenten binnen de ontwikkelafdeling van GPR beschikbaar zijn en wat het doel van

deze componenten is. Per component moet de architect weten wat de randvoorwaarden en beperkingen zijn om zo te controleren of het componenten in de architectuur past. Omdat het gebruik van standaard componenten invloed heeft op de technische verdeling van het systeem, komt deze informatiebehoefte eerder aan bod dan de good practices in de verdeling van modules.

Informatievergaring op dit moment
De ontwikkelafdeling heeft een lijst met beschrijvingen van de veel gebruikte componenten. Omdat deze lijst niet goed toegankelijk is voor de architecten wordt de lijst niet gebruikt en houdt de architect er geen rekening mee in de architectuur. Een enkel component is bekend bij de architect (database koppelingen, authenticatie module) en wordt dan ook vaak meegenomen in de architectuur.

Afhankelijkheden / criteria
In tegenstelling tot het gebruik van softwarepakketten, is het bij het hergebruik van componenten meestal voldoende om de **applicatieverwerkingsomgeving en de basisfunctionaliteiten (taken)** van het systeem te weten. Een component is tenslotte gebouwd in een bepaalde applicatieverwerkingsomgeving (bijvoorbeeld .NET). Hierbij kan gedacht worden aan printen, e-mail, error logging, gegevensversleuteling. Het aantal opties in deze lijst hangt af van de beschikbare componenten binnen GPR.

6. Good practices in de verdeling van modules
Verklaring en koppeling aan beslissingsgebieden
In het beslissingsgebied “Verdeling van modules” speelt het realiseren van kwaliteitseisen een zeer belangrijke rol. Onbewust past de architect bepaalde patronen toe om de wensen en eisen van de klant te realiseren. Het feit of deze patronen het doel halen of niet is bepalend voor het succes van het project. De architect heeft in dit beslissingsgebied sterk de behoefte aan good practices uit het verleden voor het verdelen van de verschillende modules. Voornamelijk het realiseren van schaalbaarheid speelt een belangrijke rol binnen GPR.

Informatievergaring op dit moment
Een architect haalt op dit moment bepaalde patronen uit de architectuurframeworks van vorige projecten. Vaak zien systemen er globaal hetzelfde uit en kunnen modellen en omschrijvingen uit het verleden hergebruikt worden. In een dergelijk geval wordt de softwarearchitectuur (module architectuur) van het desbetreffende project geraadpleegd.

Afhankelijkheden / criteria
Een good practice in de verdeling van modules realiseert een bepaalde kwaliteitseis. Kwaliteitseisen kunnen verdeeld worden in twee groepen, “Runtime” kwaliteitseisen en “designtime” kwaliteitseisen [Clements 06]. Bij de verdeling van modules gaat het voornamelijk om de statische “designtime” kwaliteitseisen, zoals onderhoudbaarheid, bouwbaarheid en time-to-market. Om de good practices in de verdeling van modules aan te kunnen bieden aan de architect, dient er bekend te zijn welke kwaliteitseisen belangrijk worden gevonden door de klant. Afhankelijk van het aantal aanwezige good practices bij het verdelen van modules binnen GPR, kan een lijst worden opgesteld met gerelateerde kwaliteitseisen. Een architect kan vervolgens aangeven welke eisen van toepassing zijn op het te bouwen systeem. Het achterhalen van deze kwaliteitseisen kan niet door simpelweg een kwaliteitsattribuut zoals Beveiliging als belangrijk aan te vinken. Kwaliteitsattributen worden pas concreet in een context. Voor het kenbaar maken en vinden van kwaliteitseisen kan gebruik worden gemaakt van zogenaamde **kwaliteitsattribuut scenario’s** [Bass 03]. Op die manier kunnen kwaliteitseisen duidelijk omschreven worden, gekoppeld worden aan een kwaliteitsattribuut en gemakkelijk gebruikt worden om good practices te kunnen hergebruiken. Het is de vraag of het mogelijk is om good practices op die manier automatisch aan een architect voor te leggen tijdens het beslissingsproces of dat er een omgeving moet worden gecreëerd waarin de architect gemakkelijk zelf op zoek kan gaan naar good practices op dit gebied.

7. Good practices in de verdeling van componenten
Verklaring en koppeling aan beslissingsgebieden
Als de statische modules van het systeem zijn bepaald, dan worden de dynamische runtime componenten bepaald. Hoe wordt de broncode (modules) verdeeld in componenten, hoe loopt de communicatie tussen de componenten en waar komt elk component te draaien? Hierbij houdt een architect sterk rekening met **kwaliteitseisen**. In deze fase zijn de kwaliteitseisen sterk gericht op de implementatie, prestatie en integratie van het systeem. Als de klant een hoge releasefrequentie wenst, dan wordt de code in kwestie gebundeld in een gemakkelijk vervangbaar component. Bepaalde kwaliteitseisen komen vaak terug in de projecten en een architect past daarom vaak dezelfde patronen toe op component niveau om de kwaliteitseisen van de klant te bereiken. Bij het maken van keuzes in dit beslissingsgebied heeft de architect veel baat bij een overzicht van good practices uit vorige projecten.

Informatievergaring op dit moment
Op dit moment kennen voornamelijk de infrastructuur architecten een aantal patronen op de eisen en wensen van de klant te realiseren. Deze patronen zijn impliciet bekend bij de verschillende architecten en zijn niet op een centrale plek beschreven. Soms gebruikt een architect een patroon uit een vorig project. In dat geval wordt de runtimearchitectuur van het desbetreffende project geraadpleegd.

Afhankelijkheden / criteria
Elke good practice in de verdeling van componenten dekt een bepaalde **kwaliteitseis**. Bij deze beslissingen gaat het voornamelijk om de "Runtime" kwaliteitseisen, zoals prestaties, beveiliging en beschikbaarheid. Aan de hand van de beschikbare good practices binnen GPR kan een lijst worden gegenereerd met mogelijke kwaliteitseisen. Een architect kan dan vervolgens aangeven welke kwaliteitseisen op toepassing zijn van het systeem. Ook hier kan er gebruik worden gemaakt van **kwaliteitsattribuut scenario's** [Bass 03] om de kwaliteitseisen te omschrijven.

8. Good practices in de combinaties van hardware
Verklaring en koppeling aan beslissingsgebieden
In de laatste fase van het beslissingsproces bepaalt de architect de hardwarespecificaties. De infrastructuur architect weet welke type hardware en fabrikanten met elkaar een goede prestatie leveren. Verder is deze architect op de hoogte van de verschillende contracten met leveranciers en de nieuwste ontwikkelingen op het gebied van hardware. Deze combinaties van type hardware worden vaak hergebruikt, maar zijn niet expliciet vastgelegd.

Informatievergaring op dit moment
Een infrastructuurarchitect is gespecialiseerd op hardware en kan daarom relatief gemakkelijk de keuzes maken in het beslissingsgebied Hardware. Andere architecten vragen vaak advies aan de infrastructuurarchitecten voor het bepalen van de juiste hardwarespecificaties.

Afhankelijkheden / criteria
Ook hier dekken de good practices bepaalde **kwaliteitseisen** van de klant. In de lijst met gevonden beslissingen (Bijlage B en C) is te zien dat hardware met name betrekking heeft op prestatie, betrouwbaarheid, flexibiliteit en uitbreidbaarheid. Via de architect moet achterhaald worden welke kwaliteitseisen een belangrijke rol spelen in het project en op basis van die informatie (ook hier met **kwaliteitsattribuut scenario's** [Bass 03]) kunnen de good practices worden aangeboden.

Bijlage G: Evaluatiecase

Toepassing in een fictieve case

In deze fictieve case wordt ervan uit gegaan dat de oplossing geïmplementeerd is en voorzien is van de juiste informatie. De case is samen met een architect doorlopen. Als eerste is de architect bekend gemaakt met de succesfactoren, de verschillende stappen in de intakefase, de afhankelijkheden tussen de stappen en de voorbeelduitwerking uit hoofdstuk 4.3, 4.4 en 4.5. Voordat de case is voorgelegd aan de architect, zijn de antwoorden in de verschillende stappen en de mogelijke ondersteunende informatie met voorbeelden ingevuld. De architect heeft deze tijdens de bespreking deze voorbeelden aangepast of toegevoegd. De good practices in de ondersteunende informatie zijn door de architect aangegeven, aan de hand van de geselecteerde kwaliteitsattributen en systeemprocessen. Het resultaat van de evaluatie is aan het einde van dit hoofdstuk terug te lezen.

De case

De fictieve klant S-Markt heeft GPR de opdracht gegeven om een informatiesysteem te ontwikkelen voor het bijhouden van de statistieken van alle filialen van S-Markt, een zogenaamd datawarehouse. De klant is een bekende van GPR en het beleid van deze klant is verwerkt in de kennisdatabase van GPR. Het doel van het project is om alle gegevens van de filialen te verzamelen in één databron. Aan de hand van deze databron kunnen allerlei overzichten gegenereerd worden, waaruit de business conclusies kan trekken. Voorbeelden van deze overzichten zijn:

- Welk filiaal draait in een bepaalde periode de meeste omzet?
- Welke producten worden in de maand september het meest verkocht?
- Zijn er na bepaalde aanbiedingen trends te ontdekken in de omzet?

Enkele projecteigenschappen zijn:

- Het concern S-Markt heeft 114 filialen.
- Elk filiaal heeft een eigen informatiesysteem dat dagelijks wordt bijgewerkt.
- Het systeem bestaat uit een webapplicatie en wordt benaderd via het Internet.

Eisen die de klant aan het systeem stelt zijn:

- Elk overzicht dient binnen 2 minuten geladen te zijn.
- Een businessgeoriënteerde gebruiker moet zelf zijn overzichten kunnen samenstellen
- De bedrijfsstatistieken bevatten gevoelige informatie en het ophalen en versturen van gegevens dient versleuteld plaats te vinden.
- Het systeem mag alleen gebruikt worden door geautoriseerde mensen.
- De applicatie dient te werken in een J2EE omgeving.
- De database dient ingevuld te worden met Oracle.

De intakefase

De verschillende stappen uit de intakefase zijn doorlopen en de volgende antwoorden zijn gegeven:

Stap 1 (beleid van de klant)
Zoals omschreven is het beleid van de klant bekend bij GPR. In deze stap wordt daarom de klant **S-Markt** geselecteerd.

Stap 2 (technische beperkingen)
Deze stap is automatisch ingevuld aan de hand van het beleid van de klant dat in de kennisdatabase bekend is. In de applicatieverwerkingsomgeving is **J2EE** geselecteerd en het type database is ingevuld met **Oracle**. De architect heeft zelf ter aanvulling aangegeven dat in het project de webtechnologie **JSP** gebruikt gaat worden.

Stap 3 (aanwezige expertise)
In deze stap worden geen vragen gesteld, want de benodigde informatie kan rechtstreeks uit stap 2 worden gehaald.
Stap 4 (functionaliteit van het systeem)
Vanuit de lijst met bestaande softwarepakketten die ingezet kunnen worden in een omgeving met J2EE en Oracle (antwoorden uit de vorige stap) wordt de optie **gebruikersbeheer** geselecteerd.

Stap 5 (taken van het systeem)
Vanuit de lijst met bestaande componenten, die ingezet kunnen worden in een J2EE omgeving, worden de opties **logging** en **printen** gekozen

Stap 6, 7 en 8 (kwaliteitsattributen en systeemprocessen)
Uit de boomstructuur met kwaliteitsattributen worden **verbruik van tijd** (subattribuut van *efficiëntie*) en **beveiliging** (subattribuut van *functionaliteit*) geselecteerd. Aan de hand van deze geselecteerde kwaliteitsattributen wordt gekeken welke good practices van toepassing kunnen zijn. Een good practice is immers gekoppeld aan één of meerdere kwaliteitsattributen. Aan de hand van de gevonden good practices is in deze stap een lijst met mogelijke systeemprocessen getoond. Elke good practice heeft tenslotte betrekking op een bepaald systeemproces. Uit de lijst met systeemprocessen wordt onder het attribuut 'verbruik van tijd' gekozen voor het proces **ophalen van data** en onder het attribuut 'beveiliging' wordt gekozen voor het proces **versleuteling in communicatie**.

Mogelijk gegenereerde informatie voor de architect

Stap 1: Beleid van de klant
(beslissingsgebieden: alle)

De klant ZBO is geselecteerd. Deze klant schrijft het volgende beleid voor:

Technologieën:

Applicatieverwerkingsomgeving:	J2EE
Type database:	Oracle
Gegevensbenadering:	-
Webtechnologie:	-
Besturingssysteem:	-
Systeemsoftware:	-
Programmeertaal:	-
Ontwikkeltool:	-

Zie stap 7 voor de good practices die door de klant zijn voorgeschreven.

Stap 2: Good practices in de combinatie van technologieën
(beslissingsgebieden: applicatieverwerkingsomgeving, type database, gegevensbenadering, webtechnologie, besturingssysteem, gebruik van systeemsoftware, programmeertaal en ontwikkeltool)

Er is aangegeven dat de beperkingen liggen in de applicatieverwerkingsomgeving (J2EE), het type database (Oracle) en de webtechnologie (JSP). De volgende good practices zijn hierbij gevonden:

Good practice 1:

Applicatieverwerkingsomgeving:	J2EE
Type database:	Oracle
Gegevensbenadering:	Oracle Data Provider
Webtechnologie:	JSP
Besturingssysteem:	Red Hat Linux
Systeemsoftware:	Apache, Oracle RDBMS 9i, Red Hat Enterprise Linux v.3, IBM Developer Kit for Linux, Java™ 2 Technology Edition Version 1.4, Eclipse SDK 3.2
Programmeertaal:	JAVA
Ontwikkeltool:	Eclipse 3.2

Good practice 2:

Applicatieverwerkingsomgeving:	J2EE
Type database:	Oracle
Gegevensbenadering:	Oracle Data Provider
Webtechnologie:	JSP
Besturingssysteem:	Windows 2003 Server
Systeemsoftware:	Apache, Oracle RDBMS 9i, JavaTM 2 SDK Standard Edition Version 1.4.2
Programmeertaal:	JAVA
Ontwikkeltool:	JBuilder 10

Stap 3: Aanwezige expertise

(beslissingsgebieden: applicatieverwerkingsomgeving, type database, gegevensbenadering, webtechnologie, besturingssysteem, gebruik van systeemsoftware, programmeertaal en ontwikkeltool)

Aan de hand van de bovenstaande technologieën kunnen de volgende expertises van toepassing zijn:

Expertisegebied	Specialisatie	Afdeling	Persoon	E-mail
Ontwikkelomgeving	J2EE	Java ontwikkelstraat	A. Jansen	A.Jansen@Domein.com
Ontwikkelomgeving	J2EE	Serviceline Architectuur	S. Limmen	S.Limmen@Domein.com
Programmeertaal	Java	Java ontwikkelstraat	C. Hendriks	C.Hendriks@Domein.com
Ontwikkeltool	JBuilder	Java ontwikkelstraat	R. Bergen	R.Bergen@Domein.com
Ontwikkeltool	Eclipse	Java ontwikkelstraat	P. van der Steen	P.vd.Steen@Domein.com
Webtechnologie	JSP	Java ontwikkelstraat	B. Jansen	B.Jansen@Domein.com
Type database	Oracle	Databaseteam	S. Willems	S.Willems@Domein.com
Besturingssysteem	Linux	Serviceline Architectuur	P. Leeuwen	P.Leeuwen@Domein.com
Besturingssysteem	Windows 2003 Server	Serviceline Architectuur	R. Molenaar	R.Molenaar@Domein.com
Systeemsoftware	Apache	Serviceline Architectuur	M. Huisman	M.Huisman@Domein.com
Systeemsoftware	JAVA SDK	Java ontwikkelstraat	W. Winter	W.Winter@Domein.com

Stap 4: Gebruik van bestaande softwarepakketten

(beslissingsgebieden: gebruik van softwarepakketten)

Aan de hand van de omschreven functionaliteiten en technische specificaties kunnen de volgende softwarepakketten worden ingezet:

Software pakket	Leverancier / intern project	Functionaliteit	Criteria	Contactpersoon
User Manager 2.3	SystemWorks	Gebruikersbeheer	Oracle database	S. Willems

Stap 5: Gebruik van standaard componenten

(beslissingsgebieden: hergebruik van componenten)

Aan de hand van de omschreven systeemtaken en technische specificaties kunnen de volgende componenten worden ingezet:

Component	Eigenaar	Omschrijving	Criteria	Afdeling
EventLogger	GPR	Verzorgt het loggen van fouten en gebeurtenissen. Het component behoed het systeem van crashen door via een extra architectuurlaag alle fouten af te handelen.	J2EE	Java ontwikkelstraat
PrintController	GPR	Verzorgt het printen van formulieren, documenten of grafische afbeeldingen.	J2EE	Java ontwikkelstraat

UserAuthentication	Sun	Verzorgt het authenticeren van gebruikers en biedt een grafische interface voor de webapplicatie.	JSP	Serviceline Architectuur

Stap 6: Good practices in de verdeling van modules (statische toestand)

(beslissingsgebieden: verdeling van modules, verdeling van verantwoordelijkheden, communicatie tussen modules, dataverwerking)

-

Stap 7: Good practices in de verdeling van componenten (dynamische toestand)

(beslissingsgebieden: verdeling van componenten, communicatie tussen componenten, locatie van componenten)

Patroonnaam: OLAP	**Patroontype:** Architectuurpatroon	
Omschrijving	Online Analytical Processing (OLAP) staat voor een methode, waarmee managers op een intuïtieve manier antwoord kunnen krijgen op complexe vragen. Door gebruik te maken van OLAP tools, kunnen zij op eenvoudige wijze ad hoc queries samenstellen op een vooraf gedefinieerde gegevensset, zonder dat zij hierbij op de hoogte hoeven te zijn van de structuur van de onderliggende data. Tevens maakt OLAP het mogelijk om de berekeningen en filtering van data op de server uit te laten voeren, om vervolgens alleen het resultaat terug naar de client te sturen.	
Context	Een gebruiker moet de mogelijkheid hebben om queries samen te stellen, zonder daarbij kennis nodig te hebben van de onderliggende datastructuur. De queries hebben betrekking op een grote hoeveelheid data, waardoor er zuinig moet worden omgegaan met de data dat heen en weer gestuurd moet worden tussen de client en de server.	
Probleemomschrijving	De communicatie tussen de datalaag en de presentatielaag verloopt via een netwerkverbinding en de mate van prestatie is sterk afhankelijk van de hoeveelheid data dat wordt verstuurd tussen de client en server. In het geval van grote data aanvragen loopt de prestatie sterk achteruit omdat de berekeningen op de client worden uitgevoerd, waardoor de server vaak grote (overbodige) delen van de gegevensverzameling naar de client moet sturen.	
Voorgestelde oplossing	Gebruik OLAP op de database server. Berekeningen worden zo op de server afgehandeld en alleen het resultaat van een aanvraag of opdracht wordt teruggestuurd naar de client. Op die manier wordt de hoeveelheid data dat verstuurd moet worden geminimaliseerd.	
Factoren die mogelijk een probleem kunnen vormen	- De gebruiker wenst een zeer gedetailleerd niveau te rapporteren, hierdoor kunnen queries erg complex worden. - Een hoog detailniveau in de queries kan het systeem alsnog traag maken.	
Beschikbare tactics	-	
Betrokken kwaliteitsattributen	**Positief**	**Negatief**
	Verbruik van tijd, aanpasbaarheid, beveiliging	Helderheid
Betrokken abstracte scenario's	S1: Een gebruiker van een webapplicatie heeft de mogelijkheid om gegevens de filteren op adres. De gegevens staan op de server en het resultaat van de zoekfunctie wordt getoond in een browser.	
	S2: De webbeheerder van het systeem kan een overzicht uitdraaien van het aantal unieke bezoekers per maand. Hierbij kan hij zelf een bepaalde filter op de gegevens toepassen.	
Voorbeelden van gebruik	Projecten InfoBeheer en PlanningProgram	
Betrokken proces(sen)	Ophalen van data	
Betrokken klant	-	
Criteria	**Technisch beslissingsgebied**	**Criteria**
	-	-

Patroonnaam: SSL beveiliging	**Patroontype:** Architectuurpatroon	
Omschrijving	Deze good practice zorgt voor een beveiligde verbinding tussen systemen. Met de inzet van SSL (Secure Socket Layer) worden gegevens versleuteld kunnen gegevens die verstuurd worden over een netwerkverbinding, niet door derde personen worden gelezen of gewijzigd.	
…	…	
Betrokken kwaliteitsattributen	**Positief**	**Negatief**
	Beveiliging	Prestatie
Betrokken proces(sen)	Versleuteling in communicatie	
Betrokken klant	ZBO	

Stap 8: Good practices in de combinatie van hardware

(beslissingsgebieden: gebruik van hardware)

Patroonnaam: Load sharing	**Patroontype:** Architectuurpatroon	
Omschrijving	Dit patroon zorgt ervoor dat het verwerken van data wordt verdeeld onder meerdere machines. Elke machine doet een deel van het rekenwerk en daarmee kan de data sneller verwerkt worden.	
…	…	
Betrokken kwaliteitsattributen	**Positief**	**Negatief**
	Verbruik van tijd, betrouwbaarheid	Onderhoudbaarheid
Betrokken proces(sen)	Ophalen van data	

Bijlage H: Ongebruikte literatuur

Een uitgebreide bibliografie met de omschrijving per paper kan gevonden worden in het einddocument van de literatuurstudie.

[MAT04] M. Matinlassi, 'Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA', VTT Technical Research Centre of Finland, 2004, blz. 127-136

[BBCDP00] F. Bachmann, L. Bass, G. Chastek, P. Donohoe and F. Peruzzi, 'The Architecture Based Design Method', Software Engineering Institute, January 2000, blz. 9-17

[CLE02] P. Clements and L. Nothrop., 'Software Product-Lines : Practices and Patterns', Addison-Wesley, 2002, 608 pagina's

[FAU01] S.R. Faulk, 'Product-Line Requirements Specification (PRS) : an Approach and Case Study', University of Oregon, 2001, 8 pagina's

[KMHC05] S. Dong Kim, H. Gi Min, J. Her and S. Ho Chang, 'DREAM: A Practical Product-Line Engineering using Model Driven Architecture', Soongsil University, Seoul Korea, 2005, 6 pagina's

[DS99] J.M. DeBaud and K. Schmid, 'A systematic approach to derive the scope of product-lines', Proceedings of the 21st Int. Conf. On Software Engineering, California, USA, May 1999, blz. 34-43

[CHW98] J. Coplien, D. Hoffman and D. Weiss, 'Commonality and Variability in Software Engineering', Bell Labs, 1998, 9 pagina's

[GBS01] J. Gurp, J. Bosch and M. Svahnberg, 'On the Notion of Variability in Software Product Lines', University of Groningen, Blekinge Technical University, 2001, 10 pagina's

[TH02] S. Thiel and A. Hein, 'Modeling and Using Product-Line Variability in Automotive Systems', Robert

[KCL05] S. Dong Kim, S. Ho Chang and H. Jung La, 'Traceability Map: Foundations to Automate for Product Line Engineering', Soongsil University, Seoul Korea, 2005, 8 pagina's

[MY04] M. Moon and K. Yeom, 'An Approach To Developing Core Assets in Product Line', Pusan National University, Republic of Korea, 2004, 3 pagina's

[MYC05] M. Moon, K. Yeom and H. Chae, 'An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line', IEEE Transactions on Software Engineering, vol. 31, no. 7, July 2005, blz. 551-569

[SOMZ05] J. Savolainen, I. Oliver, M. Mannion and H. Zuo, 'Transitioning from Product-Line Requirements to Product-Line Architecture', Glasgow Caledonian University, 2005, 10 pagina's

[DSNB04] S. Deelstra, M. Sinnema, J. Nijhuis and J. Bosch, 'COSVAM: A Technique for Assessing Software Variability in Software Product Families', Department of Computer Science and Mathematics, Rijksuniversiteit Groningen, 2004, 5 pagina's

[Barnum 05] Sean Barnum, Gary McGraw, 'Knowledge for Software Security', *IEEE Security and Privacy*, vol. 03, no. 2, pp. 74-78, Mar/Apr, 2005

[Dai 05] L. Dai en K. Cooper, 'Modeling and Analysis of Non-functional Requirements as Aspects in a UML Based Software Architecture Design', Proceedings of the 6th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, May 23 - 25, 2005, pp. 178-183.

[Dieste 00] O. Dieste, N. Juristo, AM Moreno, J. Pazos, A. Sierra, 'Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends', Handbook of Software Engineering and Knowledge Engineering, 2000

Het gebruik van conceptmodellen in Software Engineering (SE) wordt vergeleken met het gebruik van conceptmodellen in Knowledge Engineering (KE). Kennis kan op verschillende manier gemodelleerd worden:

- *Problem-Solving Methods (PSM)*
- *Ontologies*
- *Knowledge-Based Systems (KBS) development methodologies*

[Freeze 05] Ron Freeze en Uday Kulkarni, 'Knowledge Management Capability Assessment: Validating a Knowledge Assets Measurement Instrument', *hicss*, p. 251a, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 8, 2005

Geeft een criterialijst waarmee het kennisbeheer in een organisatie beoordeeld kan worden. De paper geeft echter geen algemene criteria die op een bestaande organisatie gemeten kunnen worden, maar de paper beschrijft het resultaat van een uitgebreid uitgevoerd onderzoek naar de volwassenheid van het kennisbeheer in een bepaalde organisatie. In dít onderzoek laat de planning het echter niet toe om het verbetervoorstel te toetsen met zo'n uitgebreide toetsing, zoals in deze paper beschreven staat.

[In 01] H. In, R. Kazman en D. Olson, 'From requirements negotiation to software architectural decisions', In Proceedings of the First International Workshop from Software Requirements to Architectures (STRAW'01), 2001

[Kishi 01] Tomoji Kishi, Natsuko Noda en Takuya Katayama, 'Architectural Design for Evolution by Analyzing Requirements on Quality Attributes', *apsec*, p. 111, Eighth Asia-Pacific Software Engineering Conference (APSEC'01), 2001

[Kolp 05] M. Kolp, J. Castro en J. Mylopoulos, 'A Social Organization Perspective on Software Architectures', University of Toronto, 2005

[Lee 01] J. Lee, S. Ha, K. Kang, Y. Choo, Y. Hong en H. Hwang, 'Quality Requirement Elicitation for the Architecture Evaluation of Process Computer Systems', *apsec*, p. 335, Eighth Asia-Pacific Software Engineering Conference (APSEC'01), 2001

Elicitatie proces van de architect:

- *Bepalen van de juiste kwaliteitsattributen*
- *Een scenario kan meerdere kwaliteitseisen beïnvloeden*
- *Na het selecteren van de kwaliteitsattributen dienen alle stakeholders hun opvattingen over de kwaliteitsattributen kenbaar te maken*
- *Het bepalen van de juiste meetwaarden*
- *Het prioriteren van QA scenario's*

Scenario's met informele notatie missen vaak belangrijke elementen om een architectuur te kunnen beoordelen. Bijvoorbeeld scenario's zonder meetwaarde

[Liew 05] A. Liew en D. Sundaram, 'Complex Decision Making Processes: their Modelling and Support', *hicss*, p. 88c, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 3, 2005

[Liu 03] W. Liu en S. Easterbrook, 'Eliciting Architectural Decisions from Requirements using a Rule-based Framework', in Proceedings of the Second International Workshop From Software Requirements to Architectures (STRAW'03), co-located with ICSE 2003, Portland, Oregon, May, 2003

In deze paper wordt een methode voorgesteld om de architect te assisteren bij het opzetten van een architectuur. De paper gaat alleen in op de aanpak die gevolgd moet worden om een architect te kunnen ondersteunen en richt zich daarbij voornamelijk op de requirementsfase. Welke stappen moeten gevolgd worden en welke vragen moeten beantwoord worden? Vragen zoals, welke beslissingen worden regelmatig gemaakt? Hoe staan de beslissingen in verband met de requirements? Hoe kunnen deze verban-

den geclassificeerd worden? Hoe kunnen de beslissingen uit bestaande systemen geabstraheerd worden?

[Nuseibeh 00] B. Nuseibeh and S. M. Easterbrook, 'Requirements Engineering: a Roadmap', presented at Proceedings of the International Conference on Software Engineering ICSE, 2000

Deze paper helpt bij de beeldvorming van de fases in het requirements engineering. De paper beschrijft vijf stappen:

- *elicitatie*
- *modelleren en analyseren van requirements*
- *communiceren van requirements*
- *onderhandelen/overeenkomen tussen requirements*
- *evolueren van requirements*

[Schwanke 05] Robert W. Schwanke, 'GEAR: A Good Enough Architectural Requirements Process', *wicsa*, pp. 57-66, 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005

De GEAR methode beschrijft drie aanpakken voor het architectuur requirements proces, namelijk model-driven, kwaliteitsattribuut scenario's en globale analyse. Deze paper heeft bijgedrage aan de algemene beeldvorming van de requirementsfase van een architect. De drie voorgestelde methodes waren echter lastig te koppelen aan de werkwijze van de architect binnen GPR. Teven is het onderzoek meer gericht op de informatiebehoeftes bij het maken van beslissingen. De requirementsfase heeft ook wel informatiebehoeftes, maar is al doorlopen voordat een architect de beslissingen moet maken.

[Shaaban 01] S. Shaaban, S. Lockley, H. Elkadi, 'Information Visualisation for the Architectural Practice', *iv*, p. 0043, Fifth International Conference on Information Visualisation (IV'01), 2001

Het succes van het weergeven van informatie wordt bepaald door vier factoren:

- *De onderliggende datastructuur*
- *Zoektechnieken*
- *Het identificeren van weergave technieken*
- *Het analyseren van het gebruikersprofiel*

Architectuurinformatie kan in de praktijk verdeeld worden in twee categorieën:
Project gerelateerde informatie en algemene informatie.
Het is aan de architect om deze informatie te combineren tot een betekenisvolle vorm.

Het bouwen van een informatiesysteem kent twee technische problemen:

- *Het toegankelijk maken van informatie*
  - *Het missen van informatie tijdens het filteren van de zoekresultaten.*
  - *Het aanbieden van teveel informatie, waardoor de gebruiker te veel moet lezen voordat hij/zij de relevante informatie heeft gevonden.*
  - *Het is een tijdrovend proces*
  - *Het missen van een juiste interactie met de gebruikerswensen. Het is zeer lastig voor een gebruiker om zijn wensen duidelijk te maken in bijvoorbeeld een query.*
- *De kwaliteit van de weergave van de eindresultaten*

[Sridharan 03] B. Sridharan en Kinshuk, 'Reusable Active Learning System for Improving the Knowledge Retention and Better Knowledge Management', icalt, p. 72, Third IEEE International Conference on Advanced Learning Technologies (ICALT'03), 2003

[Yacoub 00 a] S. Yacoub, H. Xue, H. Ammar, 'POD: A Composition Environment for Pattern-Oriented Design', *tools*, p. 263, Technology of Object-Oriented Languages and Systems (TOOLS 34'00), 2000

[Yacoub 00 b] S. Yacoub, H. Ammar, 'Pattern-Oriented Analysis and Design (POAD): A Structural Composition Approach to Glue Design Patterns', *tools*, p. 273, Technology of Object-Oriented Languages and Systems (TOOLS 34'00), 2000

[Zeng 06] A. Zeng, D. Zheng en H. Peng 'Knowledge Acquisition Based on Rough Set Theory and Principal Component Analysis', IEEE Intelligent Systems archive, Volume 21 , Issue 2, 2006

Rough Set (RS) afhankelijkheden tussen attributen op een database-achtige manier (zoals een beslissingstabel). RS maakt vaak alleen gebruik van afhankelijkheden tussen condities en de eigenschappen van beslissingen. Nieuwe methode in de paper: RS en Principal Component Analysis.