Project Notificatieservices – Bijlage Architectuur

'*work in progress*'

Versie	Datum	Auteur	Toelichting
0.1	23-4-2021	Gerhson Jansen Jeanot Bijpost Ad Gerrits	Eerste versie. Beschikbaar voor review door community na 1e online sessie d.d. 23-4-2021.
0.2	10-6-2021	Ad Gerrits	Hoofdstukken 5, 6 en 7 toegevoegd. Beschikbaar voor review na 2e online sessie d.d. 10-6-2021.

Vragen, correcties en aanvullingen bij voorkeur in het document opnemen en mailen naar ad.gerrits@vng.nl .

Fundamentele vraagstukken kunnen (ook) in de Pleio community-omgeving worden geplaatst als vraag of discussie.

Inleiding en leeswijzer

Dit document beschrijft een aantal architectuuraspecten die van belang zijn bij het ontwerpen en uitvoeren van notificatieprocessen.

Dit document is een bijlage bij het hoofddocument dat binnen project Notificatieservices wordt opgeleverd. Het hoofddocument beschrijft een aantal onderwerpen die hier niet worden herhaald zoals aanleiding, projectdoel, projectscope, het waarom van gebeurtenisgedreven en notificeren en hoe de huidige situatie binnen de overheid er uit ziet.

Een aantal onderwerpen komen in beide documenten voor (bijv. Terminologie, Aspecten en een samenvattend hoofdstuk 'Architectuur'). In deze bijlage worden onderwerpen met meer diepgang beschreven. In het hoofddocument zijn ze kort en vereenvoudigd opgenomen. Tenslotte bevat dit document ook onderwerpen die niet in het hoofddocument staan (bijv. Protocollen).

Voor het lezen van de bijlage wordt een zekere mate van architectuurkennis verondersteld. Zo wordt in deze bijlage gebruik gemaakt van de Archimate architectuurbeschrijvingstaal voor onder andere begripsdefinities, architectuurbeschrijvingen en diagrammen (terwijl in het hoofddocument vereenvoudigde 'schetsafbeeldingen' worden gebruikt).

Het is het meest logisch om eerst het hoofddocument te lezen en dan de bijlage. Maar de bijlage kan ook als afzonderlijk document worden gelezen. Zowel het Hoofddocument als de Bijlage Architectuur zijn werkdocumenten. Tijdens het project zal worden besloten hoe informatie over diverse onderwerpen geordend en gepubliceerd zal worden.

U vindt in dit document de volgende hoofdstukken:

1	Begrippen	Een beschrijving van begrippen die worden gebruikt.
2	Architectuurstijlen	Verschillende architectuurstijlen die bruikbaar bij het ontwerp van notificatieprocessen.
3	Soorten gebeurtenissen en notificaties	Beschrijving van verschillende soorten 'gebeurtenissen' en hoe daarmee omgegaan moet worden.
4	Aspecten	Verschillende aspecten die van belang zijn om aandacht aan te schenken bij notificeren.
5	Publish-subscribe notificatiepatroon	PubSub patroon als notificatiepatroon en beschrijving er van aan de hand van Wikipedia definitie.
6	Standaarden voor PubSub patroon	Mogelijk bruikbare standaarden bij gebruik van het PubSub patroon voor notificeren
7	Implementatiepatronen	Bruikbare patronen om gebeurtenisgedreven te werken inclusief notificeren.
8	Protocollen	Beschrijving van een aantal toepasbare protocollen met voor- en nadelen.
9	Bijlages	Diverse bijlages.

1 Begrippen

1.1 Inleiding

Notificatieprocessen overschrijden organisaties, domeinen en disciplines en moeten voor alle betrokkenen helder een eenduidig zijn. Gebruik van een gezamenlijk begrippenkader is randvoorwaardelijk om overheidsbreed meer en beter event-driven te gaan (samen)werken.

Dit hoofdstuk beschrijft de belangrijkste begrippen die bij notificeren een rol spelen. Voor sommige van die begrippen geldt dat er in de praktijk verschillende termen worden gebruikt. Andersom geldt dat een bepaalde term soms in verschillende betekenissen wordt gebruikt. Dit hoofdstuk beschrijft voor welke termen is gekozen en welke betekenis ze hebben. Alle termen zijn opgenomen in de Begrippenlijst. De vastgestelde termen worden gebruikt om notificatieprocessen eenduidig te beschrijven. Gelet op het feit dat overheidsbreed notificeren betekent dat er vaak veel organisaties betrokken zullen zijn is het wenselijk om bij ontwerp en inrichting van notificatieprocessen van dit begrippenkader gebruik te maken.

Gebruik de vastgestelde terminologie bij het ontwerpen van notificatieprocessen

Voor het maken van architectuurbeschrijvingen en modellen maken we zoveel mogelijk gebruik van de architectuurbeschrijvingstaal Archimate. Archimate is een wereldwijde standaard en wordt al gebruikt in referentiearchitecturen binnen de Nederlandse overheid (bijv. NORA en GO). Bij ieder gebruikt begrip wordt aangegeven welke element uit de Archimate taal daarbij past en nemen de formele betekenis van dat element over. Om vergelijkbare redenen als bij terminologie geldt dat het wenselijk is om bij het maken van architectuurdiagrammen en beschrijvingen gebruik te maken van Archimate als beschrijvingstaal.

Gebruik Archimate bij het maken van architectuurdiagrammen en architectuurbeschrijvingen

Archimate onderscheidt (onder andere) een bedrijfs-, applicatie- en technologielaag. De bedrijfslaag beschrijft actoren, hun gedrag en de diensten die aan de buitenwereld worden geleverd. Bij het uitvoeren van bedrijfsprocessen wordt gebruik gemaakt van diensten uit de applicatielaag. De applicatielaag op haar beurt maakt gebruik van diensten uit de technologielaag. De bijlage ‘Archimate elementen’ beschrijft de belangrijkste elementen uit de Archimate-taall en hun betekenis in de context van notificatieprocessen.

1.2 Rollen bij notificatieprocessen

Bij notificatieprocessen zijn minimaal twee rollen van belang: een gegevensverstrekker en een gegevensafnemer. Afhankelijk van de context worden verschillende termen voor deze twee rollen gebruikt. Onderstaande tabel geeft een aantal in gebruik zijnde combinaties termen weer.

Rol 1	Rol 2	Context	Bedenking
(Dienst)aanbieder	(Dienst)afnemer	Dienstgerichte benadering.	Te algemeen
Verstrekker	Ontvanger	Algemeen	Te algemeen
Leverancier	Klant	Algemeen	Te algemeen
Provider	Consumer	Informatietechnologie	Te technisch
Producer	Consumer	Informatietechnologie	Te technisch

Publisher	Subscriber	Gebruik van 'Publish Subscribe' patroon.	Te specifiek
Houder Landelijke Voorziening	Afnemer	Digilevering	Te specifiek
Subject	Observer	Softwareontwikkeling	Te specifiek
Sender	Receiver	Informatietechnologie	Te specifiek

Kijkend naar de doelstellingen van het project Notificeren ('overheidsbrede afspraken over notificeren van gebeurtenissen') is gekozen voor gebruik van de termen: '**(Dienst)aanbieder**' en '**(Dienst)afnemer**'[1]. Redenen daarvoor zijn dat de termen:

- Passen bij de dienstgerichte benadering en terminologie die binnen de NORA en andere overheidsarchitecturen worden gehanteerd (o.a. Landelijke API-strategie en GEMMA Gegevenslandschap).
- (Ook) begrijpelijk zijn voor de bij notificatieprocessen belangrijke groep business-vertegenwoordigers.
- Algemeen genoeg zijn om verschillende type notificatieprocessen binnen verschillende contexten te beschrijven.

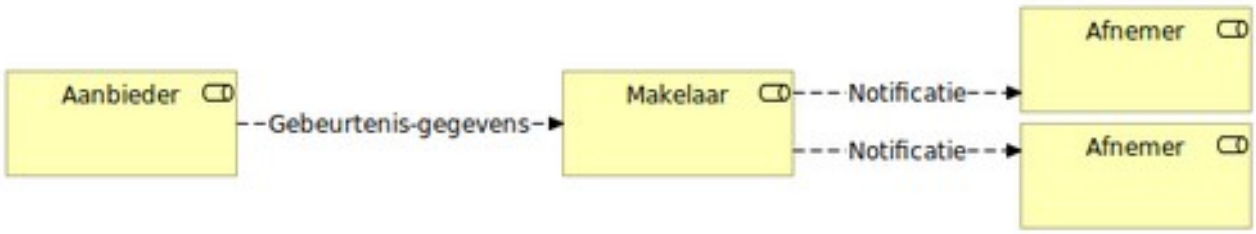
In zijn eenvoudigste vorm verstrekt een aanbieder notificaties aan afnemers:

De pijl geeft aan in welke richting gegevens stromen. Níet dat de aanbieder notificaties via een push-mechanisme aflevert bij afnemers want er kan ook gebruik worden gemaakt van pull-mechanismes. De formulering 'notificaties verstrekken aan afnemers' doet dus geen uitspraak over of daarbij gebruik wordt gemaakt van push- of pull-mechanismen.

Naast de rollen van 'aanbieder' en 'afnemer' kan ook sprake zijn van een derde rol die we aanduiden met de term '**(notificatie)makelaar**'[2]. De applicatie (of organisatie van wie de applicatie is) fungeert daarbij als tussenpersoon die notificatie-gerelateerde taken uitvoert voor zowel aanbieder als afnemer(s). Hij kan bijv. namens een aanbieder notificaties verstrekken aan afnemers of voor afnemers tijdelijk notificaties bewaren[3].

In die gevallen ontstaan onderstaande gegevensstromen:

[1]Voor de leesbaarheid wordt vaak volstaan met de rolomschrijving 'aanbieder' en 'afnemer'.
[2]Voor de leesbaarheid wordt vaak volstaan met de rolomschrijving 'makelaar'.
[3]Applicatiecomponenten die de rol van makelaar invullen worden vaak aangeduid met Engelstalige termen zoals 'messagebroker', 'eventbroker', '(message)queue'. Deze termen worden door IT'ers ook vaak gebruikt om deze rol aan te duiden.



Bij notificeren richten we ons vooral op gegevensuitwisseling tussen aanbieder en afnemers. Makelaars spelen daarbij vooral een technisch ondersteunende rol. Omwille van de eenvoud formuleren we zaken vaak in termen van ‘aanbieder en afnemer’. Alleen als de rol van makelaar van belang is wordt die expliciet benoemd. Een voorbeeld daarvan is als de rol van makelaar door een derde partij wordt ingevuld en het nodig is om daarmee afspraken over taken en verantwoordelijkheden te maken.

1.3 Notificeren en notificatie

Het werkwoord ‘notificeren’ is afgeleid van het Latijnse ‘notificare’ en betekent letterlijk: ter kennis brengen, bekendmaken, aankondigen (bron: Van Dale woordenboek). Binnen deze context gaat het om het bekendmaken van plaatsgevonden gebeurtenissen aan actoren of systemen die daarin geïnteresseerd zijn. Met de zin ‘Een aanbieder of makelaar notificeert een afnemer’ bedoelen we dus: ‘een aanbieder of makelaar maakt een plaatsgevonden gebeurtenis bekend aan een afnemer die hierin is geïnteresseerd’.

‘Notificatie’ is het zelfstandig naamwoord dat is afgeleid van ‘notificeren’ en heeft als betekenis: bekendmaking. Met de zin ‘Een aanbieder of makelaar levert een notificatie aan een afnemer’ bedoelen we: ‘een aanbieder of makelaar levert een bekendmaking van een plaatsgevonden gebeurtenis aan een afnemer die hierin is geïnteresseerd’.

In termen van de drie rollen ‘aanbieder’, ‘makelaar’ en ‘afnemer’ geldt dat de aanbieder, onder andere, informatie levert aan de makelaar over een plaatsgevonden gebeurtenis. Om die reden wordt dit soms ook aangeduid met de termen ‘notificeren’ en ‘notificatie’. Om misverstanden te voorkomen met notificaties die aan afnemers worden verstrekt doen we dit niet. De gegevens die de aanbieder aan de makelaar zijn een combinatie van stuurgegevens (‘header-informatie’) voor de makelaar om afnemers te kunnen notificeren en inhoudelijke gegevens (‘payload-informatie’) die als notificatie richting afnemers worden verstrekt. We duiden de informatieverstrekking van aanbieder naar makelaar daarom niet aan als ‘notificeren’ maar als (het verstrekken van) ‘gebeurtenisgegevens’.

Gebruik de term ‘notificatie’ alleen voor gegevens die naar aanleiding van plaatsgevonden gebeurtenissen aan afnemers worden verstrekt.

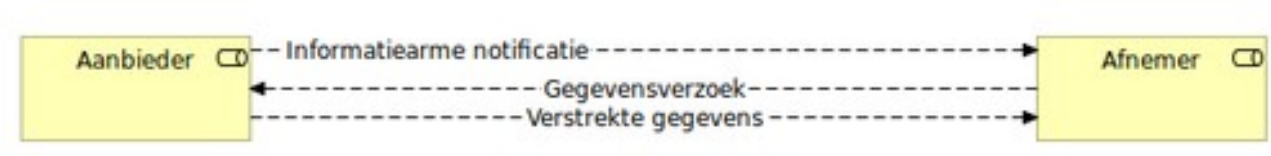
We maken onderscheid in twee fundamenteel verschillende patronen van notificeren: informatierijk en informatiearm. Bij informatierijk notificeren bevatten notificaties uitgebreide informatie over een plaatsgevonden gebeurtenis en de gevolgen daarvan. Afnemers hebben daaraan voldoende om eventuele acties op basis van de ontvangen informatie te gaan ondernemen. Bij informatiearm notificeren wordt alleen informatie verstrekt aan afnemers over het feit dát zich een bepaalde gebeurtenis heeft voorgedaan. Voordat een eventuele vervolgactie kan worden gestart zal de afnemer eerst aanvullende informatie moeten gaan opvragen.

In enge zin kan een notificatieproces als afgesloten worden beschouwd op het moment dat een notificatie aan een afnemer is verstrekt. In het geval van informatiearm notificeren is er vanuit het perspectief van

afnemers echter nog geen sprake van een afgerond proces. Wij zien de extra stap om aanvullende gegevens op te vragen daarom als onderdeel van notificatieprocessen in ruime zin.

Voor het notificatieproces betekent dit dat er na ontvangst van een notificatie van een extra processtap sprake kan zijn waarbij:

- De afnemer op basis van de ontvangen notificatie een verzoek doet bij de aanbieder (aanvullende) gegevens (‘Gegevensverzoek’) en
- De aanbieder de gevraagde gegevens verstrekt aan de afnemer (‘Verstrekte gegevens’).



1.4 Gebeurtenisgedreven (event-driven)

Met ‘gebeurtenisgedreven’ bedoelen we dat plaatsgevonden gebeurtenissen aanleiding zijn voor aanbieders om in de vorm van notificaties snel informatie daarover te verstrekken aan geïnteresseerde afnemers. Gebeurtenissen fungeren daarbij dus als trigger voor het initiëren van gegevensuitwisseling. Dit in tegenstelling tot situaties waarin afnemers op eigen initiatief informatie gaan opvragen bij een aanbieder.

Internationaal is de term ‘event-driven’ gebruikelijk. Binnen deze Bijlage Architectuur gebruiken we naast de term ‘gebeurtenisgedreven’ ook de term ‘event-driven’. De reden daarvoor is dat daarmee wordt aangesloten bij wat internationaal gebruikelijk is qua terminologie binnen architectuur en informatietechnologie.

De term ‘gebeurtenisgestuurd’ wordt binnen Nederland soms als synoniem gebruikt. We kiezen voor de term ‘gebeurtenisgedreven’ omdat dit beter aansluit bij de Engelse term ‘event-driven’.

1.5 Diverse aspecten

Bij het ontwerp en de uitvoering van notificatieprocessen kunnen op een aantal aspecten keuzes worden gemaakt. Hieronder staat welke term we voor een aantal aspecten gebruiken en wat er mee bedoeld wordt. In een apart hoofdstuk wordt toegelicht welke keuzes per aspect mogelijk zijn en welke voorkeuren daarbij gelden.

Term	**Betekenis**
Push	Bij gebruik van push-mechanismen worden notificaties bezorgd bij afnemers (die dus in staat moeten zijn om ze te ontvangen)
Pull	Bij gebruik van pull-mechanismen stelt een aanbieder notificaties beschikbaar en kunnen afnemers die op een zelf te bepalen moment verwerken.
Synchroon	Bij synchrone communicatie zijn aanbieder en afnemer gelijktijdig actief tijdens informatie-uitwisseling en wacht de aanbieder op (inhoudelijk) antwoord van de afnemer.
Asynchroon	Bij asynchrone communicatie zijn aanbieder en afnemer niet gelijktijdig actief maar stellen ‘om beurten’ informatie beschikbaar. De aanbieder wacht niet op

	(inhoudelijk) antwoord maar hooguit op een (technische) ontvangstbevestiging.
(Ont)koppeling	De mate van ‘koppeling’ tussen applicaties geeft aan in welke mate ze in hun functioneren afhankelijk zijn van elkaar. Het wenselijk dat applicaties voldoende ontkoppeld zijn.
Betrouwbaarheid (van notificeren)	‘Betrouwbaarheid’ gaat binnen deze context over de mate waarin notificatieprocessen in staat zijn om notificatie van afnemers betrouwbaar te laten verlopen.
Veiligheid (van notificeren)	‘Veiligheid’ gaat binnen deze context over de mate waarin notificatieprocessen in staat zijn om notificatie van afnemers veilig te laten verlopen.

2 ARCHITECTUURSTIJLEN

2.1 INLEIDING

Voor het ontwerpen en uitvoeren van notificatieprocessen kan gebruik worden gemaakt van verschillende architectuurstijlen. Iedere stijl kent een aantal fundamentele uitgangspunten met betrekking tot hoe processen en systemen moeten worden ontworpen. Dit hoofdstuk beschrijft een viertal bewezen architectuurstijlen en de manier ze bij het ontwerpen van notificatieprocessen zijn te gebruiken. Doel hiervan is om in de praktijk verschillende stijlen te kunnen toepassen en combineren om binnen de eigen context optimale notificatieprocessen te kunnen ontwerpen.

2.2 SERVICE-ORIËNTATIE

Service-oriëntatie is een binnen de overheid bekende architectuurstijl waarbij sprake is van aanbieders van diensten en afnemers die diensten conform afspraken afnemen. Kenmerkend voor de diensten is:

- *Een service is virtueel: de afnemer heeft geen weet van de implementatie van de dienst. De dienst is onafhankelijk van de afnemer. Scheiding van verantwoordelijkheid is expliciet vastgesteld. Voordeel hiervan is onafhankelijkheid van veranderingen van afnemer en leverancier. Voldoen aan het contract staat immers centraal;*
- *Een service is gestandaardiseerd: er is slechts één implementatie aanwezig van een verantwoordelijkheid. Voordeel is het rationaliseren van de standaard. Standaardiseren=massa, massa=kassa. De dienst wordt 'commodity' en eenvoudig te vervangen door een andere leverancier of goedkoper door een efficiëntieslag;*
- *Een service is modulair (vervangbaar) en compositioneerbaar. Standaard is niet flexibel. Flexibiliteit wordt bereikt door combineren (componeren) van standaards tot een nieuwe standaard;*
- *Een service is abstract: generiek, niet afgestemd voor 1 specifieke afnemer, maar op een doelgroep van afnemers;*
- *Losgekoppeld: afnemer en leverancier zijn maximaal onafhankelijk van implementatie van beide. Elke service is daarom autonoom. Er bestaat geen directe link of relatie tussen verschillende services. Services zijn zich ook niet van elkaar bewust. – Bron: Wikipedia*

De Nederlandse Overheid Referentie Architectuur (NORA) benoemt service-oriëntatie als de leidende architectuurstijl voor de Nederlandse overheid. Naast de brede toepasbaarheid ervan is een belangrijke reden dat deze stijl zeer geschikt is om binnen de overheid ‘interoperabiliteit’ te realiseren:

> *Interoperabiliteit is het vermogen van organisaties (en hun processen en systemen) om effectief en efficiënt informatie te delen met hun omgeving – bron: NORA*

Notificatieprocessen spelen zich af binnen een gedistribueerd en heterogeen landschap van organisaties en applicaties. Interoperabiliteit is nodig op bedrijfsmatig, gegevens- en applicatie- en technologisch niveau. Applicaties in de rol van aanbieder en afnemer van notificaties kennen verschillende verantwoordelijkheden (‘separation of concerns’) en worden zoveel mogelijk losgekoppeld (‘high cohesion, low coupling’). Het aspect ‘losgekoppeld’ wordt meer in detail besproken in het hoofdstuk ‘Niet functionele aspecten’.

Services stellen hun diensten beschikbaar voor gebruik door middel van Application Programming Interfaces (API’s). Er zijn verschillende stijlen om API’s te ontwikkelen, met eigen voor- en nadelen. Binnen

de overheid wordt de REST-stijl steeds vaker toegepast. Reden om die later in dit hoofdstuk apart te beschrijven.

Ook bij notificatieprocessen is sprake van aanbieders en afnemers van diensten. Service-oriëntatie is daarom ook voor het ontwerp van notificatieprocessen een bruikbare architectuurstijl.

2.3 Message-driven

Applicaties moeten tegenwoordig in staat zijn om via het uitwisselen van gegevens samen te werken met andere applicaties. Daarbij zijn verschillende integratiestijlen toe te passen. Bekende integratiestijlen zijn:

- Gebruik van bestanden of databases: De aanbieder biedt hierbij een bestand of database aan dat beschikbaar wordt gesteld voor gebruik door afnemers. Op basis van afspraken over inhoud en formaat kunnen afnemers gegevens lezen en gebruiken. Beschikbaarstelling kan via delen of (deel) kopieën. Bij uitwisseling van kopieën wordt gebruik gemaakt van bestandsformaten zoals CSV, JSON en XML, push- of pull-mechanismes, protocollen zoals FTP, HTTP en RPC.
- Remote Procedure Call (externe applicatiefunctionaliteit aanroep): Afnemers roepen hierbij een door de aanbieder beschikbaar gestelde service aan die bepaalde functionaliteit biedt en bijvoorbeeld bepaalde gegevens naar de afnemer retourneert. Het meest gebruikte transportprotocol is HTTP, het meest gebruikte berichtprotocol SOAP en de gangbare berichtformaten zijn XML en JSON. gRPC is een framework voor communicatie via RPC met hoge performance.

Beide stijlen zijn binnen bepaalde contexten bruikbaar maar kennen ook grote nadelen bij meer complexe integratievraagstukken waarbij bijv..

- processen moeten worden getriggerd
- gegevens realtime beschikbaar moeten komen
- op grote schaal, met veel organisaties en applicaties, moeten worden gewerkt
- het cruciaal is om applicaties ontkoppeld te houden[4].

De integratiestijl die dan het meest geschikt is, en in de praktijk het vaakst wordt toegepast, is ‘messaging’. Daarbij verzenden aanbieders informatie naar afnemers in de vorm van ‘berichten’:

> *Een bericht is een gegevensrecord dat een berichtensysteem kan verzenden via een berichtkanaal. – bron: Enterprise Integration Patterns, G. Hohpe and B. Woolf*

Berichten zijn geschikt om als zelfstandig object te vervoeren en tijdelijk of permanent op te slaan en kunnen van metagegevens voor verwerking worden voorzien. Een bericht kan bijv. van een bestemming worden voorzien om door een in berichtbezorging gespecialiseerde partij bij de juiste afnemer te laten bezorgen.

Gebruik bij het ontwerp van notificatieprocessen de integratiestijl messaging

Communicatie vindt bij messaging in de regel asynchroon plaats om maximale ontkoppeling van applicaties te kunnen realiseren waardoor ze onafhankelijk van elkaar en met hoge betrouwbaarheid kunnen functioneren. Bij notificatieprocessen is dit een zeer belangrijk aandachtspunt. Reden om berichtuitwisseling bij voorkeur asynchroon te laten verlopen.

[4] ‘Ontkoppeling’ kent een aantal aspecten en is in zijn algemeenheid, maar zeker binnen notificatieprocessen, van belang. In het hoofdstuk ‘aspecten' wordt het apart besproken.

Gebruik bij notificatieprocessen bij voorkeur asynchrone berichtuitwisseling

Vaak worden daarbij gespecialiseerde componenten (‘enterprise service bus middleware’) gebruikt die in de rol van ‘makelaar’ berichtuitwisseling ondersteunen. Middleware voorzieningen kunnen bijv. zorgen voor het tussentijds opslaan van berichten om te voorkomen dat ze kwijtraken, berichten bufferen en als ‘schokdemper' optreden als het tempo waarin berichten beschikbaar komen voor een afnemer te hoog is of garanties bieden dat berichten daadwerkelijk zijn afgeleverd. Het leveren van dit type functionaliteit is, zeker binnen meer complexe omgevingen met veel partijen, niet eenvoudig te realiseren. Door dit type functionaliteit te beleggen bij gespecialiseerde voorzieningen is de kwaliteit van berichtuitwisseling te verhogen en kunnen bedrijfsapplicaties zich beperken tot hun kernfuncties (‘separation of concerns’).

Gebruik gespecialiseerde voorzieningen als robuust en betrouwbaar berichtenverkeer nodig is

Voor transport van berichten kan zowel gebruik worden gemaakt van algemene protocollen zoals HTTP of van speciale, voor messaging ontwikkelde, protocollen zoals AMQP, STOMP. XMPP en MQTT. Ieder protocol heeft zijn eigen karakteristieken en toepassingsgebied. Het ene protocol is bijv. heel geschikt om zeer betrouwbare uitwisseling te ondersteunen maar relatief langzaam in gebruik terwijl een ander protocol minder betrouwbaarheid garandeert maar wel zeer hoge snelheid van berichtuitwisseling mogelijk maakt. In een apart hoofdstuk wordt enkele veelgebruikte protocollen voor notificeren meer in detail beschreven.

2.4 EVENT-DRIVEN

De steeds dynamischer en digitalere wereld waarin we leven, waarin alles met elkaar is verbonden, dwingt organisaties en applicaties om informatie snel, bij voorkeur realtime, te verwerken. Na het plaatsvinden van gebeurtenissen moeten er snel en betrouwbaar diensten worden geleverd.

Binnen de event-driven (‘gebeurtenisgedreven’) architectuurstijl staat het werken met gebeurtenissen (‘events’) centraal[5]:

> *Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events. – bron: Wikipedia*

Gebeurtenissen functioneren daarbij als trigger om losgekoppelde actoren, organisaties en applicaties, in actie te laten komen op de momenten dat het nodig is. De event-driven architectuurstijl is te beschouwen als een noodzakelijke aanvulling op de servicegerichte en messaging stijl. Waar klassieke service-oriëntatie zich veelal richtte op meer statische vraag-gedreven bedrijfsprocessen ontstaat bij een event-driven architectuur een netwerk van diensten die elkaar informatie verstrekken op het moment dat zich bepaalde gebeurtenissen hebben voorgedaan.

Bij gebruik van de messaging-stijl is er vaak sprake van communicatie tussen twee actoren (‘point-to-point'). Ieder actor heeft een uniek adres dat gebruikt wordt om hem iets te vragen. Bij gebruik van de event-driven stijl stellen aanbieders op eigen initiatief nieuwe informatie beschikbaar. Organisaties en applicaties zijn daarbij maximaal ontkoppeld. Afnemers zijn in veel gevallen niet eens bekend bij de

[5]We hebben gekozen voor gebruik van de Engelstalige term ‘event-driven’ omdat het daarbij gaat om een wereldwijd gebruikte term die voornamelijk door meer technisch-georienteerde mensen wordt gebruikt. De keuze voor gebruik van de term ‘gebeurtenis’ i.p.v. ‘event’ is omdat die term ook vaak wordt gebruikt in communicatie met niet-technici.

aanbieder. Event-driven werken maakt het snel (‘near realtime’) informatie uit te wisselen. Iets dat steeds vaker gewenst of zelfs noodzakelijk is.

Gebruik de event-driven architectuurstijl als actuele informatie snel beschikbaar moet komen

Bij gebruik van de event-driven stijl kennen we ‘producenten’ (‘aanbieders’) die via ‘notificaties’ ‘consumenten’ (‘afnemers’) informeren. Daarbij wordt veelal gebruik gemaakt van de messaging integratiestijl met asynchrone berichtuitwisseling. Vaak worden gespecialiseerde voorzieningen (‘event brokers') ingezet die in de rol van ‘makelaar' asynchrone berichtuitwisseling tussen aanbieders en afnemers ondersteunen. Asynchrone berichtuitwisseling en gebruik van autonome event brokers worden zo dikwijls toegepast dat ze vaak worden beschouwd als voorwaarden om processen ‘event-driven’ te noemen.

Omdat er talloze soorten gebeurtenissen zijn is de gebeurtenisgestuurde architectuurstijl binnen allerlei contexten bruikbaar. Bijv. wanneer interactieve software moet reageren op uitgevoerde handelingen door een gebruiker (bijv. met behulp van toetsenbord of muis). Binnen het Internet of Things waar allerlei componenten zijn gekoppeld en elkaar van informatie voorzien als zich bepaalde gebeurtenissen hebben voorgedaan (bijv. een sensor die overschrijding van een signaalwaarde constateert). Bij gebruik van microservices, extreem losgekoppelde services, die elkaar notificeren als er iets notificatiewaardigs gebeurt (bijv. een bestelcomponent die bekend maakt dat er een nieuwe bestelling is geplaatst).

De event-driven stijl kan op verschillende manieren worden toegepast. Er wordt meestal gebruik gemaakt van de messaging-stijl waarbij berichten (meta- en inhoudelijke) gegevens over een gebeurtenis bevat. De makelaarrol wordt meestal ingevuld door gespecialiseerde (middleware)applicaties die met behulp van mechanismen als ‘topics’ en ‘queues‘ flexibele en robuuste oplossingen mogelijk maakt om gebeurtenisinformatie in de vorm van berichten bij afnemers te brengen.

Maar er kan ook meer worden gedaan dan alleen berichten afleveren. Beschikbare informatie over gebeurtenissen kan ook op andere manieren worden verwerkt waardoor hele nieuwe mogelijkheden beschikbaar komen, zoals:

- Het in volgorde van ontvangst (‘log based’) tijdelijk of permanent bewaren van berichten zodat ze beschikbaar blijven om bijv. analyses of op latere momenten opvragen mogelijk te maken. Bij permanente bewaring kunnen ze zelfs gaan fungeren als primaire gegevensbron (bijv. bij het event sourcing patroon).
- Het aanbieden van verschillende views die op basis van bewaarde gebeurtenisberichten zijn gemaakt (‘Statefull stream processing’). Bijv. een view die de actuele waarde van objectattributen toont of een view die geschikt is om bepaalde analyses mee te doen.
- Het ‘assembleren van verschillende componenten’ om oplossingen te creëren in plaats van gebruik van 1 gespecialiseerde applicatie. Moderne cloudomgevingen, waarin componenten gebeurtenisgedreven samenwerken, kennen bijv. serverless functies die gebeurtenisinformatie kunnen verwerken en zelf nieuwe gebeurtenissen kunnen genereren. In plaats van centrale verwerking door 1 makelaarapplicatie is daarbij eerder sprake van een keten of netwerk van verwerkende componenten.

Het hoofdstuk ‘Implementatiepatronen’ beschrijft een aantal ‘patronen’ om verschillende vormen van verwerking in te richten.

Project Notificatieservices richt zich primair op processen waarbij het nodig is dat (bedrijfs-)applicaties elkaar notificeren als zich bepaalde gebeurtenissen hebben voorgedaan. Met name als daarbij brongegevens zijn gewijzigd waar afnemers in geïnteresseerd zijn. Gebruik van de event-driven stijl stelt overheidsorganisaties daarmee in staat om elkaar snel van actuele informatie te voorzien om daarmee haar processen rond wettelijke taken beter te kunnen uitvoeren.

Binnen de overheid wordt vooral data- en procesgedreven en minder gebeurtenisgedreven gewerkt. De event-driven architectuurstijl wordt dus nog weinig toegepast binnen de overheid. Om voldoende flexibel, snel en betrouwbaar te kunnen opereren en diensten te kunnen leveren zoals burgers en bedrijven tegenwoordig die tegenwoordig verwachten is meer gebruik daarvan nodig. Effectief gebruik van de stijl vereist echter een fundamenteel andere manier van denken en ontwerpen dan op dit moment gebruikelijk is (‘paradigmashift’). Daarvoor zijn zowel aanpassingen in cultuur als technologie nodig. Iets dat meerdere jaren zal vergen. Een eerste stap die meteen gezet kan worden is om bij (her)ontwerp van processen een (meer) event-driven aanpak te overwegen. Met name waar het gaat om processen waarbij snel over nieuwe informatie beschikken kan leiden tot effectievere oplossingen.

Gebruik bij (her)ontwerp van processen waarbij informatie wordt uitgewisseld vaker de event-driven stijl

2.5 REST

De Representational State Transfer (REST) architectuurstijl is bedoeld voor het ontwerp van software binnen gedistribueerde hypermedia (‘gelinkte’) systemen. Voor de communicatie tussen applicaties gelden zes eisen. In de praktijk wordt daar vaak in verschillende mate aan voldaan. Reden om diensten en API’s vaak als ‘RESTful’ aan te duiden[6]. Het fundamentele concept van REST is de 'resource'. Alle informatie die benoemd kan worden is te beschouwen als een resource. Er is sprake van cliënt-server relaties, waarbij een server informatie over resources via een Application Programming Interface (API) aan cliënts beschikbaar stelt, vaak in gegevensformaten zoals JSON of XML. Resources kunnen eventueel worden gewijzigd, waarbij acties en relaties vindbaar zijn via ‘hypermedia’ (links naar andere bronnen). Acties worden aangeduid met HTTP-termen zoals GET voor opvragingen en POST voor creatie van resourceinstanties.

De REST architectuurstijl is relatief eenvoudig toe te passen en geschikt voor situaties waarin domeingegevens moeten worden opgevraagd of gewijzigd. De stijl wordt (ook) binnen de overheid steeds vaker toegepast ten koste van bijv. het SOAP-protocol waarmee ook HTTP webservicess zijn te ontwikkelen (bijv. om gegevens uit bronregistraties op te vragen). Om uniform en eenduidig gebruik van de REST-stijl binnen de overheid te bevorderen is de REST-API Design Rules standaard ontwikkeld.

Bij gebruik van de REST-stijl is echter slechts beperkt ontkoppeling tussen applicaties mogelijk. REST is ontworpen voor synchrone communicatie waarbij cliënts na het indienen van een verzoek (‘request’) wachten op antwoord (‘reply’) van een server. Deze vorm van synchrone communicatie leidt tot ongewenste afhankelijkheid die bij notificatieprocessen onwenselijk of zelfs niet acceptabel kan zijn. In gedistribueerde omgevingen met veel verschillende applicaties is bijv. niet te garanderen dat alle applicaties altijd gelijktijdig actief zijn en met elkaar kunnen communiceren. Ook op andere relevante aspecten, zoals het snel en betrouwbaar afleveren van berichten, biedt REST slechts zeer beperkt functionaliteit waardoor zowel bij aanbiedende als afnemende applicaties aanvullende functionaliteit nodig is.

REST is met name ontwikkeld om op een gestandaardiseerde, op HTTP voortbouwende, manier interactief met domein-entiteiten te werken. REST-API's ondersteunen daarvoor bijv. ‘CRUD-acties’ (Create, Read,

[6] Via een volwassenheidsmodel is te bepalen in welke mate een service voldoet aan de zes REST-eisen.

Update, Delete). Voor acties die geen directe relatie met resources hebben is het weinig zinvol, of zelfs onwenselijk, om aan alle REST eisen te moeten voldoen (maar is bijv. gebruik van remote procedure calls of messaging geschikter).

In tegenstelling tot wat binnen de overheid momenteel voor opvragingen het geval lijkt te zijn, is gebruik van de REST-stijl voor notificeren dus geen vanzelfsprekendheid. Daarvoor is bijv. asynchrone berichtuitwisseling geschikter. Gelet op het brede gebruik, het relatief eenvoudige gebruik en meerwaarde als notificaties sterk 'resource-georiënteerd' zijn kan gebruik van de REST-stijl desondanks geschikt zijn.

Gebruik de REST-stijl als synchrone communicatie voldoet en/of resource-oriëntatie aanwezig is

2.6 Domain Driven Design

Bij notificatieprocessen wordt informatie uitgewisseld tussen applicaties die gebruikt worden binnen verschillende 'taakgebieden' of 'domeinen', Ieder domein kent zijn eigen taal, processen, gegevens en applicaties. Om de uitwisseling van gegevens effectief te laten verlopen moeten op verschillende gebieden afspraken en standaarden worden afgesproken. Bij notificeren moet bijvoorbeeld goed worden gedefinieerd wat de betekenis is van gebeurtenissen binnen het domein van een aanbieder en hoe informatie daarover die in de vorm van notificaties aan afnemers wordt verstrekt. Afnemers moeten die informatie interpreteren en in veel gevallen gaan 'vertalen' naar begrippen die binnen hun domeincontext relevant zijn.

Domain driven design (DDD) is een stijl die de nadruk legt op het eenduidig, in termen van de business, modelleren van een bepaald domein. Business-vertegenwoordigers en software-ontwikkelaars spreken dezelfde eenduidige taal ('ubiquitous language') die de kernbegrippen binnen het domein bevat. DDD onderkent het feit dat standaardisatie van modellen en taal slechts tot op zeker hoogte mogelijk én wenselijk is binnen een divers systeem, zeker wanneer daarbij meerdere organisaties betrokken zijn.

> *"In an ideal world, we would have a single model spanning the whole domain of the enterprise. … Total unification of the domain model for a large system will not be feasible or cost-effective"*
> *– bron: Domain Driven Design, Eric Evans*

Domeinen gebruiken, vanwege het taakspecifieke karakter vaak om goede redenen, verschillende begrippen in een andere betekenis of dezelfde begrippen met een verschillende betekenis. Zo kan een 'klant' in het ene domein vergelijkbaar zijn met wat in een ander domein 'cliënt' wordt genoemd. Andersom kan het begrip 'klant' in meerdere domeinen worden gebruikt maar een, ietwat of sterk, verschillend betekenis hebben. DDD hecht daarom veel belang aan het zorgvuldig afbakenen van wat wel en niet tot een domein behoort ('bounded context') en benadrukt dat begrippen, inclusief gebeurtenissen, een domeingebonden betekenis hebben. Iets waar bij gegevensuitwisseling, bijvoorbeeld door middel van notificaties, expliciet rekening mee moet worden gehouden.

Afhankelijk van het soort relatie tussen domeinen zijn verschillende uitwisselpatronen toepasbaar. Bij domeinen die een gelijkwaardige relatie hebben kunnen bepaalde gegevens bijv. gezamenlijk worden gedefinieerd en gebruikt ('Shared Kernel'). Is de ene partij van de ander afhankelijk en hebben beide belang bij een goede uitwisseling dan spelen overleg, afspraken en standaarden een belangrijke rol ('Customer-Supplier'). Is de aanbieder volledig bepalend dan moeten afnemers zich daarnaar schikken ('Conformist') en binnen hun eigen omgeving maatregelen treffen (bijv. via een 'Anticorruption Layer').

DDD benadrukt dat naast gegevens ook gebeurtenissen binnen een domein goed in kaart gebracht en beschreven moeten worden:

> *Model information about activity in the domain as a series of discrete events. Represent each event as a domain object. These are distinct from system events that reflect activity within the software itself, although often a system event is associated with a domain event, either as part of a response to the domain event or as a way of carrying information about the domain event into the system. A domain event is a full-fledged part of the domain model, a representation of something that happened in the domain. Ignore irrelevant domain activity while making explicit the events that the domain experts want to track or be notified of, or which are associated with state change in the other model objects – bron: Domain-Driven Design Reference – Definitions and Pattern Summaries – Eric Evans*

Domeingebeurtenissen spelen een cruciale rol bij event-driven werken en notificeren. Inzichten uit DDD zijn behulpzaam zijn om te zorgen dat er voldoende aandacht is voor domeingebeurtenissen en dat informatie van aanbieders goed wordt geïnterpreteerd en gebruikt. DDD biedt zowel conceptueel als in de vorm van uitwisselpatronen bruikbare handvatten om notificatieprocessen goed in te richten.

Gebruik inzichten en patronen uit Domain Driven Design bij notificatieprocessen

2.7 Conclusies

Voor het meer event-driven gaan werken en gebruiken van notificaties als communicatiemiddel tussen applicaties kan gebruik worden gemaakt van een aantal bewezen architectuurstijlen. De servicegeoriënteerde architectuurstijl, die al vaak wordt toegepast binnen de overheid, is er daar een van. Als integratiestijl verdient messaging de voorkeur omdat daarmee de meest flexibele en betrouwbare oplossingen zijn te realiseren. De event-driven stijl is bij uitstek geschikt om event-driven te werken en daarmee snelle en responsieve dienstverlening te kunnen leveren. De REST architectuurstijl kan ook bij notificatieprocessen haar meerwaarde hebben maar is niet in alle situaties het meest geschikt (bijv. omdat berichtuitwisseling synchroon plaatsvindt). Ten slotte geldt voor Domain Driven Design dat daar aanwezige inzichten en patronen zeer bruikbaar zijn om goed om te gaan met verschillen binnen betrokken domeinen.

Gebruik bewezen architectuurstijlen bij ontwerp van notificatieprocessen

Bij de keuze voor gebruik van (elementen van) een stijl speelt context altijd een belangrijke rol. Er is dus geen spraken van ‘de beste stijl’ maar het gaat er om om binnen een specifieke context te bepalen welke stijl het meest geschikt is. Daarbij kan bijvoorbeeld de kennis, ervaring en mogelijkheden van betrokken organisaties een belangrijke rol spelen.

Gebruik stijlen die passen bij de context

Het is uiteraard ook mogelijk om gebruik te maken van een combinatie aan stijlen. Als een makelaarrol wordt ingevuld door een aparte applicatie kan voor de communicatie tussen aanbieder en makelaar bijv. gebruik worden gemaakt van een andere stijl dan bij de communicatie tussen makelaar en afnemers. Voor afnemers geldt dat er, zeker bij grote aantallen, vaak verschillen zullen zijn in aanwezige voorkeuren en mogelijkheden. In die gevallen kan gebruik van verschillende stijlen wenselijk of zelfs noodzakelijk zijn.

Gebruik meerdere stijlen bij diversiteit in mogelijkheden bij afnemers

3 Soorten gebeurtenissen en notificaties

3.1 Inleiding

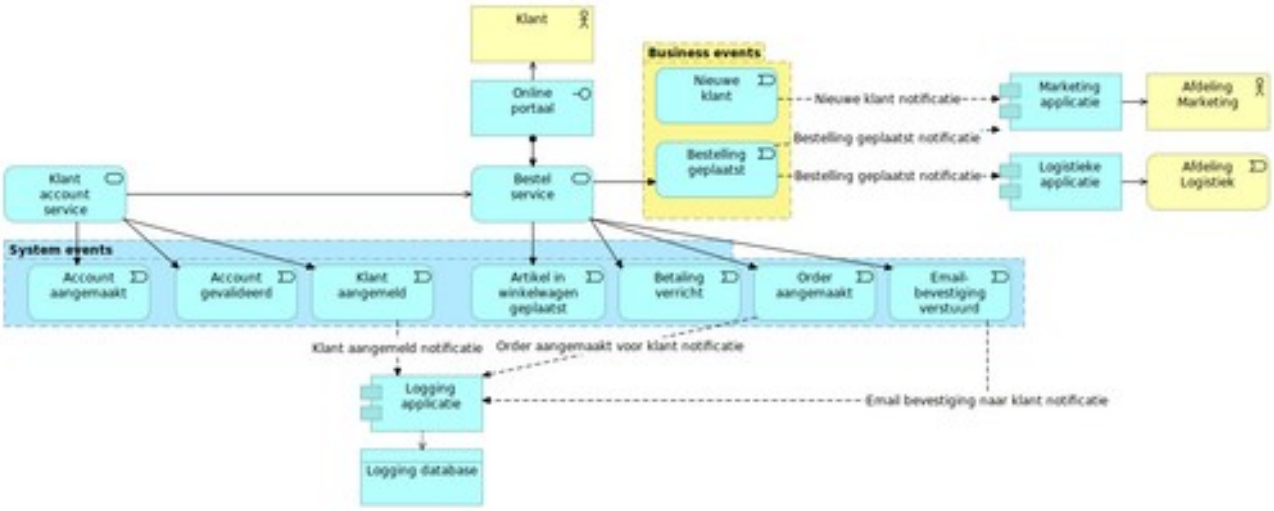
Er zijn verschillende manieren om gebeurtenissen en notificaties te typeren en onderscheiden. Het is belangrijk dat voor alle betrokkenen duidelijk is wat voor soort gebeurtenissen we onderscheiden, hoe we die noemen en wat er binnen de context van notificeren mee bedoeld wordt.

3.2 Business- en systeemgebeurtenissen

Een businessgebeurtenis is een gebeurtenis die plaatsvindt binnen een bepaald businessdomein. Zo'n gebeurtenis heeft een duidelijke betekenis voor de bedrijfsvoering binnen het domein en wordt uitgedrukt in de taal van businessvertegenwoordigers die binnen het domein werken (bijv. "order geplaatst"). Dergelijke gebeurtenissen vinden plaats ongeacht of en hoe ze met behulp van software en technologie worden vastgelegd.

Een 'systeem-gebeurtenis' is een systeem-technische term waarmee wordt aangegeven dat er iets gebeurt binnen een (software- of registratie)systeem. Vaak door te beschrijven welke gegevens naar aanleiding van een gebeurtenis zijn gewijzigd. Waar business-vertegenwoordigers het hebben over een businessgebeurtenis als 'order geplaatst' kunnen IT'ers dit 'vertalen' in systeemgebeurtenissen met namen als "order created" en "order items created". Notificaties naar aanleiding daarvan bevatten vaak gegevens die een afspiegeling zijn van de betrokken objecten en attributen zoals die in een register zijn opgeslagen. Een business-gebeurtenis leidt meestal tot meerdere systeemgebeurtenissen. Eén transactie kan bijv. leiden tot het maken of wijzigen van meerdere gerelateerde gegevensobjecten ('entiteiten').

Onderstaand diagram illustreert hoe bij een online geplaatste bestelling door een nieuwe klant sprake is van een tweetal businessgebeurtenissen ("Nieuwe klant" en "Bestelling geplaatst") en zeven systeemgebeurtenissen.



In bovenstaand diagram is zichtbaar dat afnemers, in dit geval de afdelingen Marketing en Logistiek, vooral geïnteresseerd zijn in businessgebeurtenissen en niet in de onderliggende systeemgebeurtenissen. Een uitzondering daarop vormen 'technische notificaties' voor IT-behoeften zoals het binnen de eigen omgeving repliceren van externe bronregistergegevens. Iets dat steeds minder nodig en wenselijk is naarmate het

opvragen van gegevens bij de bron en event-driven werken met notificaties toeneemt. In de regel moeten notificaties informatie bevatten over plaatsgevonden businessgebeurtenissen.

Notificeer over relevante businessgebeurtenissen

Zoals eerder aangegeven is het raadzaam om bij notificeren gebruik te maken van de taal die binnen het businessdomein wordt gesproken. Dit zorgt er voor dat ze eenduidig en herkenbaar zijn voor zowel business-vertegenwoordigers, domeinexperts en IT'ers wat communicatie vergemakkelijkt. De aanbeveling is om hier zo vroegtijdig mogelijk mee te beginnen bij nieuwe ontwikkelingen.

Gebruik de taal van het domein voor het beschrijven van gebeurtenissen en ontwerpen van notificatieprocessen

Op dit moment wordt binnen de overheid nog vaak datagedreven ontwikkeld. Bijvoorbeeld door verschillende soorten informatiemodellen te maken die gegevens en hun onderlinge relaties beschrijven. Op basis daarvan kunnen dan bijv. elementaire operaties (bijv. 'CRUD') worden gedefinieerd waarmee gegevens actueel zijn te houden. Om meer event-driven te kunnen werken moet er meer aandacht komen voor alles wat er binnen een domein gebeurt: gebeurtenissen moeten als 'first-class citizens' worden beschouwd. Daarvoor zal binnen projecten vroegtijdig met domeinexperts informatie over businessgebeurtenissen in kaart moeten worden gebracht (bijv. via 'event storming'). Daarmee is in beeld te brengen welke businessgebeurtenissen er naar aanleiding van welke triggers plaatsvinden, wie daarvoor verantwoordelijk is, etc.. Zoals eerder aangegeven betekent het een grote verandering en zijn een andere denkwijze, ontwikkeling, methoden en technieken nodig (reden waarom deze verandering eerder is getypeerd als een 'paradigmashift').

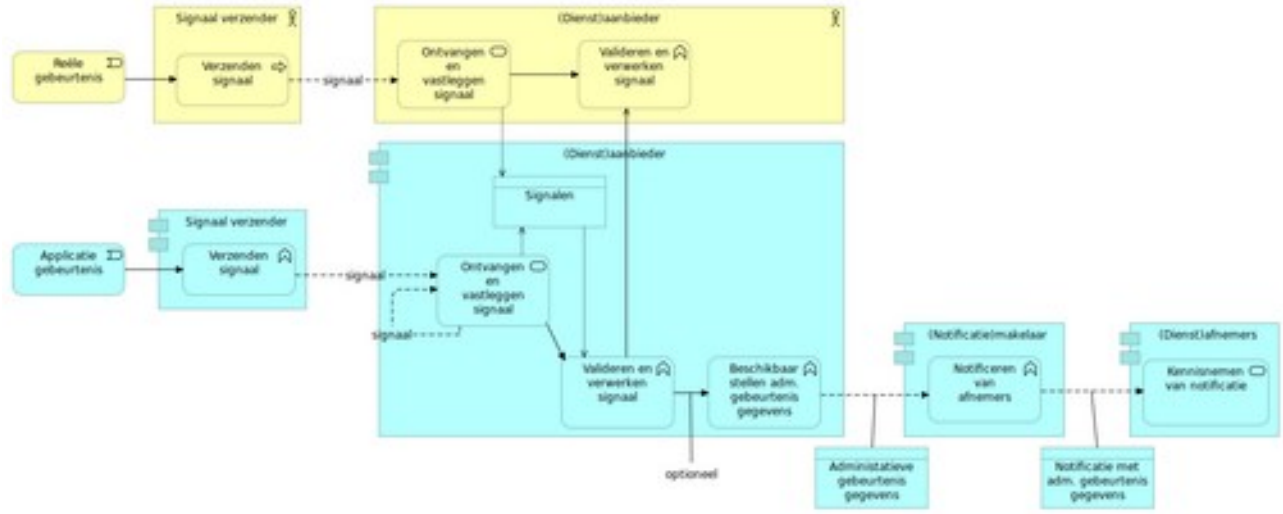
Breng vroegtijdig businessgebeurtenissen en daarbij behorende informatie in kaart

Met een event-driven mindset zal ook eerder worden nagedacht over de vraag of gebeurtenissen ook voor andere afdelingen of organisaties van belang kunnen zijn. Door hier vroegtijdig rekening mee te houden wordt het beter mogelijk om elkaar op een bruikbare manier vaker actief te informeren.

Breng vroegtijdig in kaart voor welke partijen buiten het eigen domein notificaties over businessgebeurtenissen belangrijk kunnen zijn

3.3 Reële en administratieve gebeurtenissen

Het woord ‘gebeurtenis’ wordt in het dagelijks spraakgebruik gebruikt om een voorval of feit te beschrijven. We noemen dit ‘een reële gebeurtenis’. Voordat informatie over een plaatsgevonden reële gebeurtenis als notificatie aan afnemers wordt verstrekt zijn een aantal processtappen te onderscheiden:



Er zijn verschillende manieren waarop dienstaanbieders op de hoogte kunnen komen van het feit dat er bepaalde gebeurtenissen hebben plaatsgevonden. Een ‘signaal verzender’ kan een ‘signaal’ geven dat er een reële gebeurtenis heeft plaatsgevonden. In veel gevallen is dit te zien als een ‘bewering’ die nog moet worden gevalideerd voordat tot verwerking wordt overgegaan. Zowel het verzenden van signalen als het valideren daarvan na ontvangst kan gebeuren door personen en applicaties. Valideren kan bijv. gebeuren door een baliemedewerker die een klant vraagt om bepaalde bewijsstukken of door een applicatie die de binnen een eFormulier ingevulde gegevens valideert.

Na geslaagde validatie wordt het ontvangen signaal verwerkt binnen de bedrijfs- en applicatieprocessen en vindt er een ‘administratieve gebeurtenis’ plaats. In vrijwel alle gevallen leidt die tot creatie of wijziging van gegevens in een bronregister. Een plaatsgevonden reële gebeurtenis wordt op deze manier ‘vertaald’ in een administratieve gebeurtenis die leidt tot vastlegging of wijziging van een aantal gegevens.

Naast externe gebeurtenissen waarover in de vorm van een signaal informatie binnenkomt kunnen interne administratieve gebeurtenissen ook leiden tot nieuwe signalen. Ook binnen de eigen organisatie of zelfs binnen 1 applicatie wordt zo ook gebeurtenisgedreven gewerkt en produceren (systeem)gebeurtenissen notificaties die door in- of externe processen worden verwerkt.

Eventuele notificaties die naar aanleiding van een plaatsgevonden administratieve gebeurtenis worden verstrekt aan afnemersapplicaties bevatten gegevens die de gebeurtenis en eventuele gevolgen daarvan beschrijven. De manier waarop reële gebeurtenissen binnen applicaties worden vertaald in administratieve gebeurtenissen zijn dus medebepalend hoe afnemers van notificaties over plaatsgevonden reële gebeurtenissen worden geïnformeerd.

3.4 Levensgebeurtenissen (life events)

Levensgebeurtenissen zijn belangrijke gebeurtenissen die zich tijdens het leven van een persoon kunnen voordoen. Bijvoorbeeld de geboorte van een kind, een huwelijk of het vinden of verliezen van een baan. Levensgebeurtenissen worden vaak gebruikt als zoekingang om mensen die te maken hebben met een

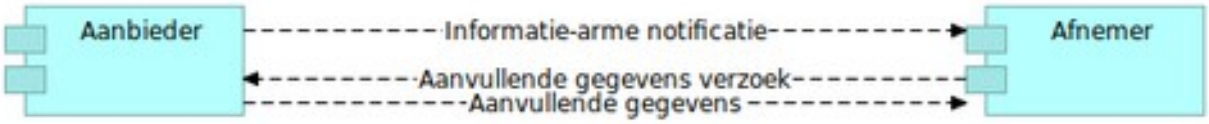
levensgebeurtenis informatie op maat aan te bieden (bijv. in dit overzicht levensgebeurtenissen) of om burgers en ondernemers een passende klantreis aan te bieden.

Levensgebeurtenissen zijn gebeurtenissen die gevolgen kunnen hebben op allerlei gebieden. Het feit dat een persoon is verhuisd of 18 jaar is geworden kan bijvoorbeeld relevant zijn voor allerlei soorten bedrijfsprocessen en registraties in verschillende domeinen. Binnen de context van project Notificatieservices worden ze gezien als een bijzonder soort reële gebeurtenissen waarbij het vaak belangrijk is om naar aanleiding van het optreden geïnteresseerde afnemers te kunnen notificeren. Binnen de overheid is er al lange tijd de wens is om burgers en ondernemers die een levensgebeurtenis meemaken een integrale klantreis aan te bieden waarbij meerdere overheidsorganisaties betrokken zijn. Dit blijkt in de praktijk niet eenvoudig realiseerbaar. Gebeurtenisgedreven werken kan hierbij bij uitstek een manier zijn om hier stappen in te zetten. Betrokken organisaties kunnen daarbij hun kerntaken zo goed mogelijk autonoom blijven uitvoeren maar kunnen elkaar via notificaties attenderen op het feit dat er gebeurtenissen hebben plaatsgevonden waar mogelijk actie op moet worden ondernomen.

3.5 Informatierijke en informatiearme notificaties

Onder ‘informatierijke notificaties’ verstaan we notificaties met daarin alle informatie die afnemers nodig over een plaatsgevonden gebeurtenis. De informatie volstaat normaal gesproken en er hoeft na ontvangst geen aanvullende informatie te worden opgevraagd. Informatierijke notificaties geven naast informatie dát zich een bepaalde gebeurtenis heeft voorgedaan dus ook informatie over wát dit aan gevolgen had. Bijv. door in de notificatie gegevens op te nemen van objecten die als gevolg van een gebeurtenis zijn gewijzigd. Het toegepaste patroon hierbij is ‘Event notification’ (zie verder).

Onder ‘informatiearme notificaties’ verstaan we notificaties waarin slechts minimaal informatie is opgenomen op basis waarvan afnemers alleen kunnen constateren dát zich een bepaalde gebeurtenis heeft voorgedaan. Voordat tot verwerking kan worden overgegaan moet de afnemer aanvullende informatie gaan opvragen. In de notificatie wordt hiervoor een link opgenomen die bijv. verwijst naar een service die de benodigde aanvullende gegevens kan leveren. Het toegepaste patroon hierbij is ‘Event-Carried State Transfer’ (zie verder)[7].



De hoeveelheid informatie in informatiearme en informatierijke notificaties kan verschillen. In informatiearme notificaties wordt vaak wat meer informatie opgenomen dan minimaal nodig is om melding te maken van het feit dat zich een gebeurtenis heeft voorgedaan. Het kan voor afnemers bijv. nodig zijn om te weten op welk object een gebeurtenis betrekking had om te kunnen beoordelen of verwerking, en het opvragen van aanvullende gegevens, nodig is. Hierbij ontstaat een mengvorm die soms aangeduid wordt met ‘hybride eventing’. Daarbij ontstaat een ongewenste koppeling tussen een aanbieder

[7]Het interactiepatroon waarbij een afnemer alleen een verwijzing krijgt naar de feitelijke inhoud staat ook bekend staat als het ‘Claim Check patroon’.

en de afnemers van notificaties omdat de aanbieder rekening moet houden met de wensen van, ook nieuwe, afnemers. Wordt gekozen voor informatiearm notificeren dan moet er daarom naar worden gestreefd om de opgenomen inhoudelijke gegevens zo beperkt en constant mogelijk te houden.

Neem in informatiearme notificaties de minimaal noodzakelijke inhoudelijke gegevens op.

De keuze voor gebruik van informatiearme of informatierijke notificaties is van verschillende factoren afhankelijk. In zijn algemeenheid geldt dat informatiearm notificeren lastiger is omdat er een extra opvraging nodig is waarbij zowel aanbieder als afnemers betrokken zijn. In situaties waarin informatierijk notificeren voldoet verdient het vanwege de eenvoudiger implementatie vaak de voorkeur (bijv. als een notificatie slechts een beperkt aantal voor iedereen toegankelijke open data bevat).

Notificeer informatierijk als er een beperkte hoeveelheid niet-vertrouwelijke gegevens worden verstrekt.

In bepaalde situaties verdient informatiearm notificeren de voorkeur. Bijv. als er sprake is van vertrouwelijke gegevens waarbij het wenselijk is om minimaal notificatiegegevens te verstrekken en bij opvraging passend bij de doelbinding van afnemers aanvullende gegevens te verstrekken. Hiermee wordt voorkomen dat er (te) veel vertrouwelijke gegevens in de vorm van notificaties worden verstrekt. Verder is het qua implementatie aantrekkelijk om authenticatie en autorisatie eenmalig te laten plaatsvinden door aanvullende gegevens te laten opvragen via al bestaande opvraagservices.

Een heel andere reden om informatiearm te notificeren kan bijv. zijn dat er grote hoeveelheden gegevens moeten worden verstrekt en dit op praktische problemen stuit. Bijv. omdat te gebruiken middleware of afnemersservices niet in staat zijn om berichten van grote omvang te verwerken.

Er kunnen dus verschillende redenen zijn om ondanks de grotere inspanningen, en soms kans op fouten, te kiezen voor informatiearm notificeren waarbij afnemers na ontvangst van een notificatie via een ander kanaal (‘out of band’) gegevens gaan ophalen.

Notificeer informatiearm als de situatie het nodig maakt (bijv. als er vertrouwelijke gegevens of grote hoeveelheden gegevens worden verstrekt).

4 ASPECTEN

4.1 INLEIDING

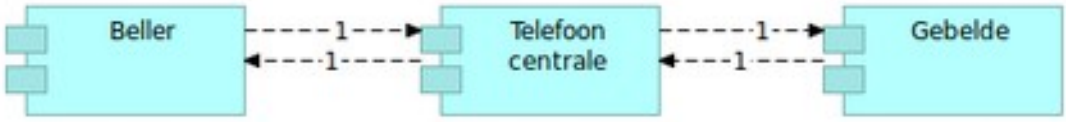
Dit hoofdstuk beschrijft een aantal aspecten waar bij het inrichten van notificatieprocessen rekening mee moet worden gehouden. Het gaat daarbij om 'kwaliteitsaspecten' die los staan van de feitelijke inhoud van notificaties maar die wel van belang voor het succesvol uitvoeren van notificatieprocessen. Daarbij kan het gaan om zaken zoals zoals veiligheid en betrouwbaarheid van uitvoering of aspecten die duurzaam gebruik mogelijk maken zoals aanpasbaarheid en schaalbaarheid.

Voor ieder aspect geldt dat er in de praktijk keuzemogelijkheden met eigen voor- en nadelen zijn. Er worden aanbevelingen gedaan met betrekking tot te maken keuzes, maar daarbij moet bedacht worden dat contextuele factoren sterk meebepalen welke keuzes mogelijk zijn.

Sommige van de hier genoemde aspecten worden in een apart hoofdstuk meer in detail beschreven.

4.2 SYNCHRONE EN ASYNCHRONE COMMUNICATIE

Communicatie verloopt *synchroon* of *asynchroon*. Bij synchrone communicatie vindt de communicatie tussen verschillende actoren *direct en gelijktijdig* plaats. Daarbij wordt een sessie opgezet die in stand blijft tot hij expliciet wordt verbroken. Een bekend voorbeeld hiervan is een telefoongesprek.



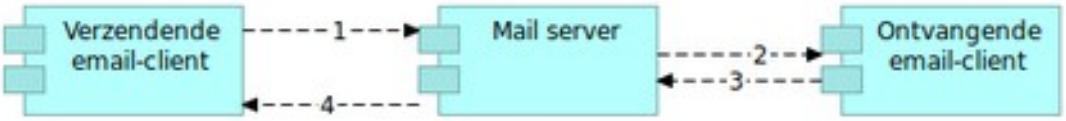
Synchrone communicatie waarbij partijen gelijkttijdig actief moeten zijn

Bij een synchroon notificatieproces verstuurt de aanbieder een notificatie naar een of meer afnemers en wacht op antwoorden van die afnemers. Aan de hand van het al dan niet ontvangen antwoord kan een aanbieder bijv. concluderen er nog vervolgacties nodig zijn. Wanneer er binnen een bepaalde tijd geen antwoord of een foutmelding komt moet hij kiezen wat hij gaat doen. Als de functionaliteit daarvoor aanwezig is kan hij bijvoorbeeld besluiten om notificaties met een bepaald tijdsinterval en een maximum aantal pogingen opnieuw te versturen.

Synchrone communicatie is relatief eenvoudig te implementeren maar kent ook een aantal nadelen. Met name omdat er ongewenste afhankelijkheden tussen applicaties ontstaan. Wanneer een van de betrokken applicaties tijdens het communicatieproces niet goed functioneert of niet actief is stokt het proces ('domino effect'). Naarmate meer applicaties met een bepaalde betrouwbaarheid zijn betrokken neemt de betrouwbaarheid van het proces steeds sneller af (99% * 99% * 99% = 97%). Door onvoldoende ontkoppeling ontstaat bij synchrone communicatie dus een kwetsbare situatie waarin het moeilijk is om notificatieprocessen betrouwbaar te laten verlopen. Zeker bij een gedistribueerd landschap met veel organisaties en applicaties moet rekening worden gehouden met het feit dat er fouten zullen gaan optreden ("Everything fails, all the time." Werner Vogels – CTO Amazon).

Gebruik van gespecialiseerde middleware kan de betrouwbaarheid van synchrone communicatie tot op zekere hoogte vergroten. Maar ook dan moeten applicaties speciale maatregelen nemen voor controle en foutafhandeling. Iets dat niet wenselijk en in veel gevallen ook niet realiseerbaar is en dus beter kan worden vermeden.

Bij asynchrone communicatie delen, in tegenstelling tot bij synchrone communicatie, applicaties de informatie niet gelijktijdig. Ze zijn daardoor ontkoppeld en hoeven ook niet op hetzelfde moment actief te zijn. Een voorbeeld hiervan is de uitwisseling van briefpost of email: verzender en ontvanger kunnen op verschillende momenten, op een moment dat zij zelf bepalen, via verschillende kanalen berichten lezen en verzenden.



Asynchrone communicatie waarbij partijen niet gelijkttijdig actief hoeven te zijn

Asynchrone systemen maken gebruik van 'store-and-forward' technieken om tijdverschillen te compenseren zodat applicaties op verschillende momenten informatie kunnen verwerken. Omdat ze onafhankelijk van elkaar kunnen functioneren wordt het mogelijk om notificatieprocessen flexibel, robuust en schaalbaar in te richten. Een aanbieder van gebeurtenisgegevens kan bijv. zijn taak, het beschikbaar stellen van informatie, autonoom uitvoeren zonder afhankelijk te zijn van het verwerken van notificaties door afnemers[8].

> *Asynchronous communication is fundamentally a pragmatic reaction to the problems of distributed systems. Sending a message does not require both systems both systems to be up and available at the same time. Furthermore, thinking about the communication in an asynchronous manner forces developers to recognize that working with a remote application is slower and prone to failure, which encourages design of components with high cohesion (lots of work locally) and low adhesion (selective work remotely). - Enterprise Integration Patterns*

We richten ons nu op notificatieprocessen waarbij 1-richting verkeer plaatsvindt (‘one-way-messaging’). Aanbieders hebben daarbij geen inhoudelijke antwoorden nodig van afnemers. Bij maximale ontkoppeling worden er géén antwoorden gegeven. Als er al antwoorden zijn gaat het om 'technische ontvangstbevestigingen’. Op basis daarvan kan de applicatie die notificeert bijv. concluderen dat zijn taak succesvol is afgerond en, afhankelijk van de afspraken, de notificatie kan verwijderen. Met het afgeven van een ontvangstbevestiging ligt de verantwoordelijkheid voor verwerking van de notificatie verder volledig bij de afnemer.

Door de betekenis van het retourbericht te beperken tot ‘ontvangen’ wordt het mogelijk om gespecialiseerde middleware te gebruiken en ontkoppeling aan de kant van de afnemer te realiseren. Toegestuurde berichten kunnen bijv. worden ontvangen en opgeslagen in een permanent actieve message queue waarna direct een ontvangstbevestiging kan worden verstuurd. De applicatie die de notificatie inhoudelijk gaat verwerken kan dit doen op een door de afnemer gekozen geschikte manier.

In lijn met de voorkeur voor asynchroon communiceren geldt ook voor ontvangstbevestigingen dat ze asynchroon en via een apart kanaal kunnen worden verstuurd. Zeker bij gebruik van gespecialiseerde middleware kan direct synchroon bevestigen van ontvangst (‘ACK') echter ook voldoende snel en betrouwbaar werken.

Communiceer bij notificeren asynchroon

[8] Afhankelijk van de inrichting van het notificatieproces kan de aanbieder een ontvangstbevestiging (‘ACK’) willen ontvangen. Is dat niet nodig dan volstaat voor de aanbieder het eenvoudige ‘fire and forget' patroon.

Voor het inrichten van notificatieprocessen via synchrone of asynchrone communicatie zijn verschillende protocollen beschikbaar. Sommige protocollen zijn met name geschikt voor synchrone communicatie (bijv. REST, RPC), anderen zijn met name bedoeld voor asynchrone communicatie (bijv. ebMS, AMQP). Deze worden in een apart hoofdstuk beschreven.

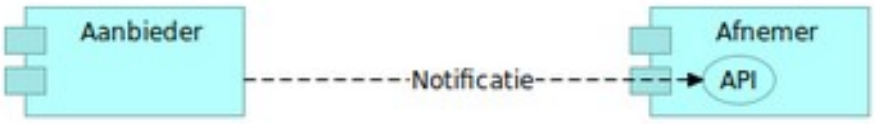
Er bestaat een breed scala aan middleware waarmee de rol van (gebeurtenis)makelaar (‘event broker') is in te vullen en asynchrone communicatie wordt ondersteund. Dit type middleware biedt zeer veel nuttige functionaliteit waarmee notificatieprocessen passend bij de behoeften kunnen worden ingericht. Notificaties kunnen bijv. tussentijds worden opgeslagen om fouten te kunnen herstellen of notificaties langer beschikbaar te houden. Of er kunnen hoge doorvoersnelheden van gegevens worden gerealiseerd of er kan juist als 'schokdemper' worden gefungeerd voor afnemers voor wie het tempo waarin notificaties beschikbaar komen te hoog is.

In lijn met het 'seperation of concerns’ principe geldt dat het, met uitzondering van zeer eenvoudige notificatieprocessen, als snel wenselijk is om voor de uitvoering van notificatieprocessen gebruik te maken van gespecialiseerde voorzieningen (‘message oriented middleware’). Binnen de context van event-driven werken wordt ook vaak gesproken over 'event-driven middleware’.

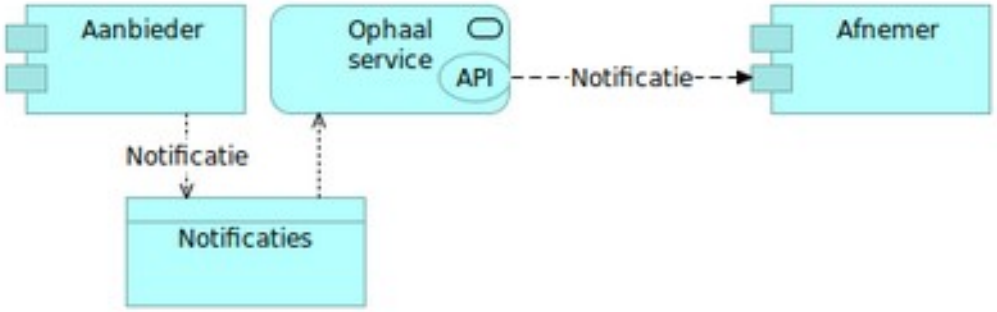
Gebruik message- of event-oriented middleware

4.3 Push en pull mechanismen

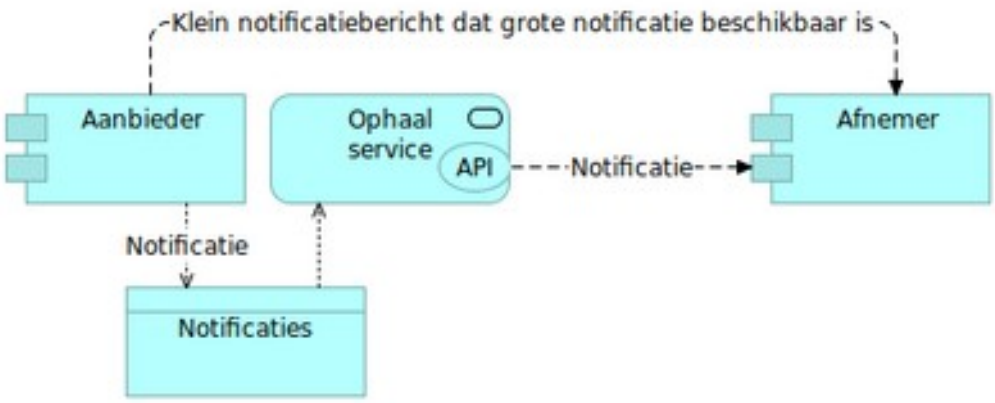
Notificaties kunnen door de aanbieder bij afnemers worden afgeleverd (‘push’). Dit kan bijv. wenselijk zijn als afnemers meteen genotificeerd willen worden als zich bepaalde gebeurtenissen hebben voorgedaan. De afnemer zal hiervoor in staat moeten zijn om notificaties te ontvangen. Bijv. via een service waar via een Application Programming Interface (API) een bericht is af te leveren.



Er kan ook voor een mechanisme worden gekozen waarbij de aanbieder notificaties op een afgesproken manier beschikbaar stelt en afnemers daarna zelf notificaties gaan ophalen (‘pull’). Dit kan bijv. wenselijk zijn als afnemers niet in staat zijn om notificaties te ontvangen. De aanbieder moet hierbij in staat zijn om afnemers notificaties te laten ophalen. Bijv. via een service waar via een API notificaties zijn op te vragen.



Ook een combinatie van ‘push’ en ‘pull’ is mogelijk. Bijv. om eerst een klein notificatiebericht te versturen (‘push’) met de melding dat een heel groot notificatiebericht kan worden opgehaald (‘pull’).



Een push-mechanisme heeft als, soms doorslaggevend, voordeel dat afnemers heel kort nadat zich een gebeurtenis heeft voorgedaan daarvan op de hoogte worden gesteld. Gebruik van push-mechanismen stelt echter wel de nodige eisen aan zowel aanbieder als afnemers. Aanbieders moeten bijvoorbeeld om kunnen gaan met verstoringen bij afnemers, waardoor direct afleveren niet mogelijk is en afnemers moeten op onvoorspelbare momenten, in het ideale geval 24x7, notificaties kunnen ontvangen. Aanbieder en afnemers moeten daarom goede afspraken maken. Ook over hoe wordt omgegaan met uitzonderingssituaties (bijv. als een afnemer notificaties in de verkeerde volgorde ontvangt) of fouten (bijv. als een afnemer notificaties mist en eerdere notificaties opnieuw wil ontvangen).

Gebruik een push-mechanisme als afnemers snel over notificaties moeten kunnen beschikken

Gebruik van een pull-mechanisme kan vergaande ontkoppeling van aanbieder en afnemers bewerkstelligen waardoor ze onafhankelijk van elkaar kunnen functioneren. Afnemers kunnen bijvoorbeeld zelf bepalen wanneer en hoe vaak zij informeren of er nieuwe notificaties zijn (‘pollen’). In tegenstelling tot bij push-mechanismen is het niet nodig dat zij te allen tijde notificaties kunnen ontvangen. Iets dat met name voor kleinere organisaties een uitdaging kan vormen. Een andere voordeel kan zijn dat afnemers als dat nodig is ook eerder verwerkte notificaties opnieuw kunnen inlezen (zo lang als de aanbieder ze beschikbaar stelt).

Hoewel bij notificeren meestal gedacht wordt aan gebruik van push-mechanismen zijn ook pull-mechanismen in de praktijk goed bruikbaar. Daarmee wordt het vaak eenvoudiger, of haalbaar, voor betrokken partijen om betrouwbare notificatieprocessen te realiseren. Waar afleveren van notificaties via push-notificaties niet noodzakelijk is kunnen pull-mechanismen vanwege hun relatieve eenvoud de voorkeur verdienen.

Gebruik een pull-mechanisme om eenvoudiger notificatieoplossingen te realiseren

Opmerkingen:

- In plaats van notificeren via ‘push’ of ‘pull’ wordt soms ook gesproken over ‘request-response of ‘event-based’ notificeren.
- Bij gebruik van zowel ‘push’ als ‘pull-mechanismen geldt het uitgangspunt dat aanbieders zo snel mogelijk nadat een gebeurtenis heeft plaatsgevonden notificaties verstrekken.

4.4 Ontkoppeling

Bij notificatieprocessen binnen een gedistribueerd en heterogeen landschap is het belangrijk dat organisaties en applicaties zoveel mogelijk ontkoppeld zijn en onafhankelijk van elkaar kunnen functioneren.

> *Losgekoppeld: afnemer en leverancier zijn maximaal onafhankelijk van implementatie van beide. Elke service is daarom autonoom. Er bestaat geen directe link of relatie tussen verschillende services. – bron: Wikipedia*

Ontkoppeling is bij een aantal aspecten rondom notificeren van belang:

- **Tijd** – Om te notificeren hoeven niet alle betrokken applicaties gelijktijdig actief te zijn. Dit vereist een vorm van 'bufferen' waarbij de aanbieder geen onmiddellijk antwoord nodig heeft. Bijv. via toepassing van asynchrone communicatie en gebruik van queues.
- **Plaats** – Betrokken applicaties moeten zich op verschillende plaatsen kunnen bevinden (bijv. on-premise of in een cloudomgeving) en verplaatst kunnen worden (bijv. m.b.v. opzoekfuncties).
- **Tempo:** Aanbiedende applicaties die in hoog tempo notificaties verstrekken mogen geen last hebben van afnemende applicaties notificaties niet in dat tempo kunnen verwerken.
- **Versie** – Notificeren moet doorgaan als de aanbiedende applicatie wordt gewijzigd; afnemende applicaties moeten zich houden zich aan de eisen die een versie kent.
- **Kardinaliteit** – Aanbiedende applicaties hoeven geen rekening te houden met het aantal applicaties dat notificaties afneemt. Bijvoorbeeld door het verstrekken van notificaties te beleggen bij een gespecialiseerde applicatie (al dan niet beheerd door een andere organisatie).
- **Vindbaarheid** – Aanroep van services vindt plaats via logische namen en adressen die vindbaar zijn in catalogi.
- **Technologie** – Betrokken applicaties kunnen van verschillende technologieën gebruik maken (bijv. door van overeengekomen standaarden gebruik te maken).
- **Protocol** – Applicaties kunnen verschillende protocollen gebruiken (bijv. omdat een makelaar notificeren met gebruik van meerdere protocollen ondersteunt).
- **Formaat** – Notificeren is mogelijk bij applicaties die andere berichtformaten gebruiken (bijv. door een makelaar translatie van berichten te laten uitvoeren).
- **Interface** – Interfaces zijn gestandaardiseerd zodat het voor nieuwe applicaties eenvoudig is om deel te nemen aan een notificatieproces.

Bij notificatieprocessen is maximale ontkoppeling van aanbieder en afnemers wenselijk. In de praktijk zullen de mogelijkheden van betrokken organisaties en applicaties meebepalen voor welke aspecten, en in welke mate ontkoppeling haalbaar is.

Streef bij notificatieprocessen naar maximale ontkoppeling van applicaties

Om ontkoppeling te kunnen realiseren zijn goede afspraken en gebruik van standaarden nodig. Bij voorkeur standaarden die vaak gebruikt worden zodat er veel kennis over beschikbaar is en partijen laagdrempelig deel kunnen gaan nemen aan notificatieprocessen.

Ontkoppel door veelgebruikte standaarden toe te passen

Realiseren van maximale ontkoppeling en voldoende betrouwbaarheid is niet eenvoudig. Het wordt daarom al snel wenselijk om gebruik te gaan maken van gespecialiseerde voorzieningen voor het invullen van de makelaarrol.

Ontkoppel door gespecialiseerde componenten de rol van makelaar te laten invullen

4.5 Betrouwbaarheid

De context bepaalt wat het gewenste of vereiste betrouwbaarheidsniveau is waarmee notificaties voor afnemers beschikbaar moeten komen. Er zijn situaties waarin een afnemer het geen probleem vindt als om af en toe een notificatie te missen. Maar bij gegevensuitwisseling tussen overheidsorganisaties zal er vaak een hoog betrouwbaarheidsniveau nodig zijn. Bijvoorbeeld als notificaties betrekking hebben op plaatsgevonden levensgebeurtenissen waarbij persoonsgegevens betrokken zijn en waar vaak rechtskracht aan kan worden ontleend.

Realiseren van hoge betrouwbaarheid vereist speciale maatregelen zoals gebruik van geschikte standaarden (bijv. Digikoppeling) en voorzieningen (bijv. 'event-oriented middleware'). Daarmee kunnen vergaande garanties worden geboden voor bijv. gegarandeerde bezorging van berichten en het kunnen opvangen van uitzonderings- of foutsituaties (bijv. via retry-mechanismes en gebruik van deadletter-queues).

Als wordt gewerkt met stijlen, standaarden en voorzieningen die inherent geen hoge betrouwbaarheid kennen zijn aanvullende maatregelen nodig bij zowel aanbieder als afnemer. Bijvoorbeeld door in bedrijfsapplicaties controles in te bouwen of zich fouten in berichtuitwisseling hebben voorgedaan (bijv. door bij ieder bericht te controleren of volgnummers van ontvangen berichten wel opeenvolgend zijn). Naarmate het vereiste betrouwbaarheidsniveau toeneemt kan dit al snel heel complex worden en is het niet meer wenselijk om dit type functionaliteit in bedrijfsapplicaties te bouwen.

In termen van communicatielagen verdient het de voorkeur om bijv. functionaliteit voor betrouwbaar transport niet binnen de applicatielaag maar in lagen daaronder te beleggen via gebruik van daarvoor bedoelde standaarden en voorzieningen.

Realiseer hoge betrouwbaarheid door middel van geschikte standaarden en voorzieningen

Opmerking: In het hoofdstuk over 'protocollen' wordt beschreven welke protocollen geschikt voor het realiseren van hoge betrouwbaarheid.

4.6 Veiligheid (uit te werken)

Notificatieprocessen zijn een bijzondere vorm van processen waarbij gegevens worden uitgewisseld. De aard van de bij uitwisseling betrokken gegevens bepaalt welk niveau van beveiliging nodig is. Daarbij zijn dezelfde mechanismen voor beveiliging zijn toe te passen als bij andere vormen van gegevensuitwisseling. Er kan bijv. gebruik worden gemaakt van beveiligde verbindingen (bijv. via 1- of 2-zijdig TLS authenticatie), gebruik van speciale certificaten (bijv. PKIoverheid), besloten netwerken (bijv. Diginetwerk) of standaarden die een aantal aspecten omvatten (bijv. Digikoppeling).

Meer algemeen geldt dat het veilig inrichten van notificatieprocessen op een aantal gebieden maatregelen vereist waarbij de BIO als kader kan worden gebruikt. Bijv. om in lijn met de BIO te bepalen welk van de drie basisbeveiligingsniveaus van toepassing is om op basis van een risicoafweging te bepalen hoe moet worden voldaan aan de gestelde beveiligingsdoelstellingen van verschillende controls. Daarmee kan

worden geborgd dat onderling uitgewisselde gegevens, in lijn met wet- en regelgeving, passend beveiligd zijn.

Om te voorkomen dat uitwisseling van gegevens niet veilig genoeg verloopt moet zo vroeg mogelijk in het ontwerpproces aandacht worden geschonken aan beveiligingsaspecten.

Schenk tijdens ontwerp van notificatieprocessen vroegtijdig aandacht aan beveiliging (‘security by design’)

Voor het aspect veiligheid volstaan we met het verwijzen naar de binnen de overheid gebruikelijke eisen en hulpmiddelen om gegevensuitwisseling passend te beveiligen.

4.7 Privacy (uit te werken)

Wanneer bij notificeren op personen herleidbare gegevens betrokken zijn moet worden gezorgd dat voldaan wordt aan privacywetgeving. Een onderdeel hiervan is om te borgen dat partijen die de rol van makelaar of afnemer vervullen geen gegevens kunnen inzien waarvoor geen doelbinding aanwezig is. Hoe eerder in het ontwerp hier aandacht voor is, hoe groter de kans dat privacy binnen processen is te borgen. Maar aandacht hiervoor blijft nodig. Bijv. bij het samenstellen en toestaan van abonnementen door afnemers. Iets waarvoor in de praktijk bijv. goede afspraken nodig zijn over verantwoordelijkheid rondom gegevensgebruik tussen partijen in de rol van aanbieder, makelaar en afnemer.

Schenk tijdens ontwerp en uitvoering van notificatieprocessen vroegtijdig aandacht aan privacy (‘privacy by design’)

5 PUBLISH-SUBSCRIBE NOTIFICATIEPATROON

5.1 INLEIDING

Dit hoofdstuk beschrijft de betekenis en rol van 'ontwerppatronen': herbruikbare vormen van oplossingen voor vergelijkbare ontwerpproblemen.

Patronen kunnen in meerdere opzichten van elkaar verschillen:

- Doel: patronen zijn voor allerlei doeleinden bruikbaar; we focussen hier op patronen die behulpzaam zijn om oplossingen te ontwerpen voor gebeurtenisgedreven samenwerken in het algemeen en notificeren in het bijzonder.
- Aard: het ene type patroon beschrijft bijv. de manier waarop interactie tussen applicaties moet plaatsvinden terwijl een ander patroon een implementatievorm daarvan beschrijft.
- Abstractieniveau: het ene patroon beschrijft uitwisseling op logisch niveau, het andere patroon op fysiek niveau.
- Modulariteit: een patroon kan een aantal patronen combineren tot een nieuw patroon of een patroon kan juist een specifieke uitwerking van een bepaald patroon zijn.

Er zijn meerdere patronen bruikbaar bij het ontwerpen van notificatieoplossingen. Het Publsh-Subscribe patroon is echter met afstand het meest bekende en toegepaste patroon. In de kern beschrijft het patroon hoe aanbieders informatie over plaatsgevonden gebeurtenissen kunnen publiceren ('Publish') en afnemers zich kunnen abonneren ('Subscribe') om over bepaalde type gebeurtenissen genotificeerd te worden. We beschouwen dit patroon op dit moment (ook) voor de overheid als het meest bruikbare patroon voor het, op onderdelen gestandaardiseerd, ontwerpen van notificatieoplossingen. We korten het Publish-Subscribe patroon verder af als het 'PubSub patroon'.

Net zoals bij andere patronen geldt dat het patroon in de praktijk niet altijd eenduidig wordt benoemd en beschreven. Verstrekken van notificaties aan afnemers via een push-mechanisme wordt bijv. soms wel en soms niet gezien als onderdeel van het patroon. Soms wordt ook als eis gesteld dat de makelaarrol door een aparte applicatie moet worden ingevuld. Dit hoofdstuk beoogt om tot overeenstemming te komen over wat we binnen de overheid precies onder het PubSub patroon verstaan. Vervolgens wordt beschreven hoe daarbij op verschillende manieren standaardisatie mogelijk is.

5.2 ONTWERPPATRONEN

In de praktijk zijn er altijd verschillende soorten oplossingen mogelijk om bepaalde problemen op te lossen. Kennis en ervaring daarmee in de praktijk kan leiden tot een aantal bewezen bruikbare '(ontwerp)patronen'. Zo'n patroon kan vervolgens als startpunt ('sjabloon') worden gebruikt om binnen vergelijkbare situaties goede oplossingen te ontwerpen:

> *A design pattern is the re-usable form of a solution to a design problem – bron: Wikipedia*

Een georganiseerde verzameling ontwerppatronen die betrekking hebben op een bepaald gebied, wordt een patroontaal ('pattern language') genoemd:

> *The elements of this language are entities called patterns. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution*

> *to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. – Christopher Alexander, A Pattern Language*

Een patroontaal levert een algemene terminologie voor het bespreken van de situaties waarmee ontwerpers worden geconfronteerd. Binnen het kader van het project Notificatieservices gaat het om het komen tot goede oplossingen voor het notificeren van applicaties naar aanleiding van plaatsgevonden gebeurtenissen en meer algemeen het ondersteunen van gebeurtenisgedreven samenwerken. Eerder is beschreven welke terminologie we gebruiken om notificeren binnen de overheid te beschrijven. Dit hoofdstuk werkt dit verder uit door te beschrijven wat we als belangrijkste patronen zien om notificatieprocessen goed in te kunnen richten.

5.3 Interactiepatronen

Bij gebruik van software is er sprake van interactie binnen een applicatie en van interactie met andere applicaties. Interactiepatronen zijn bruikbaar om oplossingen te ontwerpen waarbij sprake is van een specifieke vorm van interactie.

Basale patronen om interactie te beschrijven zijn bijvoorbeeld:

- *Observer: Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically*
- *Mediator: Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.*
- *Chain of Responsibility: Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.*
- *Command: Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.*

> *bron: Design Patterns: Elements of Reusable Object-Oriented Software – Gamma, Helm, Johnson, Vlissides, 1994*

Bij notificeren is sprake van een specifieke vorm van interactie tussen actoren met de rollen van aanbieder, makelaar en afnemer. Het PubSub patroon is te zien als een uit deze basispatronen opgebouwd interactiepatroon[9], bruikbaar voor het ontwerpen van notificatieoplossingen. Binnen de context van project Notificatieservices volstaat het om het PubSub patroon meer in detail te beschrijven. Dit laat onverlet dat genoemde, meer basale, patronen soms ook bruikbaar kunnen zijn om goede oplossingen te ontwerpen voor bijv. de interactie tussen aanbieder en makelaar of die tussen makelaar en verschillende afnemers.

5.4 Het Publish-Subscribe notificatiepatroon

Bij notificatieprocessen is er sprake van een specifieke vorm van interactie tussen actoren. Binnen de context vaan project Notificatieservices, zie ook paragraaf ‘Scope’ in het hoofddocument, zien we het PubSub patroon als het meest bruikbare interactiepatroon omdat het:

[9]Wanneer aanbieder en afnemers wel direct met elkaar communiceren is sprake van het ‘observer-patroon’. Wanneer dit wordt gecombineerd met het ‘mediator-patroon’ om ontkoppelde interactie te ondersteunen ontstaat het PubSub patroon. “Mediator (305) and Observer (326) are competing patterns. The difference between them is that Observer distributes communication by introducing Observer and Subject objects, whereas a Mediator object encapsulates the communication between other objects.” – bron: Design Patterns – Elements of Reusable Object-Oriented Software – Gamma, Helm, Johnson, Vlissides, 1994

- een geschikt patroon is om informatie over gebeurtenissen te delen tussen losgekoppelde aanbieders en een onbeperkt aantal afnemers;
- een bekend, bewezen en vaak toegepast patroon is;
- een aantal goed bruikbare implementatiepatronen kent;
- brede technische ondersteuning kent in de vorm van gespecialiseerde applicaties (‘middleware’).

Gebruik het Publish-Subscribe patroon voor het ontwerpen van notificatieoplossingen

Net als voor andere patronen geldt ook voor het PubSub patroon dat het geen kookboekrecept is dat klakkeloos gevolgd kan worden. Bij het ontwerpen van oplossingen moeten nog tal van ontwerpkeuzes worden gemaakt. Een volgend hoofdstuk beschrijft een aantal ‘implementatiepatronen’ die voortbouwen op het het PubSub patroon en specifieke implementatiekeuzes maken.

In de praktijk worden vaak verschillende dingen bedoeld met de term ‘PubSub’. Reden om eerst te beschrijven wat we er precies onder verstaan. In lijn met het uitgangspunt om zoveel mogelijk aan te sluiten bij wat wereldwijd gebruikelijk is gebruiken we de Wikipedia omschrijving van het ‘Publish–Subscribe pattern’ als uitgangspunt om te komen tot afspraken over wat we er binnen de overheid wel en niet onder verstaan.

Definitie volgens Wikipedia:

> *In software architecture, publish–subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are. – bron: Wikipedia*

Zoals eerder aangegeven verdient messaging als integratiestijl de voorkeur bij notificeren, ook bij gebruik van het PubSub patroon. We willen de toepassing van het patroon echter niet beperken tot alleen gebruik van messaging omdat zich ook situaties voordoen waarin gebruik van messaging niet mogelijk is. Bijv. als organisaties geen messaging kunnen ondersteunen maar bijv. wel bestanden met notificatie-informatie via file-transfer kunnen uitwisselen. We zien het PubSub patroon daarom als een bruikbaar patroon voor notificeren waarbij messaging weliswaar de voorkeur verdient maar ook gebruik van andere stijlen mogelijk is.

PubSub: Gebruik bij voorkeur messaging als integratiestijl maar gebruik ook andere stijlen waar dit nodig is.

We onderschrijven het belang van maximale ontkoppeling tussen aanbiedende en afnemende *applicaties* die elkaar idealiter niet eens hoeven te kennen.

PubSub: Ontkoppel aanbiedende en afnemende applicaties maximaal van elkaar

Waar het gaat om *organisaties* zal binnen de overheid vaak een bepaalde mate van koppeling nodig zijn omdat aanbiedende en afnemende organisatie afspraken moeten maken over het notificatieproces (bijv. wanneer er vertrouwelijke gegevens worden verstrekt). Toch geldt ook voor organisaties dat het nuttig is

om naar ontkoppeling te streven zodat zij zo autonoom mogelijk kunnen functioneren en zich op hun kerntaken kunnen richten. Voorbeelden daarvan zijn het beleggen van makelaartaken bij een derde partij (bijv. Logius die via de Digilevering voorziening bemiddelt tussen aanbieders van basisregistratiegegevens en afnemende organisaties) en het beleggen van de verantwoordelijkheid voor toegestaan gebruik van notificaties bij afnemende partijen.

PubSub: Ontkoppel aanbiedende en afnemende organisaties maximaal van elkaar

Filtering volgens Wikipedia:

> *In the publish-subscribe model, subscribers typically receive only a subset of the total messages published. The process of selecting messages for reception and processing is called filtering. There are two common forms of filtering: topic-based and content-based.*
>
> - *In a **topic-based** system, messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe. The publisher is responsible for defining the topics to which subscribers can subscribe.*
> - *In a **content-based** system, messages are only delivered to a subscriber if the attributes or content of those messages matches constraints defined by the subscriber. The subscriber is responsible for classifying the messages. – bron: Wikipedia*

Gebeurtenisgegevens moeten worden gefilterd om de door afnemers gewenste notificaties te kunnen verstrekken. Topic-based filtering is daarbij een goed bruikbaar en vaak toegepast middel.

Pubsub: Gebruik topic-based filtering voor het op maat verstrekken van notificaties

Bij voorkeur wordt geen gebruik gemaakt van content-based filtering. Redenen om gebruik daarvan te vermijden is dat daarmee een duidelijke scheiding van verantwoordelijkheden mogelijk wordt waarbij content ('payload') een zaak is van aanbieder en afnemers en níet van de makelaar. Het wordt hierdoor beter mogelijk om veiligheid en privacy bij gegevensuitwisseling te borgen[10]. Het sluit ook aan bij best-practice om voor zaken als filtering en routering gebruik te maken van voor dit doel opgenomen metadata.

PubSub: Gebruik geen content-based filtering

Het PubSub patroon wordt in de praktijk zowel gebruikt voor situaties waarin autorisatie van afnemers geen rol speelt ('alle afnemers mogen alles ontvangen') en voor situaties waarin autorisatie wel een rol speelt ('afnemers mogen alleen krijgen waar ze recht op hebben'). Bij notificeren binnen de overheid zijn vaak vertrouwelijke gegevens betrokken. Om te kunnen borgen dat er sprake is van voldoende doelbinding voor een specifieke vorm van filtering zijn daarbij aanvullende maatregelen nodig. Aanvullend op wat in Wikipedia staat onderscheiden we een tweetal mechanismen om dit te ondersteunen.

Een veel gebruikt mechanisme is om afnemers zelf te laten aangeven voor welke objecten, met name personen, zij notificaties mogen en willen ontvangen. De verantwoordelijkheid voor het conform

[10]De argumenten om geen content-based filtering te gebruik gelden zeker als de rol van aanbieder en makelaar bij verschillende organisaties zijn belegd. Maar ook als de rollen bij dezelfde partij zijn belegd verdient het de voorkeur om een logische scheiding tussen beide rollen te maken en content-based filtering te vermijden.

doelbinding specificeren van objecten waarop gefilterd gaat worden ligt hierbij expliciet bij de afnemer. We hebben dit eerder benoemd als 'objectfiltering'.

Pubsub: Gebruik objectfiltering om filteren conform doelbinding te realiseren

Een tweede vorm van filtering om een vorm van autorisatie te realiseren is 'attribuut-filtering'. Hierbij worden notificaties samengesteld uit beschikbare attributen waarbij rekening kan worden gehouden met de voor afnemer(s) geldende autorisatie.

Pubsub: Gebruik atribuutfiltering om filteren conform doelbinding te realiseren

Los van de precieze taakverdeling tussen aanbieder en makelaar geldt dat de aanbieder eindverantwoordelijk is om er voor te zorgen dat gegevensverstrekking conform doelbinding plaatsvindt. Het verdient daarom de voorkeur om bedrijfslogica voor verantwoorde filtering bij de aanbieder onder te brengen[11]. Een aanbieder kan naar aanleiding van 1 type gebeurtenis bijv. verschillende typen berichten aanleveren (met verschillen in zowel header- als payloadgegevens) die naar verschillende topics worden gepubliceerd. De functie van de makelaarapplicatie blijft daarmee beperkt tot meer technische functies zoals het conform afnemerswensen distribueren van notificaties ('smart endpoints, dumb pipes').

PubSub: Beleg attribuutfiltering voor autorisatie bij de aanbieder

Opmerking: attribuutfiltering is ook bruikbaar om op maat gesneden notificaties conform afnemerswensen te leveren. Daarbij kunnen afnemers zelf specificeren welke attributen in een notificatie moeten worden opgenomen.

Topologie volgens Wikipedia:

> *In many pub/sub systems, publishers post messages to an intermediary message broker or event bus, and subscribers register subscriptions with that broker, letting the broker perform the filtering. The broker normally performs a store and forward function to route messages from publishers to subscribers. In addition, the broker may prioritize messages in a queue before routing. – bron: Wikipedia*

Gegevensuitwisseling in een gedistribueerd gegevenslandschap kan om tal van redenen fout gaan. Het is niet eenvoudig om robuuste notificatieoplossingen te maken. Het is daarom wenselijk om de makelaarrol te laten vervullen door gespecialiseerde software die o.a. berichten van aanbieder naar afnemers kan bufferen en optredende foutsituaties correct kan afhandelen. Naast een hogere betrouwbaarheid ontlast dit aanbieder en afnemers waardoor zij zich kunnen richten op hun kernfunctionaliteit.

PubSub: gebruik gespecialiseerde applicaties om de makelaarrol in te vullen

> *Subscribers may register for specific messages at build time, initialization time or runtime. – bron: Wikipedia*

Een afnemer kan op verschillende momenten een abonnement op notificaties afsluiten. Binnen de huidige projectcontext, waarbij het gaat om notificeren door en van applicaties, geldt vaak dat afnemers eerst aan

[11] In sommige situaties het niet alleen wenselijk maar noodzakelijk omdat alleen de aanbieder in staat is om inhoudelijk te beoordelen voor welke gegevens een afnemer is geautoriseerd.

een aantal formele voorwaarden moeten voldoen. Bijv. omdat er eerst een gegevensleveringsovereenkomst moet worden afgesloten. Er zijn echter ook situaties waarin (delen van) het abonneren geautomatiseerd kan plaatsvinden. Bijv. als een aanbieder openbare gegevens beschikbaar stelt en iedereen die dat wil zich daar op mag abonneren. Dan geldt dat het wenselijk is om als het afsluiten en beëindigen van een abonnement geautomatiseerd kan plaatsvinden (bijv. om afnemers te kunnen laten aangeven voor welke gebeurtenissoorten en op welke manier hij notificaties wil ontvangen). Bij voorkeur verloopt dit gestandaardiseerd (bijv. zoals bij het later te beschrijven WebSub-patroon).

PubSub: automatiseer, bij voorkeur gestandaardiseerd, het afsluiten en beëindigen van abonnementen op notificaties

Voordelen volgens Wikipedia:

> *Ontkoppeling: Publishers are loosely coupled to subscribers, and need not even know of their existence. With the topic being the focus, publishers and subscribers are allowed to remain ignorant of system topology. Each can continue to operate as per normal independently of the other. – bron: Wikipedia*

Het PubSub patroon biedt veel mogelijkheden om aanbieders en afnemers op een aantal aspecten van elkaar te ontkoppelen (bijv. tijd, plaats, tempo, kardinaliteit, technologie, protocol, formaat, interface). Zeker naarmate het aantal afnemers toeneemt neemt het belang van ontkoppeling op zo veel mogelijk aspecten toe.

PubSub: maak keuzes waardoor aanbieder en afnemers op zoveel mogelijk aspecten maximaal worden ontkoppeld

> *Schaalbaarheid: Pub/sub provides the opportunity for better scalability than traditional client-server, through parallel operation, message caching, tree-based or network-based routing, etc. – bron: Wikipedia*

Zeker bij gebruik van gespecialiseerde applicaties in de rol van makelaar geldt dat het PubSub patroon geschikt is om schaalbaarheid mee te realiseren. Naast al langer bekende middleware-applicaties leveren nu ook alle grote cloudplatforms makelaarapplicaties die het PubSub patroon ondersteunen waardoor bijv. automatisch schalen mogelijk is. Het wordt daarmee steeds eenvoudige om notificatieprocessen, ook bij zeer grote aantallen afnemers en berichten, snel en robuust te laten functioneren.

PubSub: gebruik makelaarapplicaties die, bij voorkeur automatisch, schaalbaar zijn

Nadelen volgens Wikipedia:

> *The most serious problems with pub/sub systems are a side-effect of their main advantage: the decoupling of publisher from subscriber. – bron: Wikipedia*

In zijn algemeenheid geldt dat ontkoppeling en gebruik van een gedistribueerd applicatielandschap wenselijk maar ook uitdagend is omdat er zowel technische als logische fouten zullen optreden. Applicaties kunnen tijdelijk niet beschikbaar zijn, er kunnen netwerkstoringen optreden, berichten kunnen verminkt of kwijt raken, applicaties kunnen overbelast raken, etc.. Zowel bij aanbieder, makelaar als afnemers zijn maatregelen nodig om hier goed mee om te gaan. Naarmate het vereiste betrouwbaarheidsniveau

toeneemt wordt dit al snel (te) complex en is het raadzaam om foutafhandeling zoveel mogelijk te beleggen bij hierin gespecialiseerde makelaarvoorzieningen.

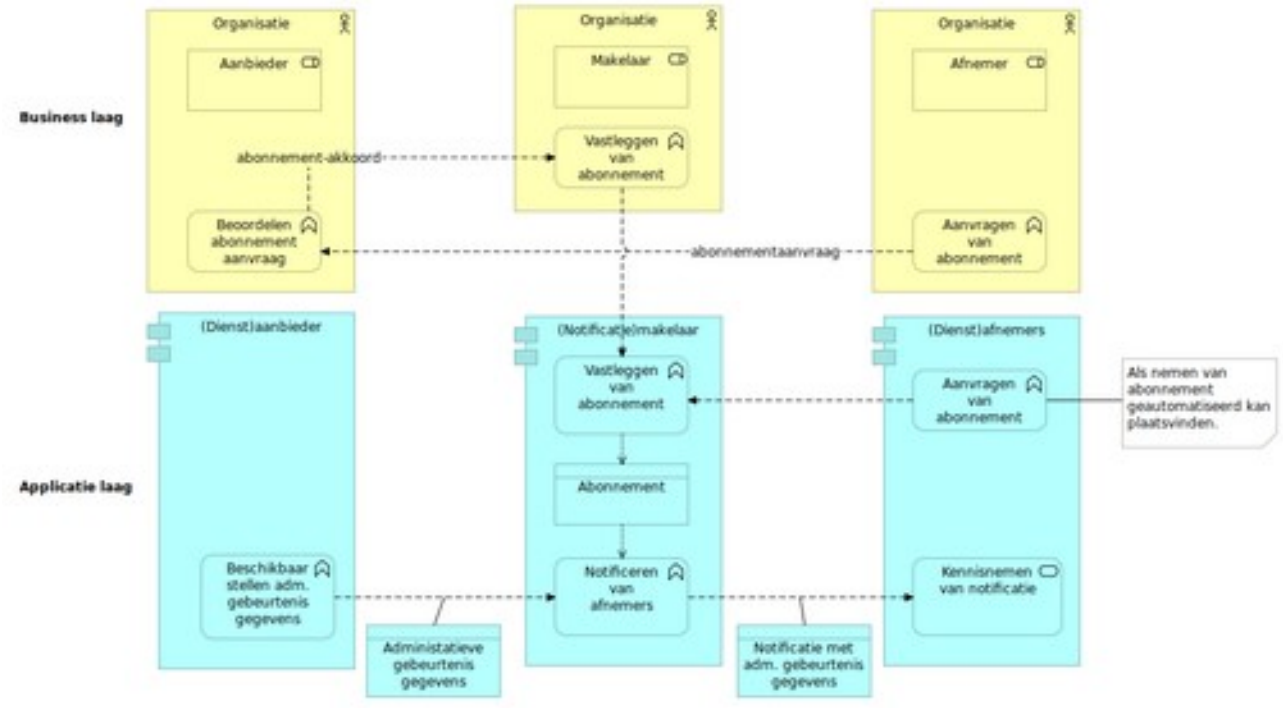
PubSub: gebruik gespecialiseerde voorzieningen voor het afhandelen van optredende communicatiefouten

Met de Wikipedia-definite van het PubSub patroon als uitgangspunt komen we naar aanleiding van bovenstaande aanbevelingen tot de volgende definitie van het PubSub patroon:

> *Het Publish Subcribe patroon (afgekort PubSub patroon) is een patroon waarbij aanbieders en afnemers van elkaar ontkoppeld zijn en aanbieders via een makelaar gegevens over plaatsgevonden gebeurtenissen verstrekken aan afnemers die hierop geabonneerd zijn.*
>
> *Bij gebruik van het PubSub patroon zijn verschillende keuzes mogelijk wat betreft implementatieaspecten (bijv. synchrone en/of asynchrone communicatie, push en/of pull mechanismen, diverse protocollen).*

In tegenstelling tot wat soms gebeurt kiezen we er bewust voor om in de patroondefinitie geen voorkeur voor implementatieaspecten op te nemen. Asynchrone berichtuitwisseling via push-mechanismen wordt bijv. regelmatig beschreven als vereiste om te kunnen spreken over PubSub patroon. We kiezen ervoor om *aanbevelingen* te doen over gebruik van het patroon omdat de projectscope zo breed is dat we te maken hebben met een zodanig diverse omgeving dat het niet altijd mogelijk is om conform voorkeuren te werken.

In de afbeelding hieronder is het patroon globaal weergegeven waarbij zichtbaar is dat notificatieprocessen zowel organisatorische als applicatietechnische aspecten kennen.



Toelichting:

- Het afsluiten van een abonnement zal vaak afspraken en handmatige acties vereisen, maar het kan ook (deels) geautomatiseerd verlopen;
- Niet afgebeeld is de mogelijkheid voor afnemers om een abonnement te kunnen beëindigen;
- Een organisatie kan een of meer rollen vervullen (aanbieder, makelaar, afnemer);
- De afbeelding geeft de eenvoudigste situatie weer met 3 actoren maar wanneer organisaties/applicaties elkaar van gebeurtenisgegevens en/of notificaties voorzien en verschillende rollen gaan vervullen kan een netwerk ontstaan waarbinnen informatie over plaatsgevonden gebeurtenissen stroomt (‘event mesh’).

6 STANDAARDEN VOOR PUBSUB

6.1 INLEIDING

Net zoals op andere terreinen kan het gebruik van standaarden ook bij gebeurtenisgedreven werken in het algemeen, of specifiek bij toepassing van het PubSub patroon, veel voordelen bieden. Gebruik van zo breed mogelijk geaccepteerde standaarden bevordert de samenwerking tussen organisaties, de interoperabiliteit van applicaties en het tempo waarin ze ontwikkeld worden.

Dit hoofdstuk beschrijft drie standaarden die helpen om gebeurtenisgedreven, en op het PubSub patroon gebaseerde, oplossingen binnen de overheid te standaardiseren. De drie standaarden hebben een verschillend karakter en een eigen standaardisatiedoel:

1 De CloudEvents berichtstandaard, uitgebreid met een aantal eigen extensies, beschrijft met name hoe gebeurtenisberichten en notificaties er uit moeten zien.

2 De WebSub interactiestandaard schrijft voor hoe bij gebruik van het HTTP protocol interactie tussen applicaties er op onderdelen uit moet zien.

3 De AsyncAPI beschrijvingsstandaard zegt hoe gebeurtenisgedreven diensten en API's moeten worden beschreven.

Onderstaande tabel toont een aantal kenmerken per standaard.

	Bericht inhoud	Interactie	Be-schrijvend	Inter-nationaal	Multi protocol	Tooling beschikbaar	Brede adoptie	In ont-wikkeling
CloudEvents	X			X	X	X	X	x
WebSub		X		X				
AsyncAPI			X	X	X	X	X	x

6.2 CLOUDEVENTS MET NL-EXTENSIONS BERICHTSTANDAARD

Een van de belangrijkste doelen van project Notificatieservices is om voor notificeren binnen de Nederlandse overheid een concept berichtstandaard te ontwikkelen. In lijn met het uitgangspunt om maximaal aan te sluiten bij internationale standaarden is er voor gekozen om gebruik te maken van de internationale CloudEvents standaard. Deze nog in ontwikkeling zijnde standaard heeft als doel om gegevensattributen van berichten over plaatsgevonden gebeurtenissen en de uitwisseling daarvan te standaardiseren.

De standaard wordt ontwikkeld vanuit de behoefte om binnen en tussen cloudomgevingen gestandaardiseerd informatie over gebeurtenissen uit te kunnen wisselen. Ontwikkeling gebeurt door een werkgroep waarin de grote cloudleveranciers zijn vertegenwoordigd (o.a. Amazon, Microsoft, Google, Alibaba). Bij oplevering van de eerste versie is o.a. gekeken naar de berichtspecificaties zoals die binnen verschillende cloudplatforms in gebruik waren.

Hoewel de standaard nog jong is wordt hij toch al vrij breed ondersteund. Hij is bruikbaar binnen een aantal cloudplatforms en er zijn al een aantal SDK's ontwikkeld voor gebruik met de meest gebruikte programmeertalen. Er is/wordt ook voor een tiental veelgebruikte protocollen beschreven hoe de standaard daar gebruikt moet worden (o.a. HTTP, Webhook, AMQP, Kafka). De manier waarop gebruik

wordt gemaakt van het HTTP protocol en Webhooks is eventueel bruikbaar als input voor het REST-full beproeven van de functionele notificatiestandaard.

De CloudEvents standaard kent de mogelijkheid om door middel van 'extensions' extra attributen te definiëren. Wanneer de CloudEvents standaard als basis wordt gekozen zijn dus specifiek voor gebruik binnen de Nederlandse overheid aanvullende attributen te definiëren. Of en hoe dit gebeurt wordt binnen project Notificatieservices uitgewerkt in een aparte werkgroep die de opdracht heeft om de beoogde berichtstandaard voor notificeren te ontwikkelen. De resultaten daarvan worden gepubliceerd in een apart document 'Notificeren standaard berichtformaat' (de 0.1 versie hiervan wordt in juni 2021 gepubliceerd).

6.3 WebSub interactiestandaard

WebSub is een op HTTP gebaseerde standaard voor op het PubSub patroon gebaseerde communicatie tussen gedistribueerde aanbieders, makelaars en afnemers en sinds 2018 een W3C aanbeveling.

> *WebSub provides a common mechanism for communication between publishers of any kind of Web content and their subscribers, based on HTTP web hooks. Subscription requests are relayed through hubs, which validate and verify the request. Hubs then distribute new and updated content to subscribers when it becomes available. – bron: WebSub W3C Recommendation*

WebSub standaardiseert op het gebruik van HTTP als protocol met een aantal afspraken over hoe de interactie tussen aanbieders, makelaar en afnemers plaatsvindt. De 'happy flow' ziet er globaal als volgt uit:

- Afnemers kunnen bij een aanbieder zien dat er een of meer makelaar(s) zijn die notificaties kan/kunnen verstrekken als de aanbieder nieuws heeft over een bepaald onderwerp (via een '<link rel="hub" href="https://hub.example.com/"'>' in headerinfo van een webpagina).
- Om zich te abonneren (of een abonnement op te zeggen) op notificaties naar aanleiding van bepaalde gebeurtenissen doen afnemers een HTTP POST request naar een genoemde makelaar:
  - De afnemer verstrekt daarbij het onderwerp (via een URL) waarop hij zich wil abonneren, het internet(callback)adres van de service waarmee hij nieuwe notificaties wil ontvangen ('webhook') en eventueel een geheime code (en evt. een periode waarvoor het abonnement moet gelden);
  - De makelaar stuurt via een HTTP GET request een verificatiebericht naar het opgegeven (callback)adres met o.a. de opgegeven onderwerp-URL en een random tekenreeks;
  - De afnemer stuurt een positief antwoord met een HTTP-2xx code met in de body van het bericht de ontvangen random tekenreeks;
  - De makelaar accepteert het abonnementsverzoek.
- Aanbieders stellen hun makelaar(s) op een overeengekomen manier op de hoogte wanneer ze nieuws hebben. Hoe ze dit doen is geen onderdeel van de standaard. (Sommige hubs vragen bijv. om een HTTP POST request met als parameters hub.mode="publish" and hub.url=(URL van een nieuwe resource).
- De hub stuurt notificaties naar alle geabonneerde afnemers via een POST request naar de door een afnemer opgegeven callback-URL; als een afnemer bij abonnementsaanvraag een geheime code heeft opgegeven zal de makelaar notificaties daarmee digitaal ondertekenen.

- De afnemer bevestigd ontvangst van een notificatie via een HTTP-2xx om te laten weten dat de notificatie succesvol is ontvangen.

Gebruik van de WebSub standaard zorgt ervoor dat de interactie tussen actoren op een aantal aspecten gestandaardiseerd verloopt (bijv. protocol, wijze van nemen en opzeggen van abonnementen en borgen van bepaalde mate van betrouwbaarheid) waardoor er minder of zelfs geen ‘maatwerkafspraken’ nodig zijn.

De standaard heeft een lichtgewicht karakter en is met name ontwikkeld voor omgevingen waarin geen hoge eisen worden gesteld aan betrouwbaarheid (er is bijv. niet gespecificeerd hoe moet worden omgegaan met fouten en storingen). Van oorsprong is de standaard ontwikkeld voor het pushen van berichten door aanbieders van nieuwssites en blogs (bijv. CNN en Blogger) naar afnemers daarvan afnemers zoals Feedly, Flipboard (vaak via ‘feedreaders’ zoals Feedly of Flipboard). Er zijn enkele makelaars (bijv. pubsubhubbub.appspot.com en pubsubhubbub.superfeedr.com) maar de status daarvan is niet altijd even duidelijk.

Gelet op de context van project Notificatieservices lijkt de standaard beperkt bruikbaar. Maar binnen bepaalde omstandigheden kan gebruik van delen van de standaard nuttig zijn om te voorkomen dat er zelf vergelijkbare functionaliteit wordt ontworpen (bijv. het mechanisme om geautomatiseerd een abonnement te kunnen afsluiten).

PubSub: gebruik de WebSub standaard om onderdelen toe te passen waar dit kan

6.4 AsyncAPI beschrijvingsstandaard

AsyncAPI is een standaard om effectief met gebeurtenisgedreven API's te kunnen werken[12]. Vergelijkbaar als de OpenAPI standaard voor REST API's maakt de AsyncAPI standaard het onder andere mogelijk om betere documentatie te maken, geautomatiseerd broncode te genereren, gegevensvalidatie te doen en in zijn algemeenheid systeemontwikkeling te vereenvoudigen en versnellen. In maart 2021 is het initiatief onderdeel geworden van de Linux Foundation.

Gelet op de veelheid aan betrokken applicaties en technologische diversiteit bij gebeurtenisgedreven werken is standaardisatie van interfacing cruciaal. De relatief jonge AsyncAPI standaard kan daarbij een belangrijke rol spelen omdat hij generiek van aard en breed toepasbaar is, breed wordt ondersteund en het gebruik ervan wereldwijd snel toeneemt[13].

[12]De naam ‘AsyncAPI’ lijkt aan te geven dat het alleen gaat om asynchrone berichtuitwisseling. Belangrijkste doel van de standaard is echter om API’s geschikt te maken om gebeurtenisgedreven te werken wat veelal synchroon verloopt maar niet daartoe beperkt hoeft te zijn. Daarom gebruiken we, net als binnen het initiatief zelf gebeurt, de term ‘gebeurtenisgedreven API’.

[13] Het is de snelst groeiende API-specificatie volgens een recente enquête onder ontwikkelaars, met een verdrievoudiging van het productiegebruik van 2019 tot 2020.

De standaard is bedoeld voor het gebeurtenisgedreven gegevens uitwisselen tussen applicaties, meestal met gebruik van het PubSub patroon. Gelet op het doel van project Notificatieservices om gebeurtenisgedreven werken en notificeren binnen de overheid te standaardiseren kan de AsyncAPI standaard hierbij behulpzaam zijn[14].

De standaard specificeert onder andere de beschrijving van berichtinhoud (opbouw en attributen), locatie van applicaties, te gebruiken protocollen (bijv. AMQP, HTTP, MQTT, Kafka, WebSockets) en authenticatie- en autorisatiemechanismen (bijv. gebruikersnaam/wachtwoord, certificaten, API-keys, OAuth2).

De standaard is zowel bruikbaar in situaties waarin 1 applicatie zowel de rol van aanbieder als makelaar invult (‘client-server model’) als in situaties waarin dit gebeurt door verschillende applicaties (‘broker-centric’).

Het feit dat de standaard laagdrempelig is toe te passen, breed wordt ondersteund, steeds meer ondersteunende software als open source beschikbaar is en geautomatiseerd genereren van documentatie en code mogelijk maakt zijn redenen om van de standaard gebruik te maken.

Gebruik de AsyncAPI standaard bij het beschrijven van gebeurtenisgedreven API’s

De huidige 2.0.0 versie van de AsyncAPI specificatie is te vinden op:
https://www.asyncapi.com/docs/specifications/2.0.0

[14]De scope van de AsyncAPI standaard is breder dan voor binnen project Notiicatieservices is gedefinieerd. AsyncAPI spreekt over ‘berichten’ waarbij de inhoud zowel betrekking kan hebben op een plaatsgevonden ‘gebeurtenis’ (‘event’) als op een ‘opdracht’ (‘command’). Berichten die uit te voeren opdrachten bevatten vallen nu buiten de scope van project Notifcatieservices (zie ‘Scope’ binnen het hoofddocument).

7 Implementatiepatronen

7.1 Inleiding

Dit hoofdstuk beschrijft een aantal implementatiepatronen die bruikbaar zijn voor het ontwerpen van gebeurtenisgedreven oplossingen. Notificeren van afnemers is daar een onderdeel van en wordt veelal ondersteund door een specifieke implementatie van het PubSub patroon. De in dit hoofdstuk beschreven patronen hebben een verschillend karakter maar voor allemaal geldt dat ze bruikbaar zijn om te komen tot goede oplossingen om gebeurtenisgedreven te kunnen werken.

Voor de patronen in de eerste helft van dit hoofdstuk (‘Change Data Capture’, ‘Basale messaging’, ‘Event Notification’, ‘Event Carried State Transfer’) geldt dat ze een duidelijke relatie hebben met notificeren. Toepassing van die patronen is ook mogelijk door gebruik te maken van bekende stijlen (bijv. service-oriëntatie en REST-stijl) en voorzieningen (bijv. message brokers, message queues en relationele databases) zoals die bij opvraging van gegevens gebruikelijk zijn.

Met name voor de patronen in de tweede helft van het hoofdstuk (‘event streaming’, ‘event processing’, ‘event sourcing’ en ‘CQRS’) geldt dat ze meer doelen hebben alleen goed kunnen notificeren van afnemers. Gebruik van die patronen betekent vaak een fundamentele verandering (‘paradigmashift’) ten opzichte van hoe oplossingen nu worden ontworpen, gerealiseerd en gebruikt. Een fundamenteel verschil is bijv. dat het gebruikelijke vastleggen van objecten met attributen vervangen wordt door het gaan vastleggen van gebeurtenissen en dat te zien als bronregistratie. De mate waarin dit type patronen geschikt zijn voor gebruik wordt sterk bepaald door de aanwezige ambities, behoeften, kennis en mogelijkheden van betrokken organisaties en applicaties. Organisaties met nog weinig ervaring in gebeurtenisgedreven werken kunnen ze nu zien als patronen die mogelijk op langere termijn toepasbaar zijn en er stapsgewijs meer gebruik van gaan maken.

Ieder patroon wordt afzonderlijk beschreven. In de praktijk kunnen bepaalde patronen ook worden gecombineerd (wat kan leiden tot een nieuw samengesteld patroon (‘compound pattern’)). Hoewel implementatiepatronen al meer richting geven aan hoe een oplossing er uit moet komen te zien geldt ook daarvoor dat ze geen kookboekrecept zijn. Er zal nog steeds binnen de eigen specifieke context moeten worden bezien of en op welke manier een of meerdere patronen goed zijn te gebruiken.

7.2 Change Data Capture

Doel	Applicaties en registraties die data georiënteerd zijn gebeurtenissen te laten publiceren.
Probleem	De meeste in gebruik zijnde applicaties en registraties binnen de overheid zijn gericht op het bijhouden van gegevens zodat de actuele status van objecten bekend is. Gegevens worden daarbij overschreven, de context van de plaatsgevonden gebeurtenis gaat verloren.
Oplossing	Leidt op basis van plaatsgevonden gegevenswijzigingen af welke gebeurtenissen hebben plaatsgevonden.
Gevolgen	Systeemgebeurtenissen zijn betrouwbaar te publiceren maar om bedrijfsgebeurtenissen te kunnen publiceren moet een vertaalslag plaatsvinden. Afhankelijk van de context is dat

wel/niet met een bepaalde betrouwbaarheid mogelijk.

Voorbeeld

Aanmaak van een nieuwe gegevensobject, bijvoorbeeld in de vorm van een rij in een tabel in een database, is aanleiding om gegevens te publiceren over een plaatsgevonden gebeurtenis ‘ObjectAangemaakt’.

Het vullen van een attribuut ‘EindDatum’ bij een object is aanleiding om gegevens te publiceren over een plaatsgevonden gebeurtenis ‘ObjectVerwijderd’.

Werking

Implementatie

Implementatie is op verschillende manieren mogelijk. Enkele mechanismen die bruikbaar zijn om af te leiden dat er informatie in de vorm van een gebeurtenis moet worden gepubliceerd:

- Gebruik van triggers op databasetabellen die reageren op het plaatsvinden van bepaalde typen gegevensbewerkingen (bijv. toevoegen, wijzigen of verwijderen).
- Periodiek scannen van data waarbij op basis van bepaalde statusgegevens (bijv. versienummers of timestamps) is af te leiden dat er een gebeurtenis heeft plaatsgevonden.
- Gebruik van softwarecomponenten met bedrijfslogica. Hiermee kan op maat worden gedefinieerd wanneer er sprake is van bepaalde type gebeurtenissen.

Opmerkingen

- Het Change Data Capture patroon zal altijd bruikbaar blijven om in specifieke behoeften te voorzien. Bepaalde wijzigingen kunnen bijv. als ‘gebeurtenissenstroom’ worden aangeboden aan voorzieningen die ze bijv. (al dan niet aangepast) door te leveren aan afnemers of om op basis ervan trends te ontdekken. Gelet op het feit dat veel in gebruik zijnde applicaties en registraties binnen de overheid nog volledig data georiënteerd zijn ingericht, en dit zeker nog een aantal jaren het geval zal zijn, zal het patroon de komende jaren vaak nodig zijn om applicaties en registraties (meer) geschikt te maken voor gebeurtenisgedreven werken.
- Het periodiek leveren van ‘was/wordt mutaties’ is te zien als een specifieke vorm van het Change Data Capture patroon. Meestal betreft het informatierijke notificaties met zowel de status van objecten vóór als de status van een object ná plaatsgevonden mutaties[15]. Het doel is vaak om objectgegevens binnen de eigen omgeving binnen een acceptabele termijn te kunnen actualiseren (bijv. als externe

[15]De mate waarin berichten informatierijk zijn kan verschillen. In minimale vorm bevat een bericht alleen was-wordt gegevens van gewijzigde objectgegevens maar een bericht kan ook alle gegevens van het gewijzigde object bevatten (of zelfs meer als een object wordt gezien als deel van een meer omvattend object).

gegevens zijn gerepliceerd) en niet om actie te ondernemen direct na het optreden van gebeurtenissen. Voor dat doel moeten direct na het optreden van gebeurtenissen notificaties worden gestuurd met informatie over de gewijzigde status van objecten ('Event-Carried State Transfer').

7.3 Basale messaging

Doel Gegevens flexibel, betrouwbaar en asynchroon kunnen uitwisselen tussen applicaties.

Probleem Uitwisselen van gegevens tussen applicaties uitwisselen is lastig als ze in veel opzichten van elkaar verschillen (bijv. technologie, locatie, beschikbaarheid).

Oplossing Verpak gegevens in zelfstandig te verwerken berichten die zijn te bewaren en op verschillende momenten zijn te verwerken.

Gevolgen Betrokken applicaties zijn van elkaar ontkoppeld en kunnen los van elkaar functioneren.

Voorbeeld Een bestelapplicatie stuurt na het plaatsen van een bestelling een bericht naar een interne logistieke applicatie en naar een financiële applicatie in een SaaS omgeving waarmee de financiële afhandeling wordt verzorgd.

Werking

Implementatie

- Ontkoppel verzender en ontvanger door gebruik te maken van een messagebroker applicatie die als makelaar bemiddelt tussen beide partijen en bijv. Tijdelijk berichten kan bufferen als een ontvanger tijdelijk niet bereikbaar is.

Opmerkingen

- De reden om te spreken over 'basale messaging patronen' is dat de 'Message driven' architectuurstijl op allerlei manieren toepasbaar is. Met de hier bedoelde patronen bedoelen we de basispatronen zoals die veelvuldig in de praktijk worden toegepast.
- Er bestaan veel patronen om aan contextspecifieke eisen bij messaging te kunnen voldoen. Voor een uitgebreid overzicht daarvan verwijzen we naar https://www.enterpriseintegrationpatterns.com/patterns/messaging/ .
- Er zijn veel verschillende, vaak al tientallen jaren bestaande, applicaties die de rol van 'message broker' kunnen vervullen. De afgelopen jaren wordt steeds meer gebruik gemaakt van extern gehoste (bijv. in een cloudomgeving) brokerapplicaties[16].
- Messaging is een algemeen bruikbaar patroon dat geschikt is voor allerlei vormen van gegevensuitwisseling tussen applicaties. Specifiek voor notificeren geldt dat aanbieders berichten publiceren met gegevens over plaatsgevonden

[16] 'Message broker' kan in deze context ook betekenen dat er sprake is van een combinatie van componenten (bijv. een apart message queue applicatie) die samen de benodigde functionaliteit realiseren (ook vaak aangeduid als 'de enterprise servicebus'). Het gaat hier om de kernfunctionaliteit om berichten te kunnen ontvangen en bezorgen.

gebeurtenissen die aan afnemers worden verstrekt.

- Messaging heeft als doel om een bericht van A naar B te kunnen brengen. Berichten worden hooguit tijdelijk bewaard voor systeemtechnische doelen (bijv. gegarandeerde aflevering) en berichten worden geïsoleerd van elkaar verwerkt[17].
- Gelet op hoe veel applicaties en registraties binnen de overheid zijn ontworpen en functioneren zal messaging in zijn basale vorm waarschijnlijk het meest gebruikte patroon blijven om te notificeren. Daarnaast zullen 'zelfs' minder wenselijke stijlen als bestandsuitwisseling in gebruik blijven. Reden waarom we binnen de context van project Notificatieservices ook dat type stijlen en patronen zien als laagdrempelige manieren om meer gebeurtenisgedreven te gaan werken.

7.4 Event notification

In het hoofdstuk 'Aspecten' is ingegaan op informatiearm en informatierijk notificeren. Bij informatiearm notificeren is het gebruikte patroon 'Event Notification'.

Doel Verstrekken van minimale gegevens over plaatsgevonden gebeurtenissen in notificaties

Probleem In bepaalde situaties is het onwenselijk of niet toegestaan om alle relevante gegevens over een plaatsgevonden gebeurtenis te verstrekken.

Oplossing Neem in notificaties alleen de voor afnemers minimaal benodigde gebeurtenisgegevens op en neem een link op voor het eventueel opvragen van aanvullende gegevens.

Gevolgen

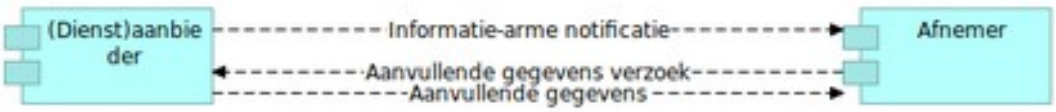
- Afnemers ontvangen minimale gegevens op basis waarvan zij moeten beslissen of zij of aanvullende gegevens moeten worden opgehaald
- Aanbieders moeten, bij voorkeur permanent, een service aanbieden waarmee afnemers aanvullende gegevens kunnen ophalen. Voor gebruik van de service moet bij de aard van de op te vragen gegevens passende authenticatie, autorisatie en beveiliging zijn geregeld.
- Aanbieder moeten, in ieder geval bij verstrekking van vertrouwelijke gegevens, zowel loggen aan welke afnemers notificaties zijn verstrekt als welke afnemers welke aanvullende gegevens hebben opgevraagd.

Voorbeeld

- Een afnemer ontvangt een informatiearme notificatie dat een nieuwe productaanvraag is ingediend en gaat met behulp van een opgenomen link de betreffende aanvraaggegevens ophalen.
- Een afnemer ontvangt een notificatie dat een bepaald type gebeurtenis heeft plaatsgevonden waarbij een object betrokken was waarmee de afnemer een relatie heeft. Naar aanleiding daarvan vraagt hij aanvullende gegevens op.

[17]Met 'geisoleerd van elkaar vewerkt' bedoelen we dat berichten niet worden beschouwd als onderdeel van een betekenisvolle stroom van berichten zoals wel het geval is bij patronen zoals 'Even streaming' en 'Event sourcing'.

Werking



Implementatie

Opmerkingen

7.5 Event Carried State Transfer

In het hoofdstuk ‘Aspecten’ is ingegaan op informatiearm en informatierijk notificeren. Bij informatierijk notificeren is het gebruikte patroon ‘Event Carried State Transfer’.

Doel Verstrekken van alle voor afnemers relevante gegevens over plaatsgevonden gebeurtenissen in notificaties

Probleem In bepaalde situaties volstaat het niet of is het onwenselijk om afnemers alleen te informeren dat er een bepaalde gebeurtenis heeft plaatsgevonden.

Oplossing Neem in notificaties alle voor afnemers relevante gebeurtenisgegevens op.

Gevolgen

- Afnemers ontvangen alle gegevens over een gebeurtenis en de gevolgen daarvan en kunnen direct na ontvangst tot verwerking overgaan.
- Aanbieders kunnen volstaan met het publiceren van informatierijke notificaties.
- Aanbieder moeten zorgen dat bekend is, in ieder geval bij verstrekking van vertrouwelijke gegevens, aan welke afnemers notificaties zijn verstrekt.

Voorbeeld

- Een afnemer ontvangt een informatierijke notificatie van een nieuwe productaanvraag en kan meteen tot verwerking daarvan overgaan.
- Een afnemer ontvangt een notificatie dat een bepaald type gebeurtenis heeft plaatsgevonden inclusief de actuele gegevens van een betrokken object waarmee de afnemer een relatie heeft en kan meteen tot verwerking overgaan.

Werking

Implementatie

Opmerkingen

7.6 Event Streaming

Doel Berichten met gebeurtenisinformatie effectief kunnen gebruiken voor verschillende doeleinden.

Probleem De intrinsieke waarde die gebeurtenisberichten heeft gaat verloren als ze na verwerking niet worden bewaard.

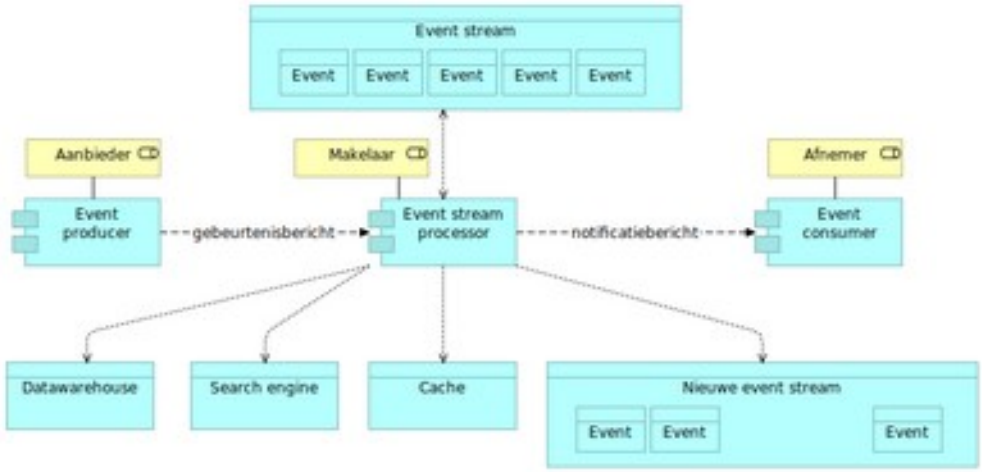
Oplossing Beschouw elkaar opvolgende berichten met informatie over gebeurtenissen als waardevol en bewaar ze op een zodanig manier dat ze bruikbaar blijven voor meerdere doeleinden[18].

Gevolgen Gebeurtenisberichten moeten met dezelfde zorgvuldigheid worden behandeld en bewaard als gebruikelijk is bij objectgegevens in bronregisters.

Voorbeeld

- Door realtime berichten over plaatsvindende gebeurtenissen in samenhang te bezien kunnen patronen en trends worden ontdekt waarop passend kan worden gereageerd (bijv. voor constateren van beveiligingsincidenten of fraudedetectie).
- Door een aantal gebeurtenisberichten opnieuw ‘af te spelen’ is te zien wat zich eerder heeft afgespeeld op basis waarvan conclusies zijn te trekken (bijv. voor auditing of om de oorzaak van opgetreden fouten te vinden).
- Door realtime berichten over het surfgedrag van een websitebezoeker te monitoren kan proactief worden gehandeld om de klantreis te optimaliseren (bijv. door hulp aan te bieden of op gerelateerde diensten te wijzen).

Werking



Het PubSub patroon voor notificeren is te implementeren via het event streaming patroon. Maar het levert daarnaast veel andere mogelijkheden op om informatie over gebeurtenissen te gebruiken. Naast de voorbeelden van ‘datawarehouse’, ‘search engine’ en ‘cache’ is hierboven ook afgebeeld dat er ook allerlei soorten nieuwe streams zijn te maken voor verschillende doelen en evt. afnemers waardoor er een netwerk aan streams kan ontstaan.

Het diagram maakt gebruik van de bij event streaming gebruikelijke terminologie (‘producer’, ‘processor’, ‘consumer’) die in relatie tot notificeren de rollen van ‘aanbieder’, ‘makelaar’ en ‘afnemer’ invullen.

Implementatie Bij toepassing van het event streaming patroon gaat het vaak om grote hoeveelheden data die snel verwerkt moet kunnen worden en voor langere tijd bewaard moeten kunnen worden (vaak wordt ook nog functionaliteit voor bewerkingen of analyse verwacht). Gebruik van het event streaming patroon vereist gebruik van gespecialiseerde applicaties

[18]De periode dat berichten worden bewaard kan verschillen. Incidenteel kan bewaring ook permanent plaatsvinden. Daarvoor is echter met name het Event Sourcing patroon bedoeld.

of ‘platforms’ met Apache Kafka als bekendste voorbeeld. Kafka kent een ‘Stream API’ waarmee aanbieders en afnemers flexibel en betrouwbaar gebruik kunnen maken van streams (o.a. via een een eigen taal voor gebruik van bijv. filters, maps, groepering en windowing.

Opmerkingen

- Gebeurtenisgegevens kunnen wel of niet gegevens bevatten die de context van gebeurtenisgerelateerde gegevens beschrijven (‘metadata’). Is dit niet het geval dan spreken we over ‘data’ en ‘data streams’ (bijv. bij sensoren die alleen een pakket ruwe data leveren). Zijn er wel contextgegevens aanwezig dan spreken we over ‘events’ en ‘event streams’ (bijv. bij gebeurtenisgegevens die moeten worden geïnterpreteerd om goed verwerkt te kunnen worden)[19].
    - Beide hebben waarde maar de aanwezigheid van (goede) metagegevens verhoogt de bruikbaarheid van berichten aanzienlijk. Binnen de CloudEvents standaard geldt bijv. dat sommige metadata verplicht aanwezig moeten zijn binnen berichten.
    - Het is wenselijk om metadata en inhoudelijke berichtdata herkenbaar te onderscheiden. Metadata worden vaak aangeduid als apart doorgegeven ‘header data’ en inhoudelijke gegevens als ‘payload data’ (bijv. via aparte secties binnen een bericht).
- Het event streaming patroon is hier opgenomen omdat het te zien is als een implementatie van het PubSub patroon. Gelet op de eisen die gebruik van het patroon stelt ligt gebruik van het patroon niet voor de hand als er uitsluitend eenvoudige notificatieberichten nodig zijn. Het patroon zal met name aantrekkelijk worden om toe te passen bij een hoger volwassenheidsniveau van gebeurtenisgedreven werken waarbij gebruik van daarbij passende technologie ingeburgerd is.

Bij de beschrijving van het event streaming patroon is opgemerkt dat er naast het bewaren en beschikbaar stellen van gebeurtenissen ook in meer of mindere mate sprake kan zijn van *verwerking* van gebeurtenisgegevens wat wordt aangeduid met de term: ‘event processing’. We beschouwen

7.7 Event processing

Doel Het verwerken van gebeurtenisgegevens tot gegevens die voor een bepaald doel of afnemer bruikbaar zijn.

Probleem Gebeurtenisgegevens zoals ze aangeleverd worden door een aanbieder zijn niet altijd voor alle doelen of afnemers geschikt.

Oplossing Verwerk gebeurtenisgegevens zodanig dat inhoud en vorm geschikt zijn voor gebruik.

Gevolgen Gegevens over gebeurtenissen worden op verschillende manieren verwerkt en leiden tot afgeleide gegevens. Bijv. door gegevens te transformeren of te combineren.

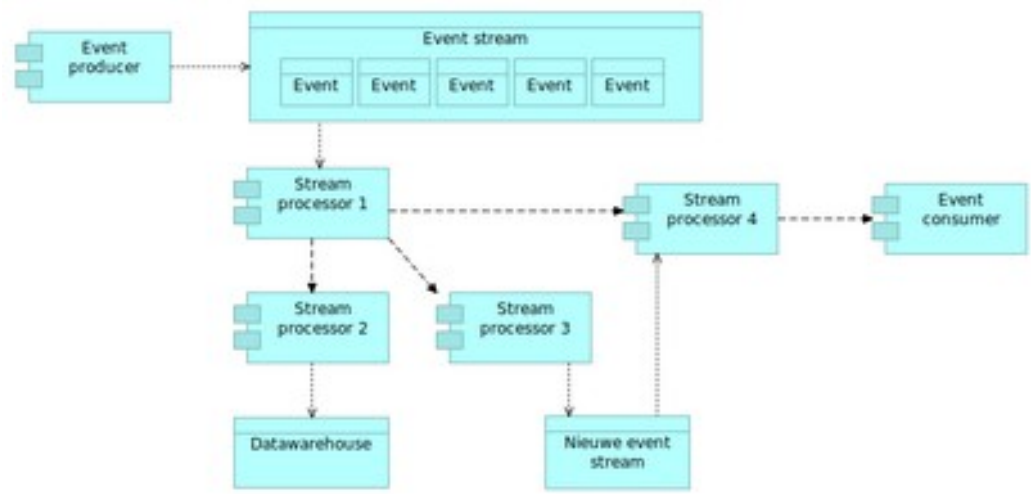
Voorbeeld Het filteren van beschikbare gebeurtenisgegevens voor verstrekking aan afnemers is een

[19] In de praktijk wordt dit onderscheid niet altijd gemaakt en wordt ook bij ruwe datastromen gesproken over ‘eventstreams’.

veelvoorkomende eenvoudige vorm van event processing.

Beschikbare gebeurtenisgegevens worden gefilterd en omgevormd tot gegevens die binnen een specifieke omgeving bruikbaar zijn (bijv. een datawarehouse of een cache waarin het laatst ontvangen bericht over een bepaald object staat).

Werking



Bij de beschrijving van event streaming is al een voorbeeld getoond van hoe ‘event processing’ plaats kan vinden door een streaming platform als Kafka. In bovenstaand diagram is weergegeven hoe event processing in plaats van met 1 applicatie/platform ook steeds mee gebeurd door gebruik te maken van aparte componenten die (vaak via events…) met elkaar communiceren. Een voorbeeld daarvan is het gebruik van serverless functies binnen cloudomgevingen waarbij een functie bepaalde afgebakende verwerkingsfunctionaliteit levert.

Implementatie

Opmerkingen

-

7.8 Event sourcing

Doel Duurzaam kunnen gebruiken van gebeurtenisinformatie

Probleem Als gebeurtenissen alleen worden gebruikt om objecten in een bronregister te creëren, wijzigen of verwijderen en niet worden bewaard gaat veel informatie verloren en is gebeurtenisgedreven werken niet optimaal te ondersteunen.

Oplossing Bewaar gebeurtenissen onveranderlijk en duurzaam in een bronregister (‘event store’) en beschouw ze als bron van waarheid (‘systems of record’)

Gevolgen Een bronregister bestaat niet meer uit gerelateerde objecten met attribuutwaarden maar uit bewaarde gebeurtenissen in een eventstore (‘eventlog’).

Om een eventstore te actualiseren moeten nieuwe gebeurtenissen er (‘log based’) aan

toe worden gevoegd.

Om de actuele waarden van attributen van objecten te bepalen moet gebruik worden gemaakt van de eventstore (of daarvan afgeleide views; zie later).

Alle gebeurtenissen worden bewaard en maken het daarmee mogelijk om:

- vanuit informatiearme notificaties via links te verwijzen naar het, eventueel al langer geleden plaatsgevonden, event dat de aanleiding voor notificeren was.
- historische bevragingen te doen (bijv. vanaf het begin, in een periode, op op een tijdstip) voor bijv. audit-doeleinden.
- verschillende toestanden, waaronder de meest actuele toestand, te reconstrueren (bijv. om een nieuwe traditionele Read-only registratie op te bouwen of om nieuwe maatwerk views met bijv. alleen bepaalde soorten gebeurtenissen op te bouwen).

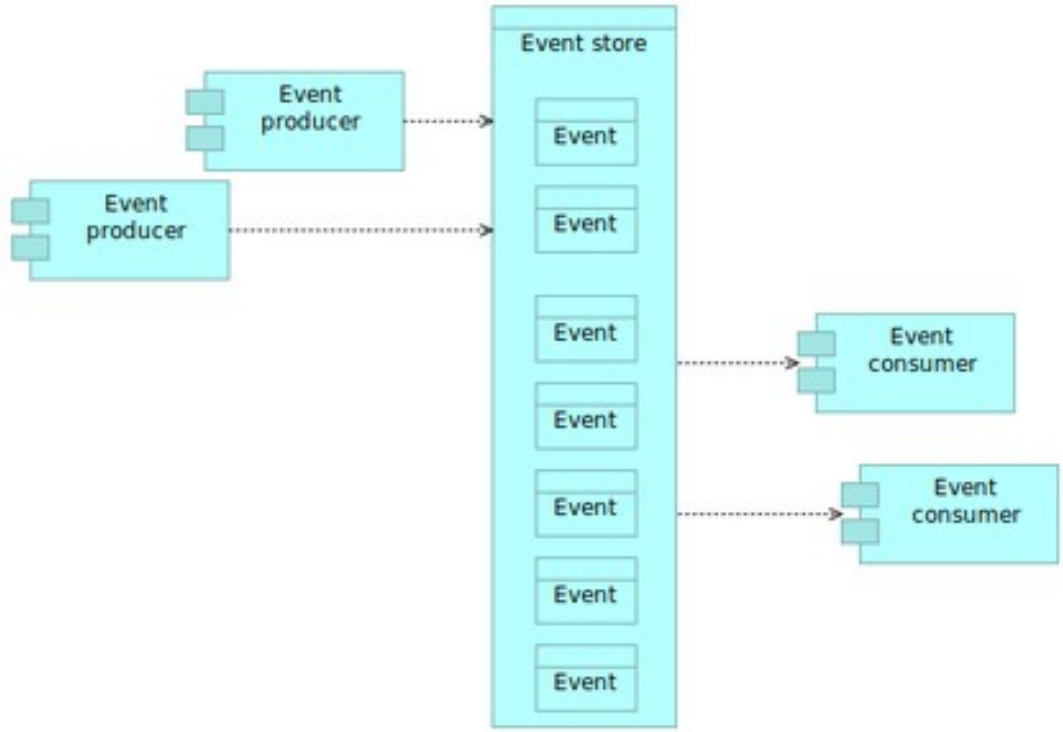
Voorbeeld

Als een klant op een website producten toevoegt en verwijdert aan zijn winkelmandje wordt voor iedere actie een ‘event’ in de eventstore opgeslagen. Bij het afrekenen is aan de hand van de opgeslagen events te bepalen wat hij moet afrekenen. Het afrekenen wordt daarna ook als event in de eventstore opgeslagen waarmee de transactie is afgerond en als event van het type ‘AankoopVoltooid’ wordt opgeslagen.

De afdeling logistiek heeft zich geabonneerd op events van het type ‘AankoopVoltooid’ en ontvangt daarover notificaties. In de eventstore is op te zoeken welke artikelen zijn aangekocht en verzonden moeten worden. De verzending van de artikelen wordt als event in de eventstore vastgelegd (op basis waarvan een klant bijv. online kan opvragen wat de actuele status van zijn aankoop is).

De afdeling Marketing kan op basis van de vastgelegde events bijv. analyses doen van het koopgedrag van klanten (bijv. hoe vaak en welke artikelen uit het winkelmandje worden verwijderd) en van de tijd tussen aankoop en verzending.

Werking



Implementatie

- Een eventstore is te implementeren met behulp van traditionele databases (bijv. een relationele- of nosql-database). Het betekent echter een heel ander soort gebruik dan waarvoor dit type databases zijn ontworpen[20]. Er zijn ook relatief jonge databases die speciaal zijn ontworpen om als eventstore te worden gebruikt (bijv. EventStoreDB).
- Vanuit het streven naar ontkoppeling is het onwenselijk als de eigenaar van de eventstore (die bij notificeren de rol van aanbieder heeft) en afnemers gebruik maken van dezelfde event store omdat eisen en wensen voorspelbaar zullen (gaan) verschillen (bijv. omdat het gewenste niveau van detail van vastlegging verschilt). Een event store is primair bedoeld om een goede bronregistratie (van gebeurtenissen) te voeren. Gebruik er van door afnemers moet mogelijk worden gemaakt via aanvullende mechanismen die zorgen voor ontkoppeling tussen store-eigenaar en afnemers. Om deze reden, en om gebruik van een eventstore te vergemakkelijken, worden vaak bepaalde patronen toegepast (zie verder)[21].

Opmerkingen

- Het is belangrijk om onderscheid te maken in ‘business events’ (herkenbaar voor domeinexperts) en ‘system events’ (herkenbaar voor IT’ers). Afnemers zullen voornamelijk behoefte hebben aan voor hen herkenbare business events. Los daarvan geldt dat vanuit het streven naar ontkoppeling het
- Afhankelijk van de context is event sourcing relatief eenvoudig of zeer complex toe te passen. Event sourcing is met name geschikt voor stabiele omgevingen

[20] In een tradtionele database worden wijzigingen in een logfile verzameld maar vormen de databaseobjecten het bronregister. In een eventstore is het precies andersom. Reden waarom het ook wel wordt aangeduid als: ‘Turning the database inside out’.

[21] Voor het bepalen van grenzen, niveau van detail, wijze van vastlegging en verstrekking naar afnemers uit andere domeinen wordt vaak gebruik gemaakt van inzichten uit de Domain Driven Design architectuurstijl.

waarin vastgelegde events gegarandeerd onveranderlijk (‘immutable’) zijn en waarin business logica niet al te vaak wijzigt. Is dat wel het geval dan kan gebruik van event sourcing al snel (te) complex worden.

- Werken met een eventstore in plaats van een (sql)database betekent een grote omschakeling voor zowel de organisatie die verantwoordelijk is voor het bronregister als voor de organisaties die er gebruik van maken. Op dit moment is de kennis en ervaring met event sourcing binnen de overheid nog zeer beperkt.

7.9 Command/Query Responsibility Segregation

Doel Het door verschillende voorzieningen afhandelen van schrijven (creëren/wijzigen/verwijderen) en alleen lezen van (gebeurtenis)gegevens.

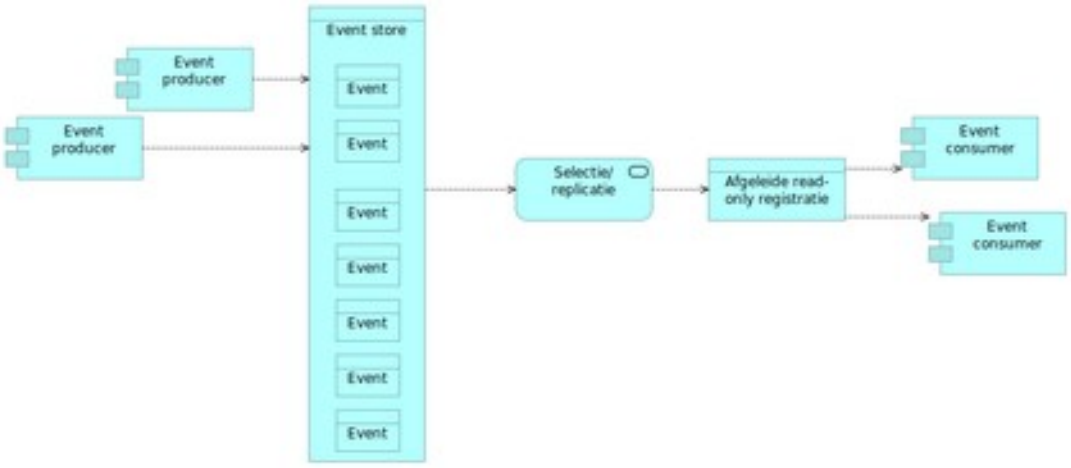
Probleem Het combineren van schrijven en lezen van gegevens leidt tot suboptimale oplossingen.

Oplossing Gebruik aparte voorzieningen die verantwoordelijk zijn voor het schrijven en voor het alleen lezen van gegevens.

Gevolgen Zowel de voorzieningen voor schrijven als voor lezen kunnen zich richten op hun kernfunctie en los van elkaar worden geoptimaliseerd. Voor schrijven worden gegevens bijvoorbeeld als gebeurtenis of in genormaliseerde objecten vastgelegd terwijl voor lezen eenvoudig toegankelijke objecten met gecombineerde gegevens worden aangeboden. Het kan zowel gaan om gegevens in een traditionele database als om gebeurtenissen die in een event store zijn opgeslagen. Het patroon wordt met name voor de laatste categorie toegepast.

Voorbeeld Applicatie A die een gebeurtenis wil vastleggen via event sourcing gebruikt daarvoor een service die gebeurtenissen met een hoog niveau van detail vastlegt in de event store. De event gegevens, of een deel daarvan, worden gerepliceerd naar een aparte registratie in een vorm die eenvoudig, snel en op maat raadplegen mogelijk maakt. Afnemers die informatie over gebeurtenissen willen opvragen, bijv. na notificatie, een service aanroepen die snel gegevens uit de ‘read only’ registratie ophaalt.

Werking



Implementatie Voor implementatie is gebruik te maken van verschillende mechanismen (bijv. selectie, filtering, replicatie). Afhankelijk van de mogelijkheden en wensen van afnemers kunnen verschillende keuzes, met verschillende 'read models', worden gemaakt. Gebeurtenissen kunnen bijvoorbeeld worden 'vertaald' naar herkenbare objecten met attributen binnen een relationele of nosql database als afnemers dit wensen. Uitgaande van 1 bronregister met gebeurtenissen kunnen zo meerdere op maat gesneden raadpleegvoorzieningen worden gerealiseerd. Aangeboden voorzieningen kunnen zowel de actuele als historische status van objecten tonen. Gelet op het feit dat veel registraties binnen de overheid slechts beperkt of helemaal niet historische bevraging ondersteunen zal in veel gevallen het kunnen raadplegen van de actuele status van objecten volstaan.

Opmerkingen

- Bij het event sourcing patroon is al opgemerkt dat het belangrijk is om te zorgen voor voldoende ontkoppeling tussen event store eigenaar en afnemers. Het CQRS patroon is hiervoor een zeer bruikbaar en vaak toegepast patroon. Ook daarvoor geldt dat het grote veranderingen betekent in ontwerp en realisatie van oplossingen.

8 COMMUNICATIEPROTOCOLLEN

8.1 INLEIDING

Om de uitwisseling van informatie via een netwerk of communicatiesysteem goed te laten verlopen zijn verschillende protocollen, sets van regels, ontwikkeld. Bij het inrichten van notificatieprocessen kan, ook nadat is gekozen voor een bepaald patroon, gebruik worden gemaakt van verschillende protocollen. In aanvulling op een berichttekst kan bijv. opslag-, routerings- en bezorgingsinformatie wordt toegevoegd vóór verzending en verwijderd van het bericht voordat het wordt afgeleverd bij de ontvangende applicatie.

Dit hoofdstuk beschrijft een aantal protocollen in relatie tot de bruikbaarheid er van bij notificeren.

8.2 COMMUNICATIEPROTOCOLLEN

> *Een netwerkprotocol is een protocol, een afgesproken communicatiewijze, voor netwerkcomponenten. Door het toepassen van een standaard protocol, kunnen componenten van verschillende leveranciers met elkaar gegevens uitwisselen.*
>
> *Een netwerkprotocol bevat afspraken omtrent:*
> - *identificatie van de verschillende communicerende componenten,*
> - *het onderhandelen over het tot stand komen van de communicatie,*
> - *de betekenis van de over en weer gezonden gegevens en informatieblokken,*
> - *het afbreken van de communicatiestroom.*
>
> *– bron: Wikipedia*

Voor het beschrijven van de uitwisseling tussen applicaties wordt vaak gebruik gemaakt van een lagenmodel (bijv. het OSI- of TCP/Ip lagenmodel) om verschillende typen functionaliteit te beschrijven. Binnen iedere laag kunnen verschillende protocollen worden gebruikt. Dit hoofdstuk beperkt zich tot een korte beschrijving van een een aantal veelgebruikte protocollen voor gegevensuitwisseling binnen de 'applicatielaag'[22]. Doel daarvan is om bij het inrichten van notificatieprocessen op basis van de aanwezige eisen te kunnen bepalen welk(e) protocol(len) het meest geschikt zijn om te gebruiken. Binnen protocollen is onderscheid te maken in standaardprotocollen (bijvoorbeeld HTTP, AMQP, MQTT, SMTP), open source protocollen (bijvoorbeeld Kafka, NATS) of platform- / leverancierssspecifieke protocollen (AWS Kinesis, Azure Event Grid).

Een belangrijk aspect van protocollen is hoe zij ondersteuning bieden voor het, waar van toepassing gestandaardiseerd, beschrijven van metadata (bijv. voor routering naar afnemers) en payloadgegevens (bijv. inhoudelijke informatie naar aanleiding van een plaatsgevonden gebeurtenis).

De tabel hieronder toont in de eerste kolom een aantal 'applicatielaag' protocollen. De tweede kolom toont een aantal protocollen of standaarden die voortbouwen op het vaak gebruikte HTTP protocol. Niet alle weergeven protocollen zijn even geschikt voor notificatieprocessen zoals die binnen de scope van project Notificatieservices vallen. Het feit dat sommige protocollen gedurende langere tijd 2-wegverkeer kunnen ondersteunen is daarbij bijv. vaak niet nodig. Maar bij andere type notificatieprocessen, die mogelijk in het vervolg op dit project aandacht gaan krijgen, kan dat type functionaliteit juist wel van belang zijn. Het hoofdstuk over 'patronen' beschrijft meer bij welk patroon bepaalde protocollen bruikbaar zijn.

[22] Binnen het OSI-lagenmodel de 'Applicatielaag' binnen het TCP/IP lagenmodel de 'Applicatie of Proces laag'

Protocol	**'Standaard'**	**Opmerking**
HTTP(S)[23]		Het Hypertext Transfer Protocol (HTTP) is een applicatielaag protocol voor gedistribueerde, collaboratieve hypermedia-informatiesystemen en de basis van datacommunicatie voor het World Wide Web.
	HTTP Polling	Bij HTTP polling wacht de cliënt op antwoord van de server om nieuwe informatie vragen via Short Polling (weinig gebruikt), Long Polling (vraag en blijf wachten) of Periodic Polling (om de zoveel tijd vraag herhalen).
	HTTP Streaming / Server-Sent Events	Bij HTTP Streaming is er een langdurige verbinding waarbij de server na het optreden van events gegevens pusht totdat de cliënt de verbinding verbreekt. Server-Sent Events is een (in ontwikkeling zijnde) standaard hiervoor.
	Webhooks	Door een afnemer gedefinieerde HTTP callback die een aanbieder gebruikt om via een HTTP POST methode notificatieberichten af te leveren.
	Digikoppeling RESTful API profiel	REST is een architectuurstijl die een aantal algemene eisen stelt aan hoe communicatie tussen cliënt en server plaatsvindt. Er zijn veel manieren om, min of meer, conform de REST eisen te communiceren. Digikoppeling's Restful API profiel, dat is gebaseerd op de REST-API Design Rules standaard, is een standaard die meer uitgewerkt RESTful uitwisselen specificeert.
	SOAP/WS-ReliableMessaging	RPC-Protocoll waarmee SOAP berichten met hoge zekerheid, ook bij storingen, worden uitgewisseld tussen gedistribueerde applicaties. Communicatieprotocol-neutraal maar maakt meestal gebruik van HTTP. Een van de Digikoppeling koppelvlakstandaarden.
	SOAP/ebMS	Op SOAP gebaseerd pakket-, routerings- en transportprotocol voor uitwisselen van op ebXML gebaseerde B2B-documenten.. Communicatieprotocol-neutraal maar maakt meestal gebruik van HTTP. Een van de Digikoppeling koppelvlakstandaarden.
	WebSockets	Het WebSocket protocol upgrade een standaard HTTP request/response naar een efficiënte permanente full-duplexverbinding waarmee, zolang als een sessie duurt, cliënt en server elkaar op elk moment berichten kunnen sturen. In tegenstelling tot HTTP biedt WebSocket full-duplex communicatie.
AMQP		Het Advanced Message Queuing Protocol is een open standaard messaging protocol dat is ontworpen om een breed scala aan berichtentoepassingen en communicatiepatronen efficiënt te ondersteunen.
MQTT		Message Queue Telemetry Transport (MQTT) is ontworpen als lichtgewicht berichtenprotocol voor gebruik via het publish-subscribe patroon. Het wordt vaak gebruikt binnen IOT-omgevingenn waarin, ook hele kleine, apparaten zoals sensoren gebeurtenissen signaleren en in de rol van aanbieder informatie daarover verstrekken.
(S)FTP		File Transfer Protocol om bestanden uit te wisselen via een op TCP/IP

[23]De hyperlinks in de kolom 'Protocol' verwijzen naar Wikipedia.

Protocol	'Standaard'	Opmerking
		gebaseerd netwerk zoals het internet. M.b.v. het TLS protocol is versleuteling en veilige identificatie van de server mogelijk. FTP kan bijv. worden gebruikt om notificaties in de vorm van een tekstbestanden op door de afnemers opgegeven locatie te plaatsen.
SMTP(S)		Simple Mail Transfer Protocol voor verzenden en ontvangen van email. Hoewel het protocol zich meer leent voor menselijke afnemers van notificaties (buiten scope van het project) kan het ook worden gebruikt om notificaties via mailberichten met bijlages te versturen die geautomatiseerd kunnen worden verwerkt.
Platform-specifieke protocollen		Met name binnen grote cloudomgevingen wordt vaak (ook) gebruik gemaakt van eigen protocollen (bijv. Amazon's AWS Kinesis en Microsofts Azure Event Grid). De voordelen van gebruik er van kunnen opwegen tegen de lock-in die ontstaat dus zijn binnen bepaalde situaties ook dit type protocollen kandidaat om te worden gebruikt.

De volgende paragrafen lichten een aantal protocollen toe die binnen de context van project Notificatieservices een belangrijke(re) rol kunnen spelen.

8.3 HTTP

HTTP is een synchroon request/reply protocol dat de basis van datacommunicatie voor het World Wide Web vormt. In documenten opgenomen hyperlinks verwijzen daarbij naar andere informatiebronnen. Binnen de Nederlandse overheid wordt het HTTP protocol intensief gebruikt voor het synchroon opvragen van gegevens. Deze paragraaf beschrijft een aantal standaarden die HTTP als basis gebruiken die bruikbaar kunnen zijn voor het realiseren van notificatieoplossingen.

8.3.1 Webhooks

> *Webhooks are "user-defined HTTP callbacks". They are usually triggered by some event, such as pushing code to a repository or a comment being posted to a blog. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. Users can configure them to cause events on one site to invoke behavior on another. ... The format is usually JSON. The request is done as an HTTP POST request. – bron: Wikipedia*

Webhooks zijn een middel waarmee aanbieders via een HTTP POST methode notificatieberichten aan afnemers leveren. Het is een relatief eenvoudig middel om via een push-mechanisme notificaties naar afnemers te sturen en wordt in de praktijk vaak toegepast[24]. Bijv. voor het verstrekken van openbare gegevens zoals nieuwsberichten.

Naarmate de eisen met betrekking tot vertrouwelijkheid en bedrijfszekerheid toenemen zijn bij gebruik van webhooks steeds meer extra maatregelen nodig. Een van de redenen daarvoor is dat aanbieder en afnemer synchroon communiceren wat, net als bij andere vormen van synchrone communicatie, leidt tot ongewenste afhankelijkheden. De afnemersapplicatie moet bijv. altijd bereikbaar zijn om de aanbieder

[24]Zie bjiv. 100 Webhook Implementations waarin 100 omgevingen worden beschreven waarin gebruik wordt gemaakt van webhooks.

notificaties te kunnen laten bezorgen en optredende storingen en fouten moeten correct worden afgehandeld. Maar er zijn bijv. ook maatregelen nodig om te voorkomen dat anderen dan de aanbieder berichten aflevert ('spoofing'), dat de ontvangende service overbelast raakt ('DDoS') of dat notificaties die vaker worden ontvangen onterecht opnieuw worden verwerkt ('deduplication').

Gebruik webhooks alleen als er geen hoge eisen aan betrouwbaarheid en bedrijfszekerheid nodig zijn.

Webhooks bouwen voort op HTTP en, eventueel REST, en leveren daarmee een zeer lichte vorm van standaardisatie waarmee nog steeds hele verschillende soorten notificatieoplossingen zijn te ontwikkelen. Gebruik van webhooks is mogelijk bij gebruik van verschillende patronen. Het WebSub patroon is een voorbeeld van een patroon waarbij op een aantal aspecten, gestandaardiseerd gebruik wordt gemaakt van webhooks.

8.3.2 Digikoppeling Restful API profiel

Aansluitend op de trend binnen de overheid om steeds vaker gebruik te maken van de REST-stijl voor gegevensuitwisseling tussen applicaties wordt de Digikoppeling standaard uitgebreid met een Restful API profiel dat is gebaseerd op de REST-API Design Rules standaard.

- Het Digikoppeling Rest API profiel is gericht op Machine-to-Machine (M2M) interactie via een gesloten dienst tussen overheidspartijen conform de algemene uitgangspunten van de Digikoppeling standaard en is wat betreft functionele toepassing vergelijkbaar met het Digikoppeling WUS profiel.
- Het Digikoppeling REST-API profiel conformeert zich volledig aan het normatieve deel van de API Design Rules standaard. Hierin zijn eisen beschreven met betrekking een aantal aspecten die een rol spelen bij de ontwikkeling en gebruiken van services.
- Een aantal regels uit de API Design Rules extensies zijn verplicht of aanbevolen. Hierbij gaat het met name over beveiliging, autorisatie en foutafhandeling.
- In versie 1.0 van het Digikoppeling REST API profiel wordt signing en encryptie niet ondersteund. In toekomstige versies van het profiel zal hier wel invulling aan worden gegeven.

Gebruik van dit profiel bij notificatieprocessen die gebruik maken van de REST-stijl leidt tot standaardisatie op een aantal aspecten. In bepaalde situaties is gebruik van de Digikoppeling standaard wenselijk of zelfs verplicht.

> *Digikoppeling moet worden toegepast op alle digitale gegevensuitwisseling met behulp van gestructureerde berichten die plaatsvindt met voorzieningen die onderdeel zijn van de GDI, waaronder de basisregistraties, of die sector-overstijgend is. Geautomatiseerde gegevensuitwisseling tussen informatiesystemen op basis van NEN3610 is uitgesloten van het functioneel toepassingsgebied. – bron: Lijst open standaarden – Forum Standaardisatie*

Het Digikoppeling REST API profiel is een wat ander type profiel dan de bestaande profielen dus moet nog duidelijk worden in welke situaties het met welke verplichting is toe te passen.

8.3.3 SOAP/WS-ReliableMessaging

8.3.4 SOAP/ebMS

8.3.5 Websocket

Een WebSocket is een standaard bidirectionele TCP-socket tussen de cliënt en de server. De socket begint als een HTTP-verbinding en wordt vervolgens "geüpgraded" naar een TCP-socket na een HTTP-handshake. Daarna kunnen beide partijen elkaar efficiënt realtime berichten verzenden. WebSocket is ontworpen om te werken via de standaard HTTP-poorten 443 en 80 en met HTTP-firewalls en proxy's .

Websocket is een lichtgewicht cross-platform protocol dat bruikbaar is binnen allerlei omgevingen (bijv. webapplicaties browsers, mobiele apps en IoT-devices).

8.4 AMQP

Het Advanced Message Queuing Protocol (AMQP) is een open standaard applicatielaag-protocol (net als bijv. HTTP) ter ondersteuning van messaging. Het is ontworpen om een breed scala aan berichtentoepassingen en communicatiepatronen flexibel en efficiënt te ondersteunen. Het biedt o.a. geavanceerde functionaliteit voor berichtuitwisseling[25], authenticatie, encryptie en garanties voor berichtaflevering[26].

Deze paragraaf beschrijft wat meer in detail dan bij andere protocollen wat AMQP aan mogelijkheden biedt. Een reden daarvoor is dat het duidelijk maakt dat berichtuitwisseling, ook bij notificeren, hele verschillende eisen kan stellen (waar in verschillende mate aan is te voldoen door verschillende protocollen). Een tweede reden is dat AMQP binnen de overheid nog weinig gebruikt wordt maar bij een migratie naar meer gebeurtenisgedreven werken een grotere rol kan gaan spelen omdat het daarvoor een bewezen standaard is.

AMQP is een binair protocol dat interoperabiliteit bevordert omdat het, zolang maar wordt voldaan aan de specificaties, door allerlei soorten applicaties gebruikt kan worden. Het is bruikbaar in heel verschillende situaties (van heel kleine schaal IOT-messaging tot hyper-scale cloudmessaging,) en bij gebruik van verschillende patronen (bijv. peer-to-peer, request-reply, publish-subscribe). AMQP wordt ondersteund door veel berichtgeoriënteerde middlewareproducten die functionaliteit bieden voor bijv. queing, routering (waaronder publish-subscribe), betrouwbaarheid en veiligheid.

Het protocol kent een gelaagde architectuur:

- een typesysteem
- de transportlaag; een efficiënt asynchroon, binair, peer-to-peer protocol voor transport van berichten over een netwerk
- een gestandaardiseerd uitbreidbaar berichtformaat
- een set gestandaardiseerde, en uitbreidbare, messaging capabilities.
- (security layers?)

Binnen AMQP worden connecties opgezet waarbinnen meerdere sessies mogelijk zijn. Binnen een sessie kunnen meerdere ‘links’ (verbindingen) worden gecreëerd om berichten uit te wisselen. Zowel op sessie- als link-niveau zijn limieten in te stellen (bijv. hoe groot berichten en througput mogen zijn en hoeveel berichten maximaal in- en uitgaand zijn te verwerken). Hiermee kan een robuuste verwerkingsomgeving

[25]Berichten kunnen bijv. naar afnemers worden verstuurd zonder vraag om respons maar als wel een respons nodig is kan een afnemer na verwerking van een ontvangen bericht bijv. aangeven of het bericht is ‘geaccepteerd’, ‘afgewezen’ of dat het niet te verwerken was.

[26]Mogelijke aflevergaranties zijn dat een bericht hoogstens 1 keer wordt afgeleverd (‘At-Most-Once’), minstens 1 keer wordt afgeleverd (‘At-Least-Once’) of precies 1 keer (‘Exactly-Once’) wordt afgeleverd.

worden ingericht die voorkomt dat er tijdens berichtuitwisseling fouten optreden als gevolg van onvoldoende systeem- of applicatie-resources[27].

Berichten kunnen via een eigen link in 1 richting worden verzonden terwijl status- en flowcontrol-informatie via een aparte link in 2 richtingen wordt uitgewisseld. Wanneer 2-weg berichtenverkeer nodig is kunnen 2 aparte links worden aangemaakt. Met behulp van een bericht- of correlatie-id kunnen dan bijv. full-duplex of request-response uitwisselpatronen worden ondersteund.

AMQP ondersteunt verschillende soorten berichtaflevering en daarbij behorende garanties:

1. een bericht wordt hoogstens 1 keer afgeleverd ('At-Most-Once') ('fire-and-forget') of
2. minstens 1 keer afgeleverd ('At-Least-Once') of
3. precies 1 keer ('Exactly-Once') afgeleverd.

Bij varianten 2 moet de ontvanger feedbackinformatie verstrekken naar de verzender door aan te geven of een bericht is:

- accepted (OK)
- rejected (met evt. toelichtende informatie)
- released (een verzoek om opnieuw verzenden van het bericht omdat er is iets misgegaan bij verwerking (bijv. een time-out))
- modified (released maar met informatie die aangeeft wat er veranderd moet worden aan het bericht voor opnieuw verzenden).

Als sessies door storingen worden beëindigd kunnen eerder aangemaakte links worden hersteld. Daarbij is fijnmazig per bericht, afhankelijk van de status, aan te geven wat van de andere partij verwacht wordt. Op deze manier kan een onverwacht onderbroken communicatieproces worden hervat vanaf het moment waarop de verstoring optrad.

Enkele kenmerken van AMQP:

- het kent een eigen typesysteem dat zowel schema-bound als schema-free is te gebruiken. Naast verplicht gebruik voor stuurgegevens in berichten kan het bijv. ook worden gebruikt om payload-gegevens in een gegevensformaat met beperkte functionaliteit om te zetten in AMQP gegevenstypes (JSON kent bijv. beperkte functionaliteit voor numerieke gegevens en timestamps).
- Er zijn veel mogelijkheden om notificatieprocessen flexibel en betrouwbaar in te richten. Een voorbeeld daarvan is de mogelijkheid voor afnemende applicaties om, bij de start of tussentijds, door te geven hoeveel berichten zij binnen een bepaald tijdsbestek maximaal willen ontvangen of om tussentijds een (tijdelijke) stop in te lassen. Beide zijn voorbeelden van de vele mogelijkheden aan 'flow control' dat AMQP biedt. AMQP biedt heel veel mogelijkheden om partijen met elkaar te laten communiceren en berichtuitwisseling fijnmazig 'op maat' in te richten.
- het is ontwikkeld binnen de financiële sector, was aanvankelijk een OAIS-standaard maar is sinds 2014 ook een ISO/IEC standaard. Documentatie er over is te vinden op amqp.org.

8.5 FTP

Het File Transfer Protocol is een al 50 jaar bestaand protocol om bestanden uit te wisselen. Eerder is beschreven dat notificeren via bestandsuitwisseling niet de voorkeur verdient. Maar in sommige situaties kan FTP wel degelijk bruikbaar of zelfs de enige beschikbare optie zijn.

[27]Bedenk hierbij dat AMQP zowel gebruikt kan worden door een klein IOT-device als binnen een 'hyper-scale' cloud-serveromgeving die miljoenen berichten moet kunnen verwerken.

Project Notificatieservices heeft als doel om gebeurtenisgedreven werken en notificeren breed binnen de overheid te stimuleren en door middel van standaardisatie te vergemakkelijken. Dit betekent dat rekening moet worden gehouden met partijen die beperkt zijn in hun mogelijkheden. FTP kan daarbij een passend protocol zijn voor verstrekken van notifcaties en/of ophalen van aanvullende informatie bij informatiearm notificeren.

8.6 SMTP

SMTP is het standaard protocol om email te verzenden en ontvangen. Net net als voor FTP geldt dat gebruik er van voor notificeren niet de voorkeur verdient maar dat het in bepaalde situaties wel kan voldoen.

Gebruik van email richting mensen valt buiten de scope van project Notificatieservices omdat notificeren van mensen hele andere eisen stelt dan notificeren van applicaties. Met de nodige maatregelen kan email worden gebruikt om communicatie tussen applicaties plaats te laten vinden. Bijvoorbeeld door in berichten of in bijlage gestructureerde notificatiegegevens op te nemen (bijv. In JSON- of XML-formaat). Door afnemers ontvangen emails kunnen dan bijv. via een polling-mechanisme automatisch worden verwerkt.

Wanneer er sprake is van vertrouwelijke gegevens in notificaties biedt gebruik van standaard email te weinig veiligheid en moet gebruik worden gemaakt van ‘veilige email’[28].

8.7 Protocollen en betrouwbaarheid

Bij communicatie tussen applicaties is het van belang dat berichten van een aanbieder door afnemers met een voldoende mate van zekerheid worden ontvangen. Afhankelijk van de context kan de gewenste mate van zekerheid variëren van ‘geen zekerheid’ tot ‘volledige zekerheid’. In geval van ‘geen zekerheid’ volstaat een weinig betrouwbaar uitwisselmechanisme en hoeft een aanbieder geen actie te ondernemen als een notificatie niet door een afnemer wordt ontvangen. Wanneer volledige zekerheid nodig is moeten daarvoor geschikte standaarden en voorzieningen worden gebruikt zodat afnemers notificaties gegarandeerd ontvangen. Van een aanbieder wordt verwacht dat hij controleert of een notificatie volgens afspraak door een afnemer is ontvangen en zo niet dat hij passende actie onderneemt (bijvoorbeeld door de notificatie opnieuw aan te bieden en na meerdere mislukte pogingen de afnemer daarover te informeren).

Protocollen zijn in verschillende mate geschikt geschikt om zekerheid van ontvangst te kunnen ondersteunen. Onderstaande tabel toont grofweg welke mate van zekerheid gebruik van een aantal bekende protocollen biedt.

Protocol	**Mate van zekerheid**	**Opmerking**
HTTP(S)	Matig tot Hoog	HTTP biedt zelf weinig mechanismen om zekerheid te garanderen. Met aanvullende maatregelen binnen de applicatielaag op te waarderen naar Hoog. Ondersteunt synchrone communicatie.
SMTP	Hoog	Ondersteunt asynchrone communicatie.
SOAP/WS-ReliableMessaging	Hoog	Protocol waarmee SOAP-berichten met hoge zekerheid, ook bij storingen, worden uitgewisseld tussen gedistribueerde applicaties.

[28]Er zijn in de praktijk verschillende manieren waarop communicatie via email veilig kan verlopen. Beschrijving daarvan valt buiten de scope van dit project.

Protocol	**Mate van zekerheid**	**Opmerking**
		Maakt meestal gebruik van HTTP maar kan ook SMTP gebruiken. Ondersteunt synchrone en asynchrone communicatie.
SOAP/ebMS	Hoog	Op SOAP gebaseerd pakket-, routerings- en transportprotocol. Open standaard en als zodanig communicatieprotocol-neutraal. Maakt meestal gebruik van HTTP en kan ook SMTP gebruiken. Geschikt voor uitwisselen van op ebXML gebaseerde B2B-documenten. Ondersteunt synchrone en asynchrone communicatie. Een van de Digikoppeling koppelvlakstandaarden.
AMQP	Hoog	Het Advanced Message Queuing Protocol is een open standaardprotocol dat werkt op de applicatielaag, waardoor berichtoriëntatie, wachtrijen en routering mogelijk zijn en interoperabiliteit, betrouwbaarheid en beveiliging biedt voor het verzenden en ontvangen van berichten tussen applicaties.
(S)FTP		Standaard netwerkprotocol om bestanden uit te wisselen via een op TCP / IP gebaseerd netwerk zoals internet. M.b.v. het TLS-protocol is versleuteling en veilige identificatie van de server mogelijk.
SMTP(S)		Protocol voor verzending en ontvangst van email.
HTTP/REST	Matig	Op te waarderen via applicatielogica bij zowel aanbieder als afnemers (zie toelichting hierna).

Protocollen die een hoge zekerheid van ontvangst bieden kennen functionaliteit die de kwaliteit van uitwisseling vergroot. Zij kunnen bijvoorbeeld garanderen dat een verzonden bericht minstens, hooguit of precies 1 keer wordt ontvangen en dat berichten altijd in volgorde van verzending aankomen. Wanneer zich fouten voordoen, zoals wanneer een bericht niet is af te leveren of er geen ‘bericht van ontvangst’ (‘acknowledgement’) wordt ontvangen, kan automatisch passende actie worden ondernomen (bijv. het op een later moment opnieuw versturen van het bericht).

HTTP is een veel gebruikt protocol voor communicatie maar biedt als protocol weinig zekerheid van aflevering. Protocollen zoals SOAP/WS-ReliableMessaging en Soap/ebMS leveren, bovenop HTTP, wel dat type functionaliteit waarmee ontvangst met hoge zekerheid mogelijk wordt.

REST is formeel geen protocol maar een architectuurstijl die (meestal) gebruik maakt van het HTTP-protocol. Gebruik van de REST-stijl levert geen extra functionaliteit ten opzichte van HTTP voor zekerheid van ontvangst. Wanneer alles goed gaat kan een aanbieder op basis van een statuscode en/of responsbericht concluderen dat aflevering is geslaagd. Er is echter geen functionaliteit die garandeert dat een verzonden bericht minstens, hooguit of precies 1 keer wordt ontvangen, berichten in volgorde van verzending aankomen en het voorkomen van ongewenste bijeffecten als een bericht via een niet-idempotente methode (bijv. POST) vaker wordt verstuurd.

Omdat HTTP geen functionaliteit voor hoge zekerheid van ontvangst biedt moet bij gebruik van REST-services de benodigde functionaliteit door de betrokken applicaties worden gerealiseerd. Betrouwbare levering kan bijvoorbeeld worden bereikt door binnen de applicatielogica rekening te houden met foutsituaties. De aanbieder moet foutsituaties kunnen constateren en daar goed mee omgaan. Bijv. door berichten die mogelijk niet goed zijn afgeleverd met een bepaald interval opnieuw te versturen (met meestal een maximum aantal retries). De afnemer moet zorgen dat bij herhaalde aanroep van niet-idempotente methodes zoals POST geen ongewenste bijeffecten optreden, zoals het onterecht vaker opvoeren van dezelfde resource. Dit kan bijvoorbeeld worden gedaan door in het bericht een uniek identificerend gegeven op te nemen dat de afnemer gebruikt om al eerder verwerkte berichten niet opnieuw te verwerken.

Het is dus mogelijk om bij gebruik van de REST-stijl meer zekerheid te realiseren dan het HTTP-protocol standaard biedt maar het is niet eenvoudig en vereist speciale acties bij zowel aanbieder als afnemers. Voor situaties waarin zekerheid van aflevering van belang is, is gebruik van HTTP/REST dus minder geschikt.

Wanneer zekerheid van ontvangst belangrijk is zijn onderstaande protocollen geschikt:

- SOAP/WS-ReliableMessaging
- SOAP/ebMS
- AMQP.

Voor SOAP/ebMS geldt dat de Stelseldienst Digikoppeling als profiel (ook) de ebMS-standaard omvat. Die wordt meestal gebruikt voor asynchroon berichtenverkeer (‘meldingen’). Daarbij is meestal niet direct een antwoord te geven. De ontvanger krijgt eerst een bevestiging dat zijn bericht ontvangen is. Later volgt het antwoord. Applicaties identificeren zich bij gegevensuitwisseling met Digikoppeling met een speciaal PKIoverheid-certificaat. Digikoppeling is daarmee geschikt voor het veilig en betrouwbaar verzenden van berichten.

Advanced Message Queuing Protocol (AMQP) een bericht-georiënteerd protocol. Andere voorbeelden daarvan zijn. XMPP en MQTT. Welk protocol het meest geschikt is contextafhankelijk. Voor notificaties vanuit bronregistraties naar processystemen bij andere organisaties is kan AMQP een geschikt bericht-georiënteerd protocol zijn. Een protocol als AMQP bevat uitgebreide functionaliteit om veilige en betrouwbare uitwisseling van berichten mogelijk te maken. Juist ook binnen situaties waarin applicaties elkaar notificaties sturen.

Tot slot geldt dat binnen notificatieprocessen ook meerdere protocollen gebruikt kunnen worden. Voor de communicatie tussen een aanbieder en een makelaar kan bijvoorbeeld gebruik worden gemaakt van een betrouwbaar protocol zoals ebMS of AMQP terwijl een makelaar voor communicatie met afnemers gebruik maakt van HTTP/REST.

Bijlage: Archimate elementen

De Archimate architectuurbeschrijvingstaal onderscheidt (onder andere) een bedrijfs-, applicatie- en technologielaag. De bedrijfslaag beschrijft actoren, hun gedrag en de diensten die aan de buitenwereld worden geleverd. Bij het uitvoeren van bedrijfsprocessen wordt gebruik gemaakt van diensten uit de applicatielaag. De applicatielaag op haar beurt maakt gebruik van diensten uit de technologielaag.

Deze bijlage beschrijft een aantal elementen uit de bedrijfs- en applicatielaag. Per element wordt kort beschreven wat de betekenis is en wat de betekenis binnen de context van notificeren is. Daarna volgt een korte toelichting op hoe elementen uit beide lagen samenwerken.

Actor (‘business actor’): A business actor represents a business entity that is capable of performing behavior. Bijv. een organisatie, afdeling, functionaris. Bij notificeren zijn ‘actoren’ meestal 'organisaties’ of, als het gaat om notificeren binnen een organisatie, ‘afdelingen'.

(Bedrijfs)rol (‘business role’): A business role represents the responsibility for performing specific behavior, to which an actor can be assigned, or the part an actor plays in a particular action or event. Bijv. dienstaanbieder, dienstafnemer, leverancier, klant. Bij notificeren kennen we meerdere rollen die al dan niet bij verschillende organisaties belegd kunnen zijn. De 3 belangrijkste rollen duiden we aan met: ‘(dienst)aanbieder’, ‘(notificaties)makelaar’ en ‘(dienst)afnemer’. We sluiten hiermee qua terminologie aan bij de NORA en haar keuze voor de servicegeorienteerdearchitectuurstijl.

Bedrijfsgebeurtenis (‘business event’): A business event represents an organizational state change. Bijv. “klacht ontvangen” of “vergunning aangevraagd”. Voor bedrijfsgebeurtenissen geldt dat ze:

- Hun oorsprong binnen of buiten de organisatie kunnen hebben (bijv. “aanvraag afgewezen” of “prullenbak vernield”).
- Aanleiding kunnen zijn om een proces te starten (bijv. “verzenden brief n.a.v. beschikking” of “verlenen opdracht voor reparatie prullenbak”).
- Geen tijdsduur hebben en in de voltooid verleden tijd worden geformuleerd (bijv. “prullenbak vernield” en niet “vernieling prullenbak”).

Bedrijfsgebeurtenissen hebben betekenis op bedrijfsniveau. De beschrijving daarvan moet herkenbaar zijn voor business-vertegenwoordigers. Om communicatie daarover te bevorderen gebruiken we eenduidige taal en terminologie, te gebruiken door zowel business-vertegenwoordigers als softwareontwikkelaars.

In het dagelijks spraakgebruik worden ‘bedrijfsgebeurtenis’ vaak aangeduid met de afkorting 'gebeurtenis'. Reden om ze expliciet te benoemen als ‘bedrijfsgebeurtenissen’ is dat in de context van notificeren ook andere type gebeurtenissen, ‘applicatiegebeurtenissen’ een rol spelen.

(Bedrijfs)proces (‘business process’): A business process represents a sequence of business behaviors that achieves a specific result such as a defined set of products or business services. Bijv. het behandelen van een klacht of een vergunningaanvraag. Een bedrijfsproces wordt vaak gestart naar aanleiding van de notificatie dat een bepaalde gebeurtenis heeft plaatsgevonden. Voor sommige processen geldt ze zo snel mogelijk na het plaatsvinden van een gebeurtenis moeten worden gestart. In die situaties is het dus van belang dat notificatie snel plaatsvindt.

Bedrijfsdienst (‘business service’): A business service represents explicitly defined behavior that a business role, business actor, or business collaboration exposes to its environment. Bijv. facturering. Bedrijfsdiensten vormen de meerwaarde die organisaties aan de buitenwereld leveren. BIj notificeren leveren aanbieders aan afnemers bijvoorbeeld als bedrijfsdienst: ‘beschikbaar stellen van relevante notificaties’. We richten ons nu primair op notificatieprocessen die geautomatiseerd plaatsvinden. De beschrijving van het geautomatiseerde deel gebeurt dan ook voornamelijk met behulp van elementen uit de applicatielaag. Maar het blijft belangrijk om ook bedrijfsmatige aspecten goed in beeld te brengen (zie ook 'contract' en 'product').

Contract: A contract represents a formal or informal specification of an agreement between a provider and a consumer that specifies the rights and obligations associated with a product and establishes functional and non-functional parameters for interaction. Bijv. een Service Level Agreement (SLA) of Gegevens Levering Overeenkomst (GLO) die de afspraken beschrijft met betrekking tot het beschikbaar stellen van notificaties door een aanbieder aan afnemers.

Voorziening (‘product’): “A product represents a coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers.” Bijv. de voorziening Digilevering die een verzameling bedrijfs-, applicatie- en technologiediensten en een daarbij behorende Serviceniveau Overeenkomst bevat. Bij geautomatiseerde notificatieprocessen is altijd sprake van applicatiediensten. Spreken over ‘voorziening’ benadrukt dat het alleen het technisch werkend krijgen van gegevensuitwisseling tussen applicaties via applicatiediensten niet voldoende is maar dat ook zaken als serviceafspraken en verantwoordelijkheden goed geregeld moeten zijn. Applicatielaag

Applicatie(component): An application component represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable. Bijv. een vergunningenapplicatie.

Een applicatiecomponent voert één of meer applicatiefuncties uit. Bijvoorbeeld om een geautomatiseerd notificatieproces mogelijk te maken.

We richten ons op gegevensuitwisseling tussen applicaties (en niet op uitwisseling binnen 1 applicatie). We spreken daarom meestal over ‘applicatie’ en laten het achtervoegsel ‘component’ weg.

In bepaalde contexten kunnen als synoniem worden gebruikt: (informatie)systeem en (software)component.

(Applicatie)interface: An application interface represents a point of access where application services are made available to a user, another application component, or a node. Bijv. de BAG API Individuele Bevragingen.

Binnen de context van dit project richten we ons op notificaties die worden uitgewisseld tussen applicaties (en dus niet notificaties voor mensen). Applicaties kennen applicatie-interfaces waarmee andere applicaties met hen kunnen communiceren. Interfaces kunnen met behulp van verschillende stijlen en technieken gebruik maken (bijv. REST-API’s, remote procedure calls, FTP).

Applicatiegebeurtenis: An application event represents an application state change. Bijv. ‘orderregel aangemaakt’.

In tegenstelling tot ‘bedrijfsgebeurtenissen’ gaat het hier om meer technische gebeurtenissen die plaatsvinden tijdens de geautomatiseerde verwerking door applicaties. Ze zijn niet direct zichtbaar voor eindgebruikers. Afhankelijk van de behoeften van afnemers kunnen ze wel of niet van belang zijn om te

notificeren. Het kan bijv. wenselijk zijn om iedere creatie van een bepaald type record te notificeren. Maar het is ook denkbaar dat afnemers niet op dit niveau notificaties willen ontvangen maar bijv. een notificatie willen ontvangen na een afgeronde transactie waar creatie van een record onderdeel is. Applicatiegebeurtenissen kunnen extern (door een andere applicatie) of intern (binnen de applicatie) worden getriggerd. Net als voor bedrijfsgebeurtenissen geldt ze geen tijdsduur hebben en in de voltooid verleden tijd worden geformuleerd (bijv. “zaakrecord aangemaakt”).

Applicatiedienst: An application service represents an explicitly defined exposed application behavior. Bijv. een zaak-aanmaak service.

Applicaties stellen in de vorm van applicatiediensten functionaliteit beschikbaar voor andere applicaties. Binnen de context van notificeren kan een specialistische applicatie bijv. applicatiediensten aanbieden die andere applicaties kunnen gebruiken om afnemers te notificeren.

Dataobject: A data object represents data structured for automated processing.

Een data-object is een op zichzelf staand informatieobject met een duidelijke betekenis op applicatieniveau en liefst ook op businessniveau. Bijv. een ‘klantrecord’, een ‘klantendatabase’ of een ‘verzekeringsclaim’. Notificaties zijn een voorbeeld van een dataobject dat applicaties met elkaar uitwisselen.

Samenwerking van elementen uit bedrijfs- en applicatielaag.

Net zoals bij veel andere processen spelen bij notificatieprocessen zowel elementen uit de bedrijfs-, applicatie- technologielaag betrokken. In de voorbereidende fase maken betrokken actoren in de vorm van een contract afspraken over welke bedrijfsdiensten ze gaan verlenen respectievelijk afnemen. Daarbij spelen bedrijfsmatige aspecten een rol (bijv. Leveringsafspraken en kosten) en applicatie-gerelateerde aspecten (bijv. protocollen, berichtformaat en beschikbaarheid). Applicaties bij de dienstaanbieder zorgen daarna dat er conform afspraken notificaties beschikbaar worden gesteld en applicaties van dienstenafnemers zorgen voor het verwerken van die notificaties. Voor deze gegevensuitwisseling worden applicatieservices gebruikt die met behulp van overeengekomen applicatie-interfaces met elkaar communiceren.

Het inrichten een goed notificatieproces vereist dus dat er zowel op bedrijfsmatig als applicatietechnisch niveau goede keuzes en afspraken worden gemaakt. Waar keuzes of aanbevelingen worden beschreven met betrekking tot applicaties moet dus altijd bedacht worden dat bedrijfsmatige factoren ook een rol spelen. Pas dan is goed te bepalen of een keuze of aanbeveling in de gegeven situaties de beste keuzes is.

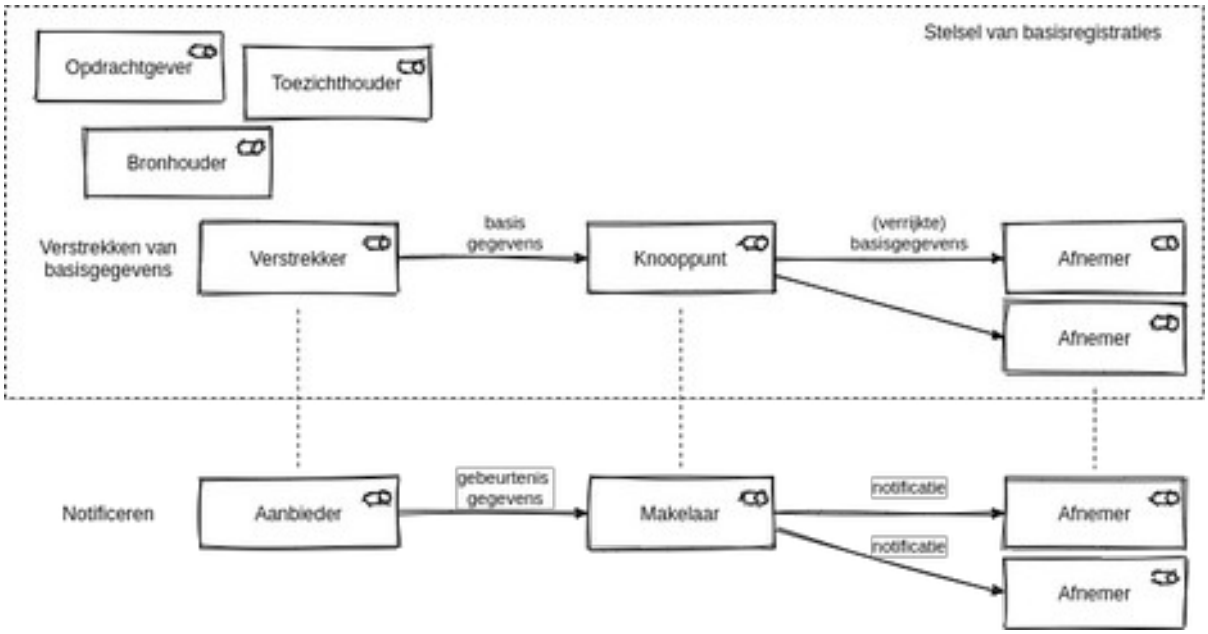
Bijlage: Relatie met basisregistraties

Binnen project Notificatieservices wordt nadrukkelijk gekeken naar hoe notificeren kan helpen om organisaties op de hoogte stellen van bepaalde wijzigingen binnen bronregistraties. De basisregistraties vormen een belangrijke categorie bronregistraties. Voor beheer en gebruik van basisregistraties zijn binnen de overheid een aantal 'stelselrollen' benoemd. Deze bijlage beschrijft kort hoe de rollen '(dienst)aanbieder', '(notificatie)makelaar' en '(dienst)afnemer' zich verhouden tot de stelselrollen zoals die voor de basisregistraties zijn benoemd.

Opdrachtgever	De opdrachtgever is het voor de basisregistratie verantwoordelijke ministerie, dat opdrachtgever is voor de 'verstrekker' (de beheerder van de landelijke voorziening). Bijvoorbeeld: het ministerie van BZK is opdrachtgever voor de Basisregistratie Personen.
Toezichthouder	De toezichthouder is de partij die er verantwoordelijk voor is dat wordt toegezien of de basisregistratie in overeenstemming met eisen, afspraken en wetgeving opereert. Een basisregistratie heeft één of meer verantwoordelijken voor toezicht. Over het algemeen is de opdrachtgever verantwoordelijk voor het toezicht op de naleving van de wettelijke bepalingen die gelden voor die basisregistratie.
Bronhouder	Een bronhouder is verantwoordelijk voor het inwinnen en bijhouden van de authentieke en niet-authentieke gegevens in een basisregistratie en voor het borgen van de kwaliteit van die gegevens (onder meer naar aanleiding van ontvangen terugmeldingen). Een basisregistratie heeft één of meer bronhouders.
Verstrekker	De verstrekker is verantwoordelijk voor het verstrekken van de gegevens aan afnemers. De verstrekker is ook verantwoordelijk voor het faciliteren van het gebruik (zoals het leveren van kennis en ondersteuning aan afnemers voor het aansluiten op de landelijke voorziening). Een basisregistratie heeft één verstrekker.
Afnemer	Een afnemer (ook wel 'gebruiker') is een overheidsorganisatie of private partij die gegevens afneemt van een basisregistratie voor gebruik in de eigen processen. Voor bestuursorganen met een publiekrechtelijke taak (zoals gemeenten, provincies, waterschappen en zelfstandige bestuursorganen) is het afnemen en gebruiken van relevante authentieke gegevens verplicht.[1]

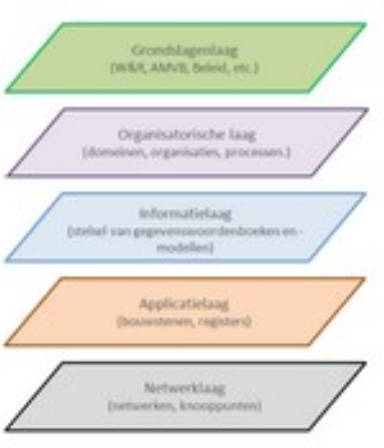
Verder wordt de rol van '*knooppunt*' onderscheiden: "een proactieve intermediair (makelaar) tussen de houders van bronnen (waaronder de basisregistraties) en de afnemende organisaties. Het regelt de gegevenslogistiek (integratie, conversie en distributie) en beheert de afspraken en gemeenschappelijke voorzieningen."[2]

De onderscheiden stelselrollen zijn bedoeld om de *registratie, beheer en verstrekking van basisgegevens* te optimaliseren en niet specifiek het notificeren van afnemers naar aanleiding van plaatsgevonden gebeurtenissen. Uitgaande van de stelselrollen geldt dat bij notificeren een 'verstrekker' vaak de rol van '(dienst)aanbieder' heeft, een 'knooppunt' de rol van '(notificatie)makelaar' en een 'afnemer' de rol van '(dienst)afnemer'. In de praktijk zijn ook andere rolinvullingen mogelijk.



Waar het relevant is kan naast de gebruikelijke rollen van ‘aanbieder’, ‘makelaar’ en ‘afnemer’ ook gebruik worden gemaakt van de stelselrollen zoals die binnen het stelsel van basisregistraties zijn gedefinieerd.

Bijlage: Relatie met NORA, GO en Forum Standaardisatie



Op een hoger plan brengen van gebeurtenisgedreven werken en notificeren binnen de overheid vereist maatregelen op verschillende niveau's. In termen van het vijflagenmodel van de NORA zijnop ieder ieder van de vijf lagen verbeteringen mogelijk. Project Notificatieservices richt zich met name op de informatie- en applicatielaag en beperkt op de organisatielaag. Als tijdens het project naar voren komt dat veranderingen binnen andere lagen nodig zijn wordt dit vastgelegd en gerapporteerd.

Binnen de Gemeenschappelijke Overheidsarchitectuur (GO), de architectuur voor doorontwikkeling van de Generieke Digitale Infrastructuur, is een van de principes: 'Afspraken voor standaarden voor voorzieningen'. Om overheidsbreed efffectiever gebeurtenisgedreven te kunnen gaan werken zijn overheidsbrede afspraken en standaarden nodig. Project Notificatieservices beoogt een aantal van de benodigde standaarden op te leveren. In de voorzieningsfeer geldt dat bij verschillende volwassenheidsniveaus erschillende typen voorzieningen passen. Deze kunnen zowel een overheidsbreed (bijv. Digilevering voor notificaties uit de basisregistraties) als een sectoraal of organisatie-eigen karakter hebben.

Vanuit project Notificatieservices wordt contact onderhouden NORA-beheer en de GO-kerngroep om te zijner tijd te bespreken hoe (ook tussentijdse) projectresultaten kunnen worden besproken, afgestemd en verwerkt.

Met betrekking tot standaarden geldt dat het Forum Standaardisatie open standaarden kan toetsen en voor kan schrijven aan publieke organisaties. Het ligt in de lijn der verwachting dat een van de resultaten van project Notificatieservices is om een of meer standaarden aan te melden voor opname op de lijst van aanbevolen of verplichte open standaarden.

Bijlage: Middleware

Er zijn legio softwareproducten te koop die kunnen ondersteunen om asynchrone communicatie tussen applicaties, bijv. tussen aanbieder en afnemers van notificaties, te ondersteunen. In veel situaties is gebruik van dit type software aan te raden om applicaties te kunnen ontkoppelen en om de, al snel complexe, functionaliteit voor asyncrhone communicatie over te laten aan daarin gespecialiseerde applicaties.

Een aantal bekende en veelgebruikte producten die hiervoor bruikbaar zijn bijvoorbeeld: Dell Boomi, TIBCO Enterprise Message Service, MuleSoft Anypoint Platform, IBM MQ, Apache Kafka, IBM Event Streams, Amazon Simple Queue Service, RabbitMQ, Azure Queue Storage, Apache RocketMQ, Ably, Solace PubSub+.

Door leveranciers worden ook, steeds vaker als SaaS, oplossingen aangeboden waar dit type producten onderdeel van zijn. Organisaties kunnen op die manier worden ontzorgd en een aantal, met name systeemtechnische, taken uitbesteden.

Waar eerder vaak gekozen werd voor een bepaald softwareproduct wordt, met name in cloudomgevingen, steeds vaker gebruik gemaakt van een combinatie van producten. In veel gevallen zijn dit leverancier-eigen producten die als op elkaar afgestemde component met afgebakende functionaliteit tot een werkende oplossing zijn te combineren. Microsoft's Azure kent bijv. De 'Azure Integration Services' met daarin producten als Azure Logic Apps with Azure API Management, Azure Service Bus ('a message queuing and publish-subscribe service') en Azure Event Grid ('a massive-event ingestion service') en een groot aantal andere meer generieke producten die bruikbaar kunnen zijn bij het maken van oplossingen (bijv. Azure Data Factory ('a serverless data integration service'), Power Automate ('citizen integrator tool built on top of Azure Logic Apps'). In tegenstelling tot de meer klassieke benadering om van 1 product(suite) gebruik te maken vindt bij gebruik van dit type platforms 'assemblage van componenten tot een werkende oplossing' plaats.