Handreiking "Hoe bouw ik een register" 0.0.1

1. Leeswijzer

Hier kan een toelichting opgenomen worden om de status en intentie van dit document toe te lichten.

1.1 Paragraaf 2

Dit wordt ook opgenomen in de index.

- Je kan bijvoorbeeld hoofdstuk 1 wat toelichten.
- En dan hier over Hoofdstuk 2 iets toelichten
- En hier over Hoofdstuk 3 en 4 en 5
- Uiteraard zou je hier ook een toelichting op kunnen nemen over hoe dit document gebruikt kan worden. Bijvoorbeeld dat het geen "menu" of "kookboek" voor een capabel register is, maar dat het een aantal onderwerpen beschrijft en daarbij overwegingen meegeeft om uiteindelijk tot weloverwogen keuzes te kunnen komen.

Inhoudsopgave

2. Command

- Werkdefinitie: een command is een opdracht aan het register, aangeboden met de intentie daarin een verandering aan te brengen.
- Aangepast vanwege een test of zeven

2.1 Twijfel en dry-run

- Aan gegevens in een capabel register kan getwijfeld worden. Dit heeft gevolgen voor het geautomatiseerd verwerken van commando's. Systemen kunnen immers niet (altijd) (geautomatiseerd) commando's verwerken als bij het beoordelen daarvan betwijfelde gegevens betrokken zijn. Dit is niet per se een registervraagstuk. Wel een vraagstuk dat zich op dit moment veelal buiten registers - en wel binnen procesondersteunende systemen - afspeelt.
- Willen we registers en degenen die ze bedienen (beter) laten werken met betwijfelde gegevens, dan is één oplossing het aanbieden van interfaces die het mogelijk maken daarover met het register te converseren. Wanneer sprake is van twijfel over de juistheid van relevante gegevens, kan in zo'n conversatie van verwerkingsregels worden afgeweken. Mits

beargumenteerd en voorzien van instructies over hoe en waar specifieke regels wel of niet moeten worden toegepast, wordt een waarschuwing of error zo dus 'overrulebaar'.

- Hieraan gerelateerd is het idee van 'dry run-functionaliteit'. Dit is een query die antwoord geeft op de vraag of een command verwerkt kan worden zonder een poging te doen dat command daadwerkelijk te verwerken. Om verschillende uitkomsten tussen 'dry run' en daadwerkelijke registratie te voorkomen, ligt het - ondanks scheiding tussen command en query (CQRS) - voor de hand deze query te laten beantwoorden aan de commandkant van het register, waar immers ook de daadwerkelijke verwerking van het command moet plaatsvinden.

2.2 Transacties

Conventie (en aanbeveling?):

- Command is een opdracht die één transactie verwerkt moet worden.

...maar: business bepaalt in eerste instantie transactiegrenzen, commands sluiten daarbij aan.

...maar: digitaal werken kan innovatie in business mogelijk maken. Sneller (en sequentieel) verwerken van wijzigingen in 'echte' wereld kan bijvoorbeeld consistentiewaarborgen van aggregaatniveau naar registerniveau (dat daardoor in feite zelf aggregaat wordt?) brengen.

3. De omgeving en grenzen van een register

Een register heeft betekenis door het proces dat het ondersteunt. Om te begrijpen waar de grenzen liggen, beginnen we bij het vijflaagsmodel van Common Ground:

Laag	Naam	Wat
5	**Interactie**	Gebruikersinterfaces, kanalen en toegang
4	**Procesinrichting**	Bedrijfsprocessen, bedrijfsregels en informatiestructuren
3	**Connectiviteit**	Veilige verbindingen
2	**Diensten**	Datadiensten, APIs en services
1	**Databronnen**	Registers, bijhouding en vastlegging
(**0**	**Infrastructuur**	Hardware en infrastructuur)

Een register bevindt zich in laag 1 (databronnen) en laag 2 (diensten). Een register sluit aan bij laag 4 (procesinrichting) door de **informatiestructuren** van het proces: de conceptuele modellen die beschrijven welke informatie het proces nodig heeft en produceert.

3.1 Autonomie en gemeenschappelijkheid

Binnen Common Ground ligt de **autonomie van gemeenten** bij de interactielaag (hoe burgers worden bediend) en de procesbesturing (hoe het werk wordt georganiseerd). Elke gemeente kan eigen keuzes maken in werkwijze en gebruikersinteractie.

De ***common ground*** - wat gemeenschappelijk is - ligt bij het domeinmodel: welke acties mogelijk zijn binnen een domein en welke gevolgen deze hebben. Dit zorgt voor interoperabiliteit terwijl gemeenten hun autonomie behouden.

Gemeenten zijn hier gebruikt als voorbeeld, want dit geldt voor alle organisaties waar autonomie en gemeenschappelijkheid een rol spelen, oftewel een gedeeld domein betreffen.

3.2 Registergrenzen: domein

Een register is een verzameling geordende informatie. De grenzen van de verzameling worden net als de betekenis en samenhang van de in het register opgeslagen informatie bepaald vanuit het **domein** dat voor het register verantwoordelijk is. Anders dan in de wiskunde kunnen we voor registers niet precies aangeven waar het ene domein ophoudt en het volgende begint. Analoog aan de definitie van *Eric Evans in de context van Domain Driven Design* beschouwen we een domein daarom als "een sfeer van kennis, invloed of activiteit". Deze definitie veronderstelt een zekere mate van samenhang. Tegelijkertijd kan het binnen één domein voorkomen dat:

- dezelfde concepten verschillend geïnterpreteerd worden,
- regels en gedrag niet consistent zijn, en
- bedrijfsprocessen niet op elkaar zijn afgestemd.

Voor wie zoekt naar aanknopingpunten over welke informatie binnen 'hoort' binnen een te ontwerpen register en welke buiten de registergrenzen zou moeten vallen, biedt het denken in 'sferen' onvoldoende aanknopingspunten. Daarom introduceren we **bounded context** als begrenzend begrip. Ook dit concept is ontleend aan [*domain-driven-design*].

3.3 Bounded context

Binnen een domein kunnen meerdere subdomeinen of taakgebieden bestaan. Daarbij horen verantwoordelijkheden, die zijn toegekend aan specifieke teams, afdelingen of bedrijfseenheden. Zij worden ondersteund door eigen semantiek, processen en regels. Deze zaken kunnen worden beschreven in een **model**: een systeem van abstracties dat beschrijft hoe men vanuit een taakgebied naar de wereld kijkt en reageert op veranderingen in de buitenwereld. Zo'n model is beschreven in **gemeenschappelijke taal** die binnen het hele taakgebied begrepen wordt. Model en taal beschrijven dus een voor betrokkenen herkenbare 'blauwdruk' van het taakgebied, die (onder

andere) de basis kan vormen voor het ontwerp van een register. Een in gemeenschappelijke taal ondubbelzinnig en samenhangend gemodelleerd taakgebied noemen we een '**bounded context**'.

Het belang van het erkennen van bounded contexten wordt door Eric Evans in *'the Blue Book'* als volgt beschreven:

> "A bounded context delimits the applicability of a particular model so that team members have a clear and shared understanding of what has to be consistent and how it relates to other contexts. Within that context, work to keep the model logically unified, but do not worry about applicability outside those bounds. In other contexts, other models apply, with differences in terminology, in concepts and rules, and in dialects of the ubiquitous language *[red. gemeenschappelijke taal]*. By drawing an explicit boundary, you can keep the model pure, and therefore potent, where it is applicable. At the same time, you avoid confusion when shifting your attention to other contexts. Integration across the boundaries necessarily will involve some translation, which you can analyze explicitly."

Uit het bovenstaande blijkt dat de ambiguïteiten die we op domeinniveau nog konden tegenkomen binnen een bounded context (idealiter) verdwijnen. Hier geldt (zoveel mogelijk) dat:

- dezelfde concepten eenduidig geïnterpreteerd worden (op basis van gemeenschappelijke taal),
- regels en gedrag consistent zijn, en
- bedrijfsprocessen op elkaar aansluiten.

3.4 'Wellbounded' registers

Bestaande registers bestrijken vaak meerdere bounded contexten. Zo zou je bijvoorbeeld binnen de Basisregistratie Personen (BRP) bijgehouden informatie over verblijfplaats, verblijfsrecht, kiesrecht en reisdocumenten ieder als 'eigen' bounded context kunnen beschouwen. Als je dat onderschrijft en op basis daarvan nieuwe registers zou gaan ontwerpen en ontwikkelen, loop je tegen een aantal (nieuwe) problemen aan:

- **Integratiecomplexiteit** (met name ten opzichte van registers met een 'breder' bereik). Registers moeten worden verbonden middels API's, events of gegevenssynchronisatie.
- **Hoge initiële leercurve.** Ontwerp en ontwikkeling van registers vereist diepgaande domeinkennis.
- **Lagere ontwikkelsnelheid.** Opdoen van domeinkennis en omzetten daarvan naar modellen en code vraagt tijd.

Tegelijkertijd worden aan overheidsregisters bijzondere eisen gesteld, onder meer als het gaat om kwaliteit en betrouwbaarheid. Onderstaande voordelen van het koppelen van registers aan een (of één) bounded context ondersteunen deze eisen, en wegen wat ons betreft zwaarder dan de nadelen:

- **Duidelijkheid**. Registers omvatten ondubbelzinnige terminologie en logica.
- **Consistentie**. Registers omvatten logisch consistente set regels en modellen.
- **Autonomie**. Registers in verschillende bounded contexten kunnen onafhankelijk van elkaar werken en (door)ontwikkeld worden.

Daarom doen we de volgende aanbevelingen voor het begrenzen van registers:

1. Een register valt samen met een (en één) bounded context, en dus
2. bepalen de bijhoudingsverantwoordelijke(n) de registergrenzen

3.5 Bepalen van de 'bounds'

In de praktijk laten de 'bounds' van een context zich niet altijd gemakkelijk herkennen en afbakenen. Bovendien kunnen bounded contexten onder invloed van impliciete of uitgesproken belangen worden 'opgerekt' of juist verkleind. In algemene zin is moeilijk te zeggen wanneer voor een register de 'ideale' bounded context is gevonden. Wel zijn enkele aanwijzingen voor een te ruime of te krappe afbakening te geven. Deze zijn hieronder beschreven.

Indicatoren en symptomen (van sterk naar zwakker) voor een te ruime bounded context:

1. **Gebrek aan gemeenschappelijke taal.** Er is discussie over concepten en betekenis, betrokkenen hanteren verschillend jargon.
2. **Regels en gedrag niet in samenhang te brengen.** Het lukt niet tot een samenhangend model te komen.
3. **Afwijkende werkwijze(n).** Bedrijfsprodcessen zijn niet te verenigen.
4. **Ontwerp-/owikkeltraject heeft onwenselijke overhead.** Het ontwerp-/ontwikkelteam is te groot (geworden).
5. **(Vermijdbare) overschrijding van team- en/of organisatiegrenzen.** Conflict tussen eigenaren en/of verantwoordelijken.
6. **Registeronderdelen vereisen verschillende opslag- en/of integratietechniek.** Niet-verenigbare technische vereisten.

Indicatoren en symptomen (van sterk naar zwakker) voor een te krappe bounded context:

1. **Registeroverstijgende consistentiewaarborgen vereist.** Saga's zijn nodig om transacties gedistribueerd af te handelen.
2. **Model beschrijft domein zonder zelfstandig bestaansrecht.** Gegevens in een register hebben ‘los’ geen enkel bestaansrecht en zijn niet te generaliseren voor gebruik in andere domeinen.

3.6 Uitvoeringsproces op hoofdlijnen

Het uitvoeringsproces binnen een domein volgt een herkenbaar patroon. Door dit patroon te begrijpen, wordt duidelijk waar het register precies bijdraagt en waar de grenzen liggen.

// **TODO** Vereenvoudigd plaatje van een administratief uitvoeringsproces (process flow plaatje?): Signaal -> Taak -> Gevolg -> Presentatie

Stap 1: Signaal ontvangen Een proces start altijd met een signaal - een melding dat er iets is gebeurd wat aandacht vraagt. Dit signaal kan van buiten het domein komen (bijvoorbeeld een melding van een burger) of van binnen (bijvoorbeeld een (automatische) controle die iets detecteert).

Stap 2: Taken creëren Het proces beoordeelt het signaal en bepaalt welke acties nodig zijn. Deze acties worden vastgelegd als taken. Een signaal kan leiden tot één taak, meerdere taken, of soms geen taken als geen actie nodig is.

Stap 3: Taken uitvoeren Elke taak wordt uitgevoerd volgens de regels van het domein. Dit kan handmatige verificatie inhouden, automatische berekeningen, of andere domeinspecifieke activiteiten.

Stap 4: Gevolgen vastleggen Het uitvoeren van taken levert gevolgen op - dit zijn de definitieve veranderingen die het domein accepteert als geldig. Deze gevolgen vormen de waarheid voor dit domein. Ze beschrijven wat er is gebeurd: een persoon is verhuisd, een huwelijk is voltrokken, een kind is geboren, een bedrijf is ingeschreven, een eigendom is overgedragen.

Stap 5: Presentatie opmaken De gevolgen worden niet rechtstreeks gedeeld, maar gebruikt om projecties op te maken die aansluiten bij specifieke informatiebehoeften. Een verhuizing kan bijvoorbeeld leiden tot verschillende presentaties: een uittreksel voor de gemeente, een adreswijziging voor de belastingdienst, of een notificatie voor de zorgverzekeraar. Elke presentatie[^1] toont alleen de informatie die relevant is voor de ontvangende partij.

Deze cyclus vormt de kern van elk register-ondersteund proces. Het register bewaart de gevolgen en stelt ze beschikbaar voor raadpleging.

 Gedetailleerd uitvoeringsproces dat het register ondersteunt *Figuur 2: Gedetailleerd uitvoeringsproces dat het register ondersteunt*

3.6.1 Het domeinmodel als kern

Het **domeinmodel** definieert wat binnen het domein mogelijk is. Het beschrijft:

- **Taakdefinities**: Welke acties mogelijk zijn binnen het domein
- **Taakverwerkingsregels**: Hoe taken worden uitgevoerd en welke gevolgen ze hebben

- **Presentatiedefinities**: Welke informatie in welke vorm beschikbaar gesteld wordt
- **Presentatieverwerkingsregels**: Hoe gevolgen worden omgezet naar specifieke informatiebehoeften

Signaalverwerking staat buiten het domeinmodel. Allerlei signalen kunnen op het proces afkomen. De signaalverwerking interpreteert deze naar het domeinmodel toe: welke van de beschikbare taken moet worden uitgevoerd?

Het domeinmodel vormt de kern van wat het register ondersteunt - de mogelijke acties en hun gevolgen.

3.6.2 Signaal ontvangen en interpreteren

- Een gebeurtenis binnen of buiten het domein leidt tot een *Signaal*
- *Signaalverwerking* beoordeelt of dit signaal relevant is voor het domein
- Relevante signalen worden vertaald naar acties die het domein kan uitvoeren

3.6.3 Taken definiëren en uitvoeren

- Uit het signaal ontstaan één of meerdere *Taken* die binnen het domein uitgevoerd kunnen worden
- *Taakuitvoering* voert deze taken uit volgens de regels van het *Domeinmodel*
- Taken volgen *Taakdefinities* en *Taakverwerkingsregels* uit het domeinmodel

3.6.4 Gevolgen vastleggen

- Uitgevoerde taken leiden tot *Gevolgen* - de daadwerkelijke veranderingen in het domein
- Deze gevolgen vormen de nieuwe waarheid voor dit domein

3.6.5 Informatie beschikbaar stellen

- *Presentatieopmaking* creëert verschillende *Presentaties* op basis van de gevolgen
- Elke presentatie is afgestemd op specifieke informatiebehoeften volgens *Presentatiedefinities* en *Effectverwerkingsregels* uit het domeinmodel
- Presentaties kunnen als nieuwe signalen dienen voor vervolgprocessen in eigen of andere domeinen

3.7 Applicatieve ondersteuning met het register

Het uitvoeringsproces krijgt concrete vorm in een applicatiearchitectuur. Het register ondersteunt dit proces door de juiste componenten en datastromen in te richten. Ultiem is een (business) domeinmodel gelijk aan een register, één *bounded context* met eigen regels en begrippen, zoals bedoeld in [*domain-driven-design*]. Veel vaker is het zo dat er meerdere bounded contexten te onderscheiden zijn in één business domeinmodel[^2]. Dit heeft te maken met verantwoordelijkheden en technologie *stacks* en hebben vaak ook historische redenen; de verzameling van applicaties die door de tijd heen ontwikkeld zijn. Toch tekenen we hier de eenvoudige en ultieme situatie als uitgangspunt.

 Applicatieve ondersteuning en architectuurelementen waaruit het register bestaat *Figuur 3: Applicatieve ondersteuning en architectuurelementen waaruit het register bestaat*

3.7.1 Van proces naar applicatie

Het abstracte proces wordt concreet door deze architectuurcomponenten:

3.7.2 Notificatie → Command (Signaalverwerking)

- **Notificatieprocessor** ontvangt *notificaties* van binnen en buiten het domein
- Beoordeelt relevantie voor het eigen domein door raadpleging van projecties
- Vertaalt relevante notificaties naar *commands* volgens domeinconventies

3.7.3 Command → Effecten (Taakuitvoering)

- **Commandprocessor** ontvangt commands en bepaalt welke *effecten* ontstaan
- Voert domeinregels uit en valideert tegen bestaande gegevens via projecties
- Effecten beschrijven de daadwerkelijke veranderingen binnen het domein

3.7.4 Effecten → Projecties (Presentatieopmaking)

- **Effectprocessor** maakt *projecties* op basis van effecten
- Projecties bevatten alleen informatie uit het eigen domein ('data bij de bron')
- Elke projectie is afgestemd op specifieke (groepen van)[^1] informatiebehoeften

3.7.5 Register = Commands + Effecten + Projecties

Het register bestaat uit:

- **Commands**: De opdrachten aan het register, aangeboden met de intentie daarin een verandering aan te brengen
- **Effecten**: De waarheid van het domein - wat er werkelijk is gebeurd
- **Projecties**: Verschillende views op deze waarheid voor verschillende gebruikers

Syntheses[^3] (views die meerdere domeinen combineren) vallen buiten het register en verdienen eigen architectuur.

[^1]: Dit is conceptueel juist, maar zeker geen dagelijkse praktijk. In de informatiebehoeften van verschillende afnemers zijn vaak overeenkomstigheden te vinden, zodat groepering voor de hand. Richtlijnen en overwegingen gaan nog (elders) beschreven worden.

[^2]: No, Your Domains and Bounded Contexts Don't Map 1 on 1 - Mathias Verraes.

[^3]: Naar Chronolexografie, een conceptueel model en specifieke architectuur voor het digitaal vastleggen en bijhouden van de rechtstoestand. De *projecties* in bovenstaand artikel zijn te relateren aan de *reductie* uit Chronolexografie. Met *synthese* wordt in beide artikelen geduid op analyse en *view* op meer dan alleen projecties en reducties uit registers. Het betreft vaak uitgebreidere verwerkingen, transformaties en combinatie met meerdere projecties uit verschillende registers.

4. Conformiteit

Naast onderdelen die als niet normatief gemarkeerd zijn, zijn ook alle diagrammen, voorbeelden, en noten in dit document niet normatief. Verder is alles in dit document normatief.

A. Index

A.1 Begrippen gedefinieerd door deze specificatie

A.2 Begrippen gedefinieerd door verwijzing

B. Referenties

B.1 Normatieve referenties

[domain-driven-design]

Reference not found.

[domain-driven-design#the-blue-book]

Reference not found.

↥