Emerging Trends

Domain-driven design



|  |  |
| --- | --- |
| **Date** | 17-05-2022 |
| **Version** | 0.4 |
| **Student Name** | Lars van den Brandt |
| **Student Number** | 434565 |
| **Teacher Fontys** | Tom Meulensteen & Marcel Boelaars |

#### Version

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Changes** | **Status document** |
| 0.1 | 20-04-2022 | Lars van den Brandt | Eerste versie | Niet af |
| 0.2 | 02-05-2022 | Lars van den Brandt | Eerste onderzoek | Niet af |
| 0.3 | 11-05-2022 | Lars van den Brandt | Verder met deelvragen | Niet af |
| 0.4 | 18-05-2022 | Lars van den Brandt | Afmaken deelvragen | Wachten voor feedback |

#### Distribution

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Name** |
| 0.4 | 19-05-2022 | Canvas |

# Inhoudsopgave

[Version 2](#_Toc103681481)

[Distribution 2](#_Toc103681482)

[Inhoudsopgave 3](#_Toc103681483)

[Introductie 4](#_Toc103681484)

[Probleem/kans 4](#_Toc103681485)

[Research methods 4](#_Toc103681486)

[Onderzoeksvragen 5](#_Toc103681487)

[Hoofdvraag 5](#_Toc103681488)

[Waarom hebben we Domain-Driven Design nodig, bij het ontwikkelen van software? 5](#_Toc103681489)

[Deelvragen 6](#_Toc103681490)

[Wat is Domain-Driven Design? 6](#_Toc103681491)

[Welke andere software ontwikkelings technieken zijn er? 7](#_Toc103681492)

[Wat zijn de voordelen van Domain-Driven Design?? 9](#_Toc103681493)

[wat is het nadeel van Domain-Driven Design? 10](#_Toc103681494)

[Bronnen 11](#_Toc103681495)

# Introductie

Softwareontwikkeling is vrij nieuw in de wereld, maar het is een onderwerp dat zich steeds verder ontwikkelt en uitbreidt. Er worden verschillende methoden gebruikt bij het maken van software. Domain-Driven Design (DDD) is een aanpak die wordt gebruikt bij softwareontwikkeling. Het legt de focus het probleem, in plaats van op de oplossing.

Dit document bevat het onderzoek naar Domain-Driven-Design. Het onderzoek zal worden uitgevoerd door het beantwoorden van één hoofdvraag, ondersteund door verschillende deelvragen.

# Probleem/kans

Domain-Driven-Design is een onderwerp waar ik zelf nog niet veel kennis over heb. Ik zie dit onderzoek als een kans om meer kennis op te doen over dit onderwerp. Door het definiëren en het onderzoeken van de hoofd- en deelvragen zal ik deze kans kunnen benutten.

# Research methods

Het onderzoek wordt uitgevoerd aan de hand van het DOT Framework. Dit Framework helpt bij het verbeteren van de kwaliteit van het onderzoek. Het DOT Framework is een toolset, die gebruik maakt van verschillende methoden die vaak worden gebruikt binnen ICT. De toolset legt de focus op onderzoek van een product of methode.   
Bij het gebruik van dit framework, wordt geen nieuwe kennis ontwikkeld. Dit wordt wel gedaan in bijvoorbeeld een wetenschappelijk onderzoek.

Alle vragen zullen worden beantwoord aan de hand van het [DOT Framework](https://ictresearchmethods.nl/The_DOT_Framework).

De methoden die ik heb gebruik tijdens dit onderzoek:[[1]](#footnote-1)

* [*Available product analysis*](https://ictresearchmethods.nl/Available_product_analysis)

Door te kijken naar andere onderzoeken, heb ik deelvragen beantwoord.

* [*Gap analysis*](https://ictresearchmethods.nl/Gap_analyisis)

Door Domain-Driven Design te vergelijken met andere onwikkelingstechnieken, heb ik meer geleerd over de verschillen en de gewenste situaties.

* [*SWOT analysis*](https://ictresearchmethods.nl/SWOT_analysis)

Door zwaktes, sterktes, kansen en dreigingen te onderzoeken, ben ik meer te weten gekomen over Domain-Driven Design

* [*Decomposition*](https://ictresearchmethods.nl/Decomposition)

Door het grote onderwerp in kleinere stukjes te splitsen, is het makkelijker om een onderzoek uit te voeren.

* [*Literature study*](https://ictresearchmethods.nl/Literature_study)

Door het op en onderzoeken van belangrijke woorden, ben ik op kennis gekomen die ik in mijn onderzoek heb gebruikt.

* [*Best good and bad practices*](https://ictresearchmethods.nl/Best_good_and_bad_practices)

Door het praten met andere studenten, die ook Domain-Driven Design hebben gekozen, kunnen we kennis uitwisselen over het onderwerp. Dit zal ons beide helpen in het onderzoek.

# Onderzoeksvragen

### Hoofdvraag

#### Waarom hebben we Domain-Driven Design nodig, bij het ontwikkelen van software?

Om deze hoofdvraag te beantwoorden heb ik de vraag opgedeeld in deelvragen. Op deze manier is het mogelijk om snel een duidelijk beeld te krijgen.   
Als eerste moet er gezegd worden dat de voorkeur van een software ontwikkelingstechniek voor een project per persoon kan verschillen. Zo kan het zijn dat de ene persoon vrij en flexibel wilt werken, terwijl de ander liever via een boekje werkt.   
Er is hier dus geen goed of slecht antwoord.

Wel ben ik er door dit onderzoek achter gekomen, dat DDD (Domain-Driven Design) een techniek is dat binnen het ontwikkelen van software een aantal grote voordelen kan hebben.   
Om kort samen te vatten brengt DDD onder andere de voordelen met zich mee:

1. *Verbeterde communicatie*
2. *Afgestemd op de bedrijfsdoelen*
3. *Flexibiliteit*
4. *Gebruikersgericht*
5. *Efficiëntie*

**Conclusie**

Waarom hebben we Domain-Driven Design nodig, bij het ontwikkelen van software? DDD stemt eisen van het eindproduct af op de belangen van het bedrijf. De techniek helpt bij het beslissen van elementen waarop gefocust moet worden. Dit gebeurd meestal door een domein te bepalen. Op deze manier zal het eindproduct naar wens zijn voor de uiteindelijke gebruiker, en niet naar wens van de ontwikkelingsteam.

Het ontwikkelen door middel van DDD kan moeilijk zijn, maar na het toepassen van deze methode wordt het ontwikkelen van software er makkelijker op. Het zet een eindproduct neer waar iedereen tevreden mee is. Ook als DDD alleen op het begin van het ontwikkelingsproject wordt toegepast, brengt het de voordelen met zich mee dat er keuzes zijn gemaakt die zijn gebaseerd op kennis en overleg, in plaats van een gok of eigenbelang.

### Deelvragen

#### Wat is Domain-Driven Design?

*Toegepast uit DOT Framework:* [*Available product analysis*](https://ictresearchmethods.nl/Available_product_analysis)*,* [*Literature study*](https://ictresearchmethods.nl/Literature_study)

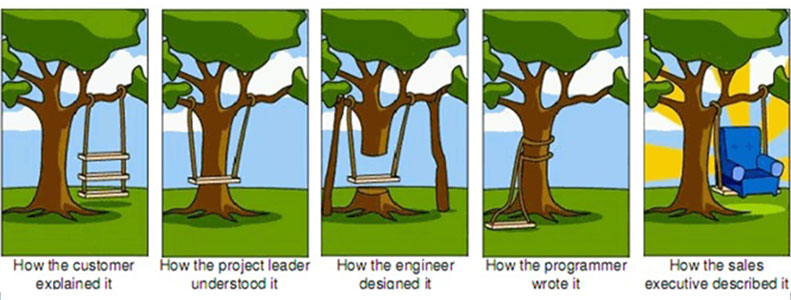
Domain-Driven-Design is een software ontwikkeling aanpak. Het is een aanpak waarbij er van te voren meer wordt gekeken naar het probleem, en nog niet zo zeer naar de oplossing.   
Bij deze aanpak steken alle ontwikkelaren de koppen bij elkaar en werken het probleem goed uit.

Deze manier van ontwikkelen is ontstaan omdat programmeurs vaak erg gefocust zijn op de techniek. De klant hierin tegen, heeft weinig tot geen interesse in de techniek, maar meer aan een makkelijke en correcte oplossing voor het probleem. Zo helpen de nieuwste technieken niet altijd bij het oplossen van het probleem.[[2]](#footnote-2)

In het begin van het project wordt de vraag gesteld: “Wat is het probleem van de klant en hoe kan ik dat zo simpel mogelijk oplossen?” Hierbij zijn mensen nodig die zich kunnen verplaatsen in de techniek die eventueel gebruikt gaat worden, maar ook mensen die makkelijk met de klant kunnen communiceren. [[3]](#footnote-3)

Daarom is het belangrijk dat er ‘dezelfde taal’ wordt gesproken. Hiermee wordt niet het verschil tussen Engels en Nederlands bedoeld, maar verschillende woorden die vaker in het project gebruikt kunnen worden. Dit concept heet ‘Ubiquitous Language’, of ‘Alomtegenwoordige taal’.[[4]](#footnote-4)

Het uiteindelijke doel van Domain-Driven-Design, is het voorkomen van misverstanden gedurende het project. Zo bestaat het zogenoemde ‘Swingtree Project’ als voorbeeld voor dit probleem. Door het niet bepalen van bepaalde termen, en het niet uitwerken van het probleem van de klant, kan een compleet andere oplossing voor het probleem in het hoofd worden bedacht. [[5]](#footnote-5)



#### Welke andere software ontwikkelings technieken zijn er?

*Toegepast uit DOT Framework:* [*Available product analysis*](https://ictresearchmethods.nl/Available_product_analysis)*,* [*Gap analysis*](https://ictresearchmethods.nl/Gap_analyisis)*,* [*Literature study*](https://ictresearchmethods.nl/Literature_study)

Er zijn door de jaren heen veel verschillende technieken bedacht binnen het ontwikkelen van software, met allemaal hetzelfde doel: de klant tevreden houden en een goed product neerzetten.

Naast Domain-Driven Design, waar dit onderzoek over gaat, zijn er twee technieken die ik heb onderzocht.

**Agile[[6]](#footnote-6)**

Agile, wat in het Nederlands wendbaar betekend, is zoals de naam al zegt een erg flexibele ontwikkeling techniek. Het is een techniek die vooral de laatste jaren erg populair is, aangezien de mensheid een stuk flexibeler wordt met alle middelen die we tot onze beschikking hebben. Zo hoeft je bij agile niet om 09:00 op het kantoor te starten, maar kan er vanuit thuis worden gewerkt.  
Het is niet meer aan de projectleider om over alle teammembers toezicht te houden, maar de projectleider is meer betrokken bij het project. Als teammembers vragen hebben, is het aan de projectleider om dit duidelijk te maken.

Agile werken komt erop neer dat we bereid zijn onze manier van werken te veranderen. Dit geld dus niet alleen om onze werktijden, onze fysieke werkplek, de technologie die we gebruiken, de rollen binnen een project, de manier van samenwerken, of de manier waarop ons werk wordt gedaan. Door flexibiliteit aan te moedigen, kunnen we ons aanpassen aan de steeds veranderende wereld om ons heen, zo kunnen we de prestaties en productiviteit verbeteren.

Binnen Agile werken zijn er verschillende methodes die kunnen worden toegepast:

1. **Scrum[[7]](#footnote-7)**

Scrum is een methode die vaak wordt toegepast binnen Agile werken, deze methode gebruiken wij zelf op school ook vaak. Scrum is een flexibele manier van werken, met als doel het opleveren van een software product. Binnen scrum binden korte sprints plaats, deze duren vaak twee tot drie weken, hierbij wordt met het team besproken wat er is gedaan in de afgelopen sprint, en wat er de komende sprint moet gebeurden. Op deze manier is iedereen op de hoogte van elkaars taken.

1. **Kanban[[8]](#footnote-8)**

Kanban s een agile methode die zich vooral focust op het visualiseren van het werk, het beperken van het daadwerkelijke werk, en het maximaliseren van efficiëntie. Zo richten de teams zich op het verminderen van de tijd van een project. Dit wordt vaak gedaan door een Kanban-bord.   
Een Kanban-bord lijkt vrij veel op een scrumboard, het verschil is dat er een maximaal aantal kaartjes kan zitten binnen een kolom, dit is om de efficiëntie te verhogen. Ook maakt een Kanban-bord gebruik van leveringspunten, dit staat vast aan de maximale aantal tijd dat tussen de afronding van kaartjes moet zitten, ook dit legt de focus op het verhogen van efficiëntie.

1. **SAFe[[9]](#footnote-9)**

SAFe staat voor ‘Scaled Agile Framework’, en combineert het beste van zowel scrum als Kanban. Zo maakt SAFe gebruik van de sprints uit scrum, maar ook van het projectboard van Kanban. SAFe legt de focus op drie punten. Alignment, het zorgt ervoor dat het team op een lijn zit. Build-in quality, er moet gedurende het gehele project kwaliteit worden geleverd, niet alleen op het einde van het project. Transparancy, er worden alleen maar over feiten gesproken, en alle teammembers zijn hiervan op de hoogte.

**Waterval[[10]](#footnote-10)**

De Watervalmethode volgt een chronologisch proces, dit houd in dat gedurende het project er gebruik wordt gemaakt van een standaard tijdsvolgorde, zo ziet de workflow (afbeelding) eruit als een waterval. Ook werkt deze methode volgens data dat in het begin van het project is bepaald. Bij deze methode is het voor de verschillende teams binnen een project niet nodig om voortdurend met elkaar te communiceren, en kunnen ze dus alleen werken. De communicatie is pas nodig op het moment dat er integraties moeten plaatsvinden binnen een project.   
Teammembers werken vaak individueel en hoeven, in tegenstelling tot de Agile methode, niet met elkaar te communiceren om de zoveel weken om rapport uit te brengen. Dit wordt vaak pas gedaan op het moment dat er integratie moet plaatsvinden.

Omdat de eisen voor een project vanaf het begin duidelijk zijn, weet elke medewerker wat en wanneer iets moet worden gedaan. Zo kan de tijd efficiënt worden ingepland gedurende het hele project. Daarnaast zijn er nog andere voordelen van de watervalmethode:

1. **Fouten opvangen**

Omdat de eisen voor het project in het begin worden vastgelegd, kunnen fouten al snel worden opgevangen. Hierdoor kunnen fouten in de implementatiefase worden voorkomen.

1. **Kosten berekenen**

Omdat de eisen voor het project in het begin worden vastgelegd, kan er een nauwkeurige kostenberekening worden gemaakt.

1. **Snel op de hoogte brengen**

Als er tijdens het project nieuwe ontwikkelaren bijkomen, kunnen deze makkelijk op de hoogte worden gebracht. Dit omdat alles van te voren is vastgesteld, en dus in een document staat.

1. **Geen nieuwe toevoegingen**

Tijdens het project kunnen klanten niet steeds nieuwe functies toevoegen aan het product. Dit omdat de eisen van te voren zijn vastgelegd, en met de klant zijn besproken.

#### Wat zijn de voordelen van Domain-Driven Design??

*Toegepast uit DOT Framework:* [*Available product analysis*](https://ictresearchmethods.nl/Available_product_analysis)*,* [*SWOT analysis*](https://ictresearchmethods.nl/SWOT_analysis)*,* [*Literature study*](https://ictresearchmethods.nl/Literature_study)

Domain-Driven Design is goed voor verschillende onderwerpen, het uiteindelijke doel is dat het helpt bij een oplossing voor een groot probleem, waarbij er gezamenlijk een zo simpel en efficiënt mogelijke manier kan worden bedacht.   
DDD zorgt voor vijf grote voordelen binnen een project.[[11]](#footnote-11)[[12]](#footnote-12)

1. **Verbeterde communicatie**

Zoals eerder benoemd wordt er binnen DDD gebruik gemaakt van Ubiquitous Language, dit zorgt voor gemakkelijkere en eenvoudigere communicatie tussen technische en niet-technische leden van het ontwikkelingsteam.

1. **Afgestemd op de bedrijfsdoelen**

Door de goede communicatie, en het vooraf vaststellen van het probleem en de oplossing, is de kans groter dat het uiteindelijke resultaat beter afgestemd is op de bedrijfsdoelen van zowel de klant als de ontwikkelaren zelf.

1. **Flexibiliteit**

Door het vooraf bepalen van termen en behandelde onderwerpen, is het ontwikkelen van het product erg flexibel. Zo kan eventueel later in het project het product nog worden aangepast als hiervoor behoefte is, terwijl er minder grote gevolgen zullen zijn.

1. **Gebruikersgericht**

Omdat binnen deze techniek de focus ligt op het domein, is dit hier belangrijker dan de gebruikers- ervaring en interface. Door DDD als ontwikkeling aanpak te gebruiken, wordt er een product ontworpen dat zich aan de behoefte van de gebruikers zal voldoen. Dit zijn wel de gebruikers die ook binnen dit domein vallen, en gebruik zullen maken van het desbetreffende product.

1. **Efficiëntie**

Het uiteindelijke product zal voldoen aan de eisen die de opdrachtgever in zijn gedachten had, omdat dit van te voren samen is besproken en bepaald. Het product zal het werk doen dat het moet doen, in plaats van het werk dat de ontwikkelaar denkt dat het product moet doen. Zo zal het product minder functies bevatten die de gebruikers uiteindelijk niet, of minder gebruiken.

DDD is een effectieve manier om duidelijke code te schrijven, die het uiteindelijke probleem oplossen. De klant/stakeholder is vaak niet geïnteresseerd in de code, aangezien ze hoogstwaarschijnlijk niet weten wat het allemaal precies betekend. De klant is daarentegen wel geïnteresseerd in een werkend product. De communicatie met de klant, zonder teveel over de technische kant te praten, is het grootste voordeel van DDD. [[13]](#footnote-13)

#### wat is het nadeel van Domain-Driven Design?

*Toegepast uit DOT Framework:* [*Available product analysis*](https://ictresearchmethods.nl/Available_product_analysis)*,* [*SWOT analysis*](https://ictresearchmethods.nl/SWOT_analysis)*,* [*Literature study*](https://ictresearchmethods.nl/Literature_study)

Naast de vele voordelen die Domain-Driven Design met zich meebrengt, brengt het ook een aantal nadelen met zich mee. [[14]](#footnote-14) [[15]](#footnote-15)

1. **Domain expert**

Er moet gedurende de hele ontwikkelingsfase een expert aanwezig zijn, die zich heeft verdiept in het doel en de eisen van het eindproduct, om mee te vergaderen over beslissingen die genomen moeten worden tijdens de ontwikkeling van het product.

1. **Tijd**

Wegens de extra communicatie die moet plaatsvinden tussen alle partijen die meehelpen aan de ontwikkeling van het product, zal er extra tijd moeten worden vrijgemaakt. Dit brengt kosten met zich mee. Ook zal het product later beschikbaar zijn voor de klant.

1. **Kosten**

Zoals in punt een benoemd, moet er een expert aanwezig zijn gedurende de ontwikkeling van het product, deze expert moet worden betaald. Ook, zoals in punt drie benoemd, brengt de communicatie meer tijd met zich mee, de tijd dat extra aan het product wordt besteed door de ontwikkelaren zal ook betaald moeten worden.

1. **Flexibiliteit**

Aangezien DDD vanaf het begin van het ontwikkelingsproces moet worden toegepast, en er in de eerste fase al veel details worden uitgewerkt, gaat de ontwikkeling minder flexibel dan bijvoorbeeld bij een Agile aanpak.

1. <https://ictresearchmethods.nl/The_DOT_Framework> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Domain-driven_design> [↑](#footnote-ref-2)
3. <https://www.isaac.nl/nl/nieuws/waarom-je-domain-driven-design-nodig-hebt-ook-als-je-nu-denkt-van-niet> [↑](#footnote-ref-3)
4. <https://www.senet.nl/blog/wat-is-domain-driven-design-ddd/?cn-reloaded=1> [↑](#footnote-ref-4)
5. <https://www.bergler.nl/domain-driven-design-in-gewoon-nederlands-artikel-2-het-model/> [↑](#footnote-ref-5)
6. <https://learnenglish.britishcouncil.org/business-english/business-magazine/agile-working#:~:text=Basically%2C%20agile%20working%20is%20about,way%20our%20work%20is%20done> [↑](#footnote-ref-6)
7. <https://nl.wikipedia.org/wiki/Scrum_(projectmanagementmethode)> [↑](#footnote-ref-7)
8. <https://agilescrumgroup.nl/wat-is-kanban-methode/> [↑](#footnote-ref-8)
9. <https://scrumguide.nl/safe-het-scaled-agile-framework/> [↑](#footnote-ref-9)
10. <https://www.workfront.com/project-management/methodologies/waterfall#:~:text=The%20waterfall%20methodology%20is%20a,detailed%20documentation%2C%20and%20consecutive%20execution>. [↑](#footnote-ref-10)
11. <https://www.isaac.nl/nl/nieuws/waarom-je-domain-driven-design-nodig-hebt-ook-als-je-nu-denkt-van-niet> [↑](#footnote-ref-11)
12. <https://www.boldare.com/blog/what-is-domain-driven-design/#:~:text=For%20complex%20applications%2C%20adopting%20a,members%20of%20the%20development%20team> [↑](#footnote-ref-12)
13. <https://simpleprogrammer.com/importance-domain-driven-design/> [↑](#footnote-ref-13)
14. <https://codezup.com/what-is-domain-driven-design-ddd-pros-cons/> [↑](#footnote-ref-14)
15. <https://www.thetechplatform.com/post/pros-and-cons-of-domain-driven-design-working-of-domain-driven-design-ddd> [↑](#footnote-ref-15)