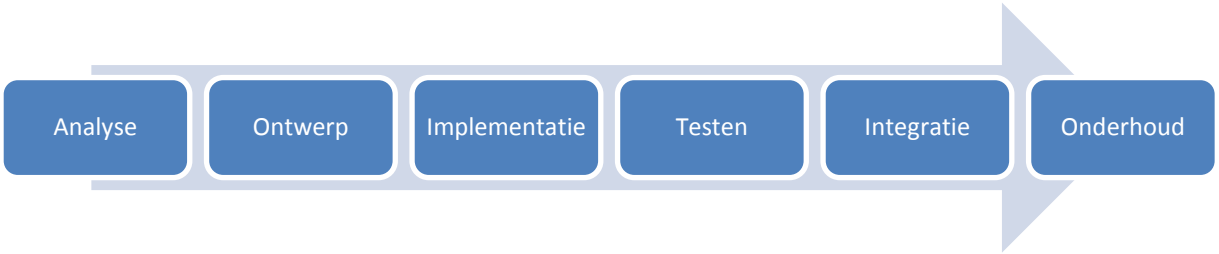
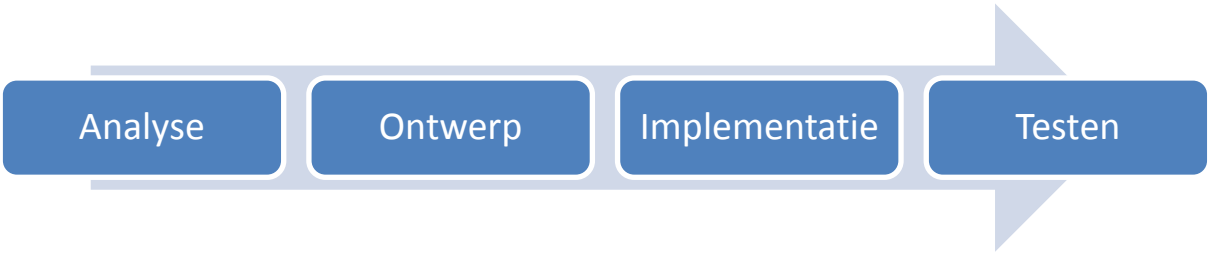
Analyse I

1 Software ontwikkelingsproces

1.1 Cyclus



1.2 Watervalmethode



1.2.1 Nadeel

Een aaneensluitend geheeld er kan niet terug gekeerd worden naar een vorige stap zonder gans opnieuw te beginnen

1.3 Iteratieve en incrementele ontwikkelmethode

Het geheel opsplitsen in kleinere delen en op elk deeltje de ontwerpcyclus toepassen.

iteratief:	voor elk deel opnieuw dezelfde cyclus toepassen
incrementeel:	na iedere iteratie komt men dichter tot het geheel
timeboxing:	vooraf beperken van de hoeveelheid tijd die men aan een bepaalde activiteit wil en mag besteden
deliverable:	document op het einde van iedere iteratie
milestone:	hetgeen wat gedaan moet zijn na een iteratie

1.4 Unified Proces

Gebaseerd op de iteratieve en incrementele methode.

Werkt eveneens in iteraties (1 iteratie = 2-6weken)

Tijdens de iteratie worden steeds dezelfde activiteiten uitgevoerd.

1.5 Agile ontwikkelingsmethode

1.5.1 Agile Manifesto

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

1.6 OOA/D

1.6.1 OOAnalyse binnen UP

- Opdrachtgever formuleert het probleem
- Analist noteert het verhaal en de eisen
- Analist vertaalt het verhaal naar use cases
- Ontwerper stelt aan de hand van de use cases scenario's en testen op
- Ontwerper stelt aan de van use cases het domeinmodel op
- Het Systeem sequentie diagram wordt opgesteld
- De operation Contracts worden opgesteld

1.6.2 OODesign binnen UP

- Verdere vertaling naar sequentie diagram
- Klassendiagram wordt opgesteld
- Testen worden voorbereid

1.7 Unified Modelling Language

- Modelleertaal om objectgeoriënteerde analyses te kunnen ontwerpen
- Geen methode, maar notatiewijze

2 Use Cases

2.1 Definitie

Use cases beschrijven de verwachtingen en voorwaarden van een gebruiker waaraan een systeem moet voldoen.

Wat is een use case?

- tekstuele beschrijving van een taak
- beschrijving van het gebruik van het systeem om een bepaald doel te verwezenlijken
- alle acties die het systeem moet uitvoeren om het doel te verwezenlijken worden duidelijk beschreven
- Begrijpelijk voor alle betrokken partijen
- vormt het contract tussen de belanghebbende partijen

2.2 Structuur

Use case = stappenplan voor een bepaalde taak van het systeem

Een use case geeft een overzicht van:

- precondities: taken die reeds uitgevoerd moeten zijn
- normaal verloop: stappen die nodig zijn om het doel te bereiken
- alternatief verloop: omschrijving van de stappen bij alternatieve wegen
- postcondities: toestand van het systeem na het uitvoeren van de use case

2.3 Use case diagram

Een use case diagram toont een overzicht van alle use cases, actors en hun onderlinge relatie voor het te ontwikkelen systeem in een grafische notatie.

2.3.1 Soorten relaties

- Includes
  - verplichte uitvoering van een deeltaak
- Extends
  - afwijking van het normale verloop
- Generalization
  - veralgemening

2.4 Use Case scenario's

Elk ander verloop van de use case levert een ander scenario op.

Alle scenario's samen (normaal + alternatief verloop) vormen de volledige beschrijving van de use case.

Uitwerken in het activity diagram

2.5 Nut van use cases

- Use case driven development
    - Specificatie van functionele eisen
    - Eenheid van planning
        - Identificeer de use cases
        - Geef prioriteiten aan de use cases
        - Schat de ontwikkeltijd
        - Lever incrementeel op (2-3 weken/iteratie)
    - Basis voor aanmaak van testen
        - Elke verschillend scenario moet getest worden
        - Als elke test die te maken heeft met de use case slaagt dan is er goede software.
    - Basis voor verdere analyse en ontwerp
    - Basis voor UI storyboard

3 Het domeinmodel

3.1 Nut van het domeinmodel

Het domeinmodel dient om op een eenvoudige manier te kunnen communiceren.

De kern van het probleem kan a.d.h.v. het domeinmodel met behulp van een betrekkelijk klein aantal begrippen beschreven worden.

Conceptuele domeinklassen vormen een belangrijk uitgangspunt en inspiratiepunt voor het ontwikkelen van softwareklassen.

3.2 Definitie

Het domeinmodel is de visuele representatie van concepten uit de werkelijkheid en hun onderlinge relatie.

3.3 UML klassendiagram

Het UML klassen diagram wordt gebruikt voor de voorstelling van het domeinmodel.

3.3.1 Conceptuele klassen

De relevante eigenschappen van deze klassen worden als attributen bijgehouden.

3.3.2 Associaties

Associaties zijn verbanden tussen instanties van klassen.

Het verband wordt vaak verduidelijkt met behulp van een associatienaam.

Als er bij associatienamen afgeweken wordt van de normale leesrichting dan moet er een pijltje geplaatst worden.

Reflexieve associatie: associatie die een verband uitdrukt tussen instanties van dezelfde klasse

3.3.3 Rollen en multipliciteiten

Via een rolnaam kan de rol die een instantie heeft in een associatie verduidelijkt worden.

Een rolnaam is optioneel.

Voor een rol kan de multipliciteit gespecificeerd worden.

Een reflexieve associatie kan nooit de verplicht zijn. Anders komt er een oneindig aantal objecten van die klasse.

3.3.4 Associatieklassen

Via een associatieklassen kunnen eigenschappen aan een associatie toegevoegd worden.

De levensduur van een instantie van de associatieklasse is dan ook afhankelijk van de associatie.

3.3.5 Aggregatie

Aggregatie is een deel/geheelrelatie tussen gelijken. (Niet-gevulde ruit)

De objecten kunnen zonder elkaar bestaan in het domeinmodel

3.3.6 Compositie

Via compositie kan uitgedrukt worden dat een instantie van een klasse opgebouwd is uit onderdelen die instanties zijn van een andere klasse. (volle ruit)

3.3.7 Afhankelijkheid

Klasse gebruikt een klasse.

Afhankelijkheden moeten vermeden worden , want anders moet er te veel aangepast worden als in de klasse, waarvan een andere klasse afhankelijk is, iets aangepast wordt.

3.4 Stappenplan om het domeinmodel te bekomen

1. Identificeer kandidaat-klassen
2. Selecteer conceptuele klassen
3. Identificeer associaties
4. Identificeer attributen

4 Systeem sequentiediagram en operation contracts

4.1 Sequentiediagram

4.1.1 Definitie

Een sequentiediagram is een UML-diagram dat toont hoe objecten met elkaar samenwerken in de loop van de tijd om een bepaalde functionaliteit te realiseren.

4.1.2 Onderdelen

Verticaal: de tijd (levenslijnen
Horizontaal: deelnemers en hun boodschappen

4.1.3 Deelnemers

instantienaam:Klasse

de naam van het object is optioneel en begint altijd met een kleine letter. De klassenaam begint altijd met een hoofdletter.

Ventje:	wordt gebruikt om de actor weer te geven
:Klasse	wordt gebruikt om een willekeurig object van Klasse aan te duiden.
instantie:klasse	wordt gebruikt om een object waarvan de naam gekend is weer te geven

4.1.4 Boodschappen

Communicatie tussen de objecten.

Pijl tussen de levenslijnen van de betrokken objecten met de nam van de boodschap en eventuele parameters.

Het sequentiediagram toont de volgorde van de boodschappen weer. Geen exacte timing.

Object die een boodschap ontvangt start een activiteit. De activiteit eindigt bij de terugkeer uit de methode.

4.2 Systeem sequentiediagram

4.2.1 Definitie

Een systeem sequentiediagram is een sequentiediagram dat alle interacties tussen de actoren en bet systeem van 1 use case scenario weergeeft.

4.2.2 Betekenis

Een use case is een beschrijving van het gedrag van een systeem vanuit het standpunt van de actor.

Een systeem sequentiediagram is een diagram waarop de interactie tussen de actor en het systeem worden vastgelegd. Het toont in welke volgorde systeemboodschappen verstuurd worden, in welke volgorde het systeem geactiveerd wordt en de gegevens die eventueel geretourneerd worden.

4.2.3 Nut

- Inzicht in de systeemeisen
- Toont de boodschappen/operaties (= systeemgedrag)
- Toont wat het systeem moet doen, niet hoe

- Startpunt voor het ontwerp.

4.2.4 Stappenplan voor het maken van een systeem sequentiediagram

1. Kies een use case
2. Teken de levenslijn voor elke actor en benoem deze
3. Teken een levenslijn voor het systeem dat als een blackbox wordt voorgesteld
4. Overloop de gekozen use case
    a. Voor alle acties die de actoren uitvoeren dient een boodschap getekend te worden
    b. Teken een activeringsblokje op de levenslijn van het systeem
    c. Als het systeem gegevens retourneert, moet een returnpijl getekend worden
    d. Stel indien nodig het Operation Contract op

Stap 4 verduidelijking:

- Is de actie van de actor?
    - Ja: Teken boodschap op systeem sequentiediagram
    - Nee: Doe niets
- Is het een reactie van het systeem?
    - Ja: Teken een returnpijl en benoem de gegevens
    - Nee: Doe niets

Opgepast: Meldingen zijn geen returnwaarden

4.3 Operation contract

Een operation contract wordt gebruikt omdat het systeem sequentiediagram niet alle informatie bevat.

4.3.1 Definitie

Een operation contract is:

- een onderdeel van de analyse.
- een document dat beschrijft wat een operatie moet bereiken
- een beschrijving van de veranderingen/gevolgen in het systeem ten gevolge van die operatie
- een declaratieve beschrijving van het effect van die operatie, het beschrijft niet hoe de veranderingen tot stand zijn gekomen.

4.3.2 Template van een operation contract

Operation	Naam van de operatie
Cross references	Opsomming van de use cases waarin de operatie gebruikt wordt.
Preconditions	
Postconditions	

4.3.3 Opstellen van een operation contract

1. identificeer de systeemoperaties uit het systeem sequentiediagram
2. Maak voor elke operatie die iets wijzigt in het systeem een operation contract
3. Gebruik bovenstaande template om het operation contract op te stellen

5 User Experience (UX)

5.1 Inleiding

Usability: effectiviteit, efficiëntie en voldoening waarmee de gebruikers het systeem, product of service gebruiken.

UX: de perceptie en de reacties van de gebruikers ten aanzien van het gebruik van het systeem, product of service.

Methode om usability en UX te integreren:
De gebruiker een centrale rol geven in het ontwikkelingsproces, zodat de ontwikkeling gebruikers gericht gebeurt.

5.2 Historiek

- Jaren 70
  - Automatisering en verbeteringen via software ontwikkeling
  - Ontwikkeling gericht op reduceren van tijd en aantal fouten
  - Focus voor de gebruikers op snelheid en efficiëntie
  - Gebruikers zijn vaak technici
  - Weinig nood aan usability
- Jaren 80
  - PC komt op de markt
  - Opkomst van productivity tools (spreadsheets, tekstverwerkers, grafische tools…)
  - Gebruikers zijn minder technisch
  - Belang van usability neemt toe
- Jaren 90
  - Windows 3.0 bereikt het grote publiek
  - Opkomst van internet
  - Gebruikers vragen bruikbare systemen
- Jaren 2000-2010
  - Computer = desktop PC
  - Brede waaier van technologische producten
  - Embedded technology
- Toekomst
  - Generation Z: nooit in een wereld zonder technologie
  - Hoge verwachtingen van de technology
  - Apps vs. m-sites, native vs. HTML5, fluid, responsive, tablets
  - Near Field communication,geolocation, augmented reality, voice, exoskeleton
  - Social web
  - E-commerce
  - Cloud
  - UX nog belangrijker

5.3 Focus

- Functies
  - Wat kan het allemaal?

- Performatie en betrouwbaarheid
    - Hoe snel, efficiënt en betrouwbaar werkt het?
- Design vision
    - Wat is de toegevoegde waarde?
    - In lijn met de missie en visie?
- Usability
    - Gemak waarmee een gebruiker het begrijpt en kan gebruiken
- Fun, Joy and Delight
    - De mate waarin de gebruiker het graag gebruikt

Functies en performantie en betrouwbaarheid hebben een onmiddellijk effect op de verkoop.

- Goedkoop
- Breed klantensegment
- Klanten zijn bereid om het product met de meeste functies te kopen

Usability en fun,joy and delight hebben pas na een tijdje effect

- Minder effect op de verkoop
- Klanttevredenheid
- Repeat business

Wanneer een product op 1 van de 5 factoren faalt heeft dit een onmiddellijk gevolg voor de verkoop en de tevredenheid.

Design visie is een oplossing voor een probleem uit de realiteit van de gebruiker. Het is gelinkt aan de missie en visie van de organisatie.

5.4 Gebruikerstaken en -context

Begrip verwerven in

- welke doelstellingen hebben gebruikers?
- wat doen ze om hun doel te bereiken?
- welke persoonlijke kenmerken hebben ze?
- wat is de invloed van voorkennis en ervaring?

5.4.1 Requirements

- wat een product moet doen
- welke eigenschap een product moet hebben

Functionele requirements:
zinvolle acties die met een product uitgevoerd moeten kunnen worden. (use cases)

Niet-functionele requirements:
eigenschappen of kwaliteiten die in een product vervat moeten zitten.

5.4.2 Observaties en interviews

Gebruikers observeren:

- ≠ vragen stellen in focus groups
- ≠ surveys en questionnaires
- ≠ gesprekken voeren met zgn. expertgebruikers
- = inzicht krijgen hoe gebruikers hun dagelijkse taken uitvoeren
- Actief luisteren
- 100% focus op het gesprek
- Uitleg van de gebruikers parafraseren
- Verwachtingen expliciteren
    - je eigen rol
    - het doel van het interview
    - vb. geen evaluatie van het functioneren van de gebruiker

Participerende observatie

- Fly-on-the-wall

Rollenspel (focus group)

- Alternatief voor de twee vorige
- minder efficiënt
- beter in onvoorspelbare situaties

5.5 Device Experience

- Hoe gebruik je het product meestal?
- Input, output & display

Beslis:

- Verschillende oplossingen voor elk toestel
- Interface design oplossingen uniek per toestel
- Primaire taken verschillen
- Lay-out en interactie moeten aangepast worden
- 1 oplossing voor alle toestellen → compromis
- een oplossing die zichzelf aanpast aan de verschillen tussen de toestellen en zelfs tussen de device experiences → Responsive design

5.6 Posture

- Hoe de toepassing zich aan de gebruiker presenteert
- Hoeveel aandacht een gebruiker aan de toepassing besteedt
- Hoe de toepassing reageert op de acties van de gebruiker

5.6.1 Lean back

Kenmerken:

- veel functies, hoge gebruiksfrequentie

- gebruikt gedurende lange periode, mentale ‘flow’
- voor brede gebruikersgroep

Aanbevelingen:

- maak gebruik van het **volledige schermoppervlak**
- gebruik een **minimalistische** visuele stijl
- geef veel **visuele feedback**
- voorzie **rijke inputmogelijkheden**

5.6.2 Lean forward

5.6.3 Quick bursts of activity in various locations throughout the day

Kenmerken:

- enkelvoudige functie, beperkte set van controls
- slechts tijdelijk actief

Aanbevelingen:

- maak de interface **eenvoudig en duidelijk**
- zorg voor **herkenbaarheid**
- wees **to-the-point**

5.7 Interface en stijl: Platform

- Platformstijlgidsen
  - bevatten algemene designprincipes
- bespreken in detail
  - menu's en toolbars
  - Vensters
  - Navigatie
  - Controls
- interactie via muis, keyboard, gestures, …
- visueel design
- heel wat informatie is reeds beschikbaar

5.8 Concept en navigatie

Personen hebben steeds een mentaal model
= voorstelling van hoe een product moet werken

- Gebaseerd op vorige ervaring
- Niet alle gebruikers hebben eenzelfde mentaal model
  - → gebruikerstaak en -context analyse
- Think aloud – Gebruikers geven inzicht in hun mentaal model
- Designers hebben ook een mentaal model

Interface communiceert het conceptuele model van het product
= het model dat door het product weergegeven wordt

Intuïtieve producten:

- Het conceptuele model matcht met de mentale modellen van de eindgebruikers

Innovatieve producten**:**

- Volledig nieuw conceptueel model dat (altijd/ volledig) afwijkt van het huidige mentale model

In plaats van het conceptuele model van het product aan te passen, pas het mentale model van de gebruikers

- Via training of ondersteuning in UI

5.8.1 Eerste ontwerpen

Expliciete, afzonderlijke fase in ontwikkeling, met doelstellingen en milestones

- Verschillende concepten bedenken
- Uitwerken in 'low fidelity' maquettes
- Testen bij eindgebruikers
- hoe interpreteren gebruikers het concept?

5.8.2 Navigatie

= schematische weergave van de opeenvolging van applicatie-onderdelen en de middelen om te navigeren

Doel:

- inzicht krijgen in de complexiteit van de gebruikersinterface
- zicht krijgen op de omvang van de applicatie
- communicatie met ontwikkelaars en opdrachtgevers

5.9 Disign Principles

Ken je gebruiker:

- Houd rekening met verschillende gebruikers
- Info uit de Gebruikerstaak- en -contextanalyse
- Usability = wegnemen van frictie
- Persuasion = aanzetten tot actie
    - Hiervoor moet je weten wat je gebruiker motiveert
- Motivatie & 'trigger'
    - moeten gelijktijdig aanwezig zijn
    - zijn essentieel om impact te hebben op gedrag

Beperk de mentale belasting:

- objecten die bij elkaar staan horen bij elkaar
- objecten die op mekaar gelijken horen bij elkaar
- lijnen suggereren een grens of een relatie

- Herkennen = op basis van een visuele trigger informatie uit het geheugen ophalen
- Herinneren = zonder visuele trigger informatie uit het geheugen ophalen

- Biedt visuele hints aan
    - Aan de hand van grootte, schaduw, vorm
    - Plaats controls logisch in de ruimte
    - Visuele hints gaan verloren

- Vertrouw niet op het geheugen
    - Gebruikers kunnen slechts beperkte hoeveelheid informatie in werkgeheugen hebben
    - - 4 +/- 2 items

- Presenteer informatie
    - die de gebruiker nodig heeft
    - op de juiste plaats
    - op het juiste moment
    - niet alles in 1 keer

- Pas het signaal bij een waarschuwing aan naar de context
- Maak het sterk wanneer het zelden gebeurt en belangrijk is om op te merken
- Gebruikers hebben een mentaal model over hoe vaak een situatie zich kan voordoen

- Veronderstel niet dat gebruikers alles zien
  →Uitkijken bij de refresh van het scherm
- Gebruikers realiseren zich niet dat ze naar een ander scherm aan het kijken zijn
  = “inattention blindness” of “change blindness”

- Bied shortcuts aan voor acties die heel vaak opnieuw uitgevoerd moeten worden
  →Eenvoudig vindbaar
- Voorzie default waardes en acties wanneer je weet wat de meest logische stap is voor de gebruiker →Enkel als het resultaat geen grote fouten introduceert

Geef de gebruiker de controle:

- Biedt alternatieven en ‘exits’ aan
    - cancel
    - (multiple) undo, history
    - proces onderbreken
        - zoek de juiste balans
        - afhankelijk van gebruikers
    - moedig exploratief leren

Design for error:

- Vermijd foutboodschappen
- Foutboodschappen
    - = gebruikers helpen in het verder uitvoeren van taak
    - ≠ bug-detectie
- Geef de gebruiker nooit de schuld en maak herstel zo eenvoudig mogelijk
- Wanneer gebruikers bepaalde acties repetitief uitvoeren, wordt dit een automatisme
    - Opgelet met foutboodschappen – ze worden irrelevant
- Pas het design van de foutboodschap aan
    - Bijvoorbeeld alle acties in 1 beweging laten uitvoeren
- Wanneer gebruikers bepaalde acties repetitief uitvoeren, wordt dit een automatisme
    - Opgelet met foutboodschappen – ze worden irrelevant
- Pas het design van de foutboodschap aan
    - Bijvoorbeeld alle acties in 1 beweging laten uitvoeren

Maak het consistent:

- Blijf trouw aan je concept en werkt het (verder) uit
- Streef naar standaardisering en consistentie
    - binnen 1 toepassing
    - over verschillende toepassingen
    - gebruikers verwachten consistent gedrag
- Nieuwe concepten vragen leertraject
- Voordelen voor gebruikers:
    - kortere leercurve
        - voorspelbaarheid
        - overdracht van kennis
    - meer gebruiksgemak
- Voordelen voor ontwikkelaars
    - minder tijd voor ontwerpbeslissingen
    - tijd besteden aan datgene wat hij best kan
- Stimulans voor de ontwikkeling van herbruikbare bibliotheken van interfacecomponenten
    - makkelijker voor onderhoud
    - makkelijker voor maquettering

5.10 Design Problemen

5.10.1 Tabs

- Om gerelateerde gegevens te tonen
    - Informatie op verschillende pagina's
- Verschillende views
- Verschillende documenten
- Exclusieve opties
- Horizontaal of verticaal

5.10.2 Uitgrijzen of verwijderen

5.11 Maquettes

- Controleren en demonstreren van analyseresultaten
- Ontwerp van de gebruikersinterface
- Beoordelen van de technische haalbaarheid van de gebruikersinterface
    - Specificatie van de gebruikersinterface
    - samen met de UI guidelines

5.11.1 Voordelen

- Relatief lage ontwikkelkost en snel te ontwikkelen
- Alternatieve ontwerpideeën toetsen alvorens te investeren in de uiteindelijke ontwikkeling
- Gebruikers en ontwikkelaars ideeën laten uitwisselen en hun relatie bevorderen
- Totale ontwikkelingstijd en nodige resources verminderen

5.11.2 Nadelen

- Beperkte functionaliteit
    - Beperkte interactie
- Look-and-feel kan afwijken van de uiteindelijke applicatie

5.11.3 Maquetteringstools

- Voordelen
    - snelle resultaten
    - korte leertijd voor het gebruik van de tool
    - ‘libraries’ met standaard user interface-componenten
- Nadelen
    - niet gebonden aan de mogelijkheden en beperkingen van de ontwikkelomgeving
    - niet recupereerbaar