Eisenanalyse

Robbert

[Jaar]

Inhoud

[2 Inleiding 4](#_Toc501445861)

[2.1 Opdrachtbeschrijving 4](#_Toc501445862)

[2.2 Soorten eisen 4](#_Toc501445863)

[2.3 Het usecasemodel 4](#_Toc501445864)

[2.3.1 Doelstellingen 5](#_Toc501445865)

[2.3.2 Onderdelen 5](#_Toc501445866)

[2.3.3 Stappenplan usecasemodellering 8](#_Toc501445867)

[2.3.4 Opdrachtbeschrijving 8](#_Toc501445868)

[3 Eisenanalyse – uitgebreid 9](#_Toc501445869)

[3.1 Eisen zoeken: vinden usecases 9](#_Toc501445870)

[3.2 Actors zoeken 9](#_Toc501445871)

[3.3 Use cases zoeken (actor based) 9](#_Toc501445872)

[3.4 Relaties tussen usecases 9](#_Toc501445873)

[3.4.1 Extend-relatie (met conditie) 9](#_Toc501445874)

[3.4.2 Include-relatie 10](#_Toc501445875)

[3.5 Sea-level niveau 11](#_Toc501445876)

[3.6 Kiss 11](#_Toc501445877)

[4 niet-functionele eisen 12](#_Toc501445878)

[4.1 Implementatie 12](#_Toc501445879)

[4.2 Externe interface 12](#_Toc501445880)

[4.3 Performantie 12](#_Toc501445881)

[4.3.1 Antwoordtijden 12](#_Toc501445882)

[4.3.2 Schaalbaarheid 12](#_Toc501445883)

[4.4 Kwaliteitseisen 12](#_Toc501445884)

[4.4.1 Software 12](#_Toc501445885)

[4.4.2 Hardware 12](#_Toc501445886)

[4.5 Checklist 13](#_Toc501445887)

[5 website vs web applicatie 14](#_Toc501445888)

[5.1 Doelen 14](#_Toc501445889)

[5.2 Analyse gebruikers en taken 14](#_Toc501445890)

[5.3 Ontwerp 14](#_Toc501445891)

[5.4 Functies 14](#_Toc501445892)

[5.5 Home page 14](#_Toc501445893)

[5.6 Help 15](#_Toc501445894)

[6 prototyping 16](#_Toc501445895)

[6.1 Definities 16](#_Toc501445896)

[6.2 Doel 16](#_Toc501445897)

[6.3 Voordelen 16](#_Toc501445898)

[6.4 Soorten 16](#_Toc501445899)

[6.5 Storyboard 16](#_Toc501445900)

[6.6 Hi-fi vs lo-fi 17](#_Toc501445901)

[6.7 Best practices hifi prototyping 17](#_Toc501445902)

[7 WRM 18](#_Toc501445903)

[7.1 Wat? 18](#_Toc501445904)

[7.2 Werkwijze 18](#_Toc501445905)

[8 datamodellering basis 19](#_Toc501445906)

[8.1 Soorten gegevens 19](#_Toc501445907)

[8.1.1 Samengestelde gegevens 19](#_Toc501445908)

[8.1.2 Procesgegevens 19](#_Toc501445909)

[8.1.3 Primary key 19](#_Toc501445910)

[8.1.4 Surrogate key 19](#_Toc501445911)

[8.1.5 Alternate key 19](#_Toc501445912)

[8.1.6 Foreign key 19](#_Toc501445913)

[8.2 Relaties tussen tabellen 19](#_Toc501445914)

[8.3 Datamodellering 20](#_Toc501445915)

[8.3.1 ERD 20](#_Toc501445916)

[8.3.2 Goed datamodel 21](#_Toc501445917)

[8.3.3 Wanneer? 21](#_Toc501445918)

[8.3.4 UML notatie 21](#_Toc501445919)

[9 synthesemethode 22](#_Toc501445920)

[9.1 Werkwijze 22](#_Toc501445921)

[9.2 Toewijzen 22](#_Toc501445922)

[9.2.1 Kernentiteit 22](#_Toc501445923)

[9.2.2 Primaire sleutel 22](#_Toc501445924)

[9.2.3 Alternate key 22](#_Toc501445925)

[9.2.4 Surrogate key 23](#_Toc501445926)

[9.2.5 Associatie-entiteit 23](#_Toc501445927)

[9.2.6 Foreign key 23](#_Toc501445928)

[9.2.7 Karakteristieke entiteit 23](#_Toc501445929)

[10 specs voor datamodellen 24](#_Toc501445930)

[10.1 Specificaties attributen 24](#_Toc501445931)

[10.1.1 NA/NNA 24](#_Toc501445932)

[10.1.2 Datatypes 24](#_Toc501445933)

[10.2 Specificaties FK 24](#_Toc501445934)

[10.2.1 Optionaliteit relatie 24](#_Toc501445935)

[10.2.2 DTR/DTC/DTN 24](#_Toc501445936)

[11 special topics 25](#_Toc501445937)

[11.1 Recursieve koppeling 25](#_Toc501445938)

[11.2 Tijdsaspect 25](#_Toc501445939)

[11.2.1 Enkel actuele waarde 25](#_Toc501445940)

[11.2.2 Ook waarden verleden 25](#_Toc501445941)

[11.3 Standaardwaarden en uitzonderingen 25](#_Toc501445942)

[11.3.1 Enkel standaardwaarde 25](#_Toc501445943)

[11.3.2 Enkel uitzondering 25](#_Toc501445944)

[11.3.3 Beide 25](#_Toc501445945)

[11.4 Vast aantal coderingen 26](#_Toc501445946)

[12 super- en subtypen 27](#_Toc501445947)

[12.1 Supertype 27](#_Toc501445948)

[12.1.1 Voordelen 27](#_Toc501445949)

[12.1.2 Nadelen 27](#_Toc501445950)

[12.2 Subtype 27](#_Toc501445951)

[12.2.1 Voordelen 27](#_Toc501445952)

[12.2.2 Nadelen 27](#_Toc501445953)

# Inleiding

## Opdrachtbeschrijving

* Achtergrond:
  + Wie wil nieuw systeem?
  + Hoe verloopt het op dit moment?
  + Wat loopt nu mis/minder efficiënt?
* Doelstellingen:
  + Wat zal men opleveren met welk doel?
  + Wat zal het opleveren voor de organisatie?
* Doelgroepen:
  + Wie zal baat hebben bij dit systeem?
  + Wie zal met systeem moeten/willen werken?

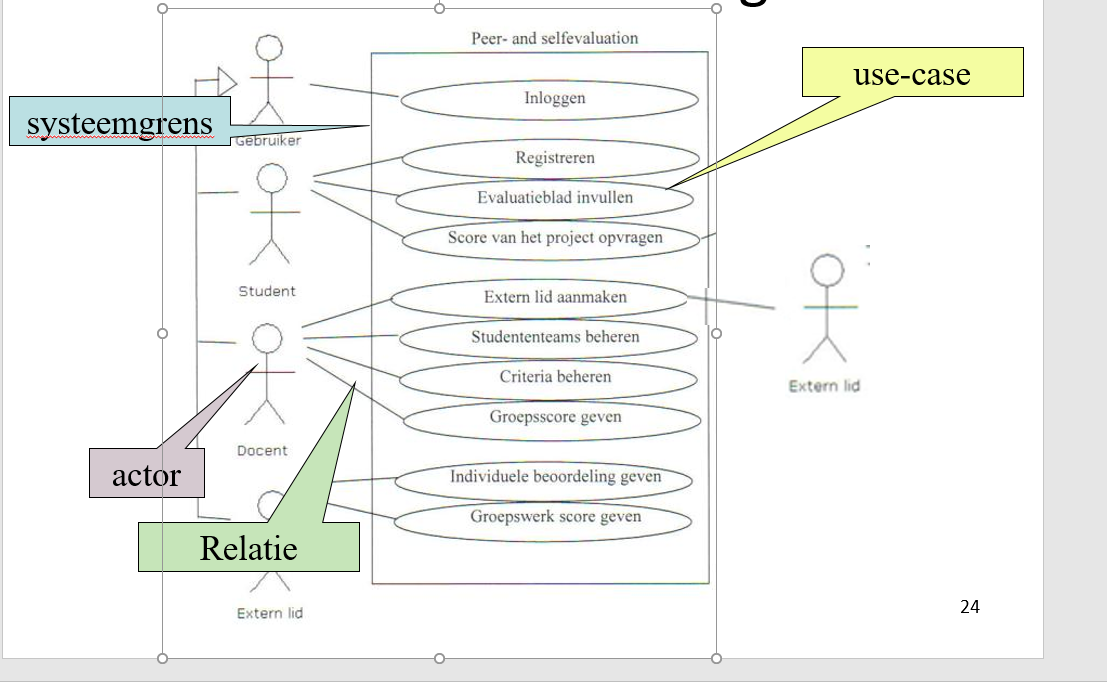
## Soorten eisen

* Functioneel:
  + Wat wil gebruiker dat systeem kan?
  + = functionaliteit en gegevens
* Niet-functioneel:
  + Randvoorwaarden
  + = kwaliteit en beperkingen

## Het usecasemodel

= hulpmiddel om functionele eisen overzichtelijk in kaart te brengen en te beschrijven

Bevat:

* Één usecasediagram (meerdere usecases)
* Meerdere usecasebeschrijvingen

Wat doet het?

* Geeft overzicht van de functionele eisen per type gebruiker
* Usecasebeschrijvingen geven details over hoe de functionele eis kan gerealiseerd worden (documentatie in tekstvorm)
* Systeem wordt beschouwd als “black box”, we zijn enkel geïnteresseerd in de interactie tussen gebruiker en systeem
* WAT systeem moet doen in reactie op gebruiker, NIET HOE

### Doelstellingen

* Vastleggen functionele eisen tijdens analyse en design
* Basis voor testen
* Communicatie-instrument
* Documentatie van systeem

### Onderdelen

* Systeemgrens
* Actoren
* Usecases
* Relaties
* Beschrijving

#### Systeemgrens

=scope

* Systeem = rechthoek met systeemnaam
* = start analyse

#### Actor

= entiteit die buiten het systeem staat

* Gebruikt systeem
* Heeft interactie met systeem
* Communiceert met systeem

=

* Rol die iets of iemand in context van systeem speelt
* Organisatie(eenheid)
* Systeem

2 soorten:

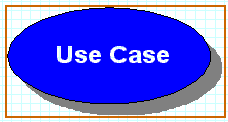
* Actieve (links):  
  initieert functionaliteit
* Passieve (rechts):  
  neemt deel na initialisatie

Actor zoeken:

* Wie gebruikt belangrijkste functies?
* Wie heeft functionaliteit nodig voor dagelijkse taken?
* Wie heeft belangstelling in resultaten/rapporteringen?
* Van wie heeft systeem info/gegevens nodig?

#### Usecase

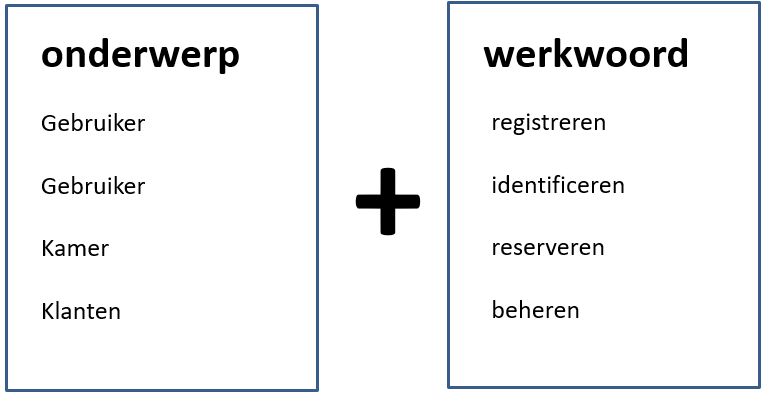
* Geval/situatie waarvoor gebruiker systeem wil gebruiken
* Doelstelling waarvoor >=1 actoren systeem willen gebruiken
* Opeenvolging van interacties tussen >=1 actoren en het systeem met welbepaald doel

Hoe bepalen?

* Welke funcites verwacht actor van systeem?
* Wat moet actor doen met gegevens?
* Welke input/outpuit moet actor geven/krijgen?
* Welke gebeurtenissen zijn van nut voor actor?
* Welke functies kunnen gebruikt worden om werk actor te vereenvoudigen?

2 soorten:

* Primair:  
  ondersteunt hoofdfunctionaliteit van systeem
* Secundair:  
  ondersteunt functionaliteit om hoofdfunctionaliteit mogelijk te maken



#### Usecasebeschrijving

* Functionaliteit:
  + Beschrijf functionaliteit in vorm van user story
  + Wie/wat kan
  + Als [rol]
  + kan ik [functionele eis/ usecase]
  + Vb: Als student, kan ik me inschrijven voor een lezing
* Normaal verloop:
  + Beschrijven successcenario
  + Afwisselend wat actor doet en wat systeem vervolgens doet
  + Vb: De student geeft zoektermen (onderwerp, spreker of datum) om een lezing te vinden. Systeem [a] geeft resultatenlijst. Student selecteert hieruit een lezing waarvoor hij zich wil inschrijven. De student geeft vervolgens zijn studentennummer in waarna het systeem de inschrijving registreert en de student een bevestiging stuurt.
* Alternatieven:
  + Optioneel
  + Beschrijven alternatieve scenario’s
  + 1 paragraaf per scenario
  + Vb: [a] Bij de lezingen die volzet zijn, geeft het systeem de student de mogelijkheid om zich op de wachtlijst te zetten. Het systeem stuurt dan een aangepaste bevestiging en registreert de student op de wachtlijst.
* Extra opmerkingen:
  + Optioneel
  + Extra opmerkingen die opdrachtgevers bij functionaliteit hebben vermeld en waar rekening mee gehouden moet worden bij uitwerken
  + Vb: Wanneer de resultatenlijst te lang is, verdeelt het system de lezingen over meerdere pagina’s waardoor de student kan bladeren.

Wat?

* Hoe systeem zich in bepaalde omgeving gedraagt, NIET interne structuur
* Logische gebeurtenissen, NIET gebruikersinterface
* Interacties actor <> systeem, NIET interacties actoren onderling of met andere systemen

### Stappenplan usecasemodellering

1. Identificeer grenzen systeem
2. Vindt actoren
3. Definieer usecases voor iedere actor
4. Beschrijf elke usecase
   1. Bepaal precondities
   2. Bepaal interacties
   3. Bekijk mogelijke alternatieven
   4. Beschrijf extra opmerkingen
5. Maak een proper usecasediagram

### Opdrachtbeschrijving

(voor het starten met eisenanalyse)

* Achtergrond:
  + Wie wil nieuw systeem?
  + Hoe verloopt proces nu?
  + Wat loopt nu mis?
* Doelstellingen:
  + Wat met welk doel?
  + Wat zal het opleveren voor organisatie?
* Doelgroepen:
  + Wie zal baat hebben bij systeem?
  + Wie zal met systeem moeten/willen werken?

# Eisenanalyse – uitgebreid

## Eisen zoeken: vinden usecases

* Analyseer de bedrijfsprocessen:
  + Welke functies moeten uitgevoerd worden?
  + Kan hier computersysteem gebruikt worden?
* Wie potentiële gebruikers en wat doen zij?
  + OK voor eenvoudige systemen
* Kan werk actor eenvoudiger/efficiënter?
* Welke input/output, van waar naar waar?
* Voornaamste problemen huidig systeem?

## Actors zoeken

* Wie gebruikt hoofdfunctionaliteit?
* Wie ondersteuning nodig van systeem voor dagelijkse taken?
* Wie/wat geïnteresseerd in resultaten?
* Interactie met andere systemen? Wie initieert contact?
* Wie zorgt basisgegevens “up-to-date” blijven?

## Use cases zoeken (actor based)

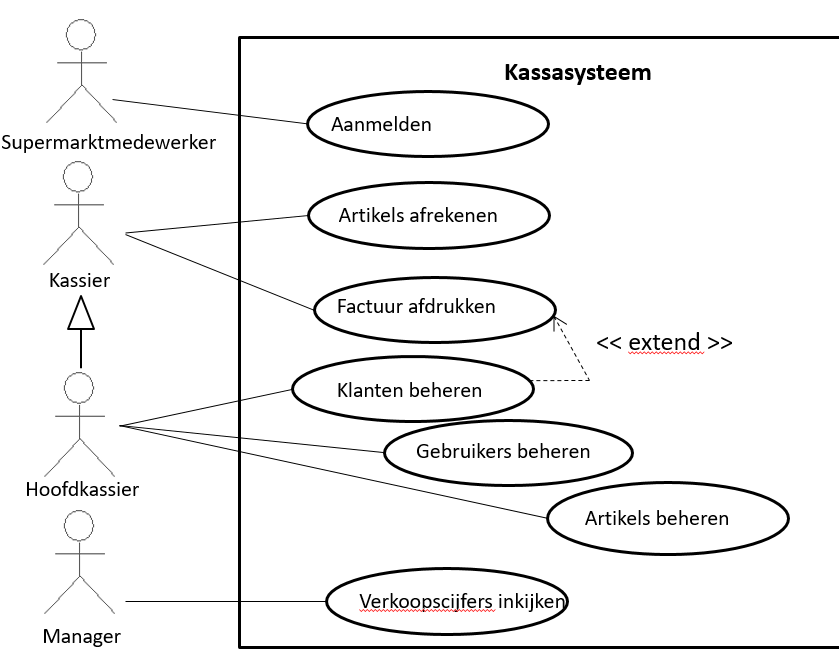
* Belangrijkste taken die systeem voor actor moet uitvoeren?
* Zal actor data aanmaken, opslaan, wijzigen, verwijderen of lezen?
* Moet actor systeem informeren over externe wijzigingen?
* Moet actor geïnformeerd worden over bepaalde gebeurtenissen?

## Relaties tussen usecases

### Extend-relatie (met conditie)

Geeft aan dat aan gedrag usecase extra gedrag wordt toegevoegd

* Wordt optioneel uitgevoerd
* Nieuw gedrag in nieuwe usecase

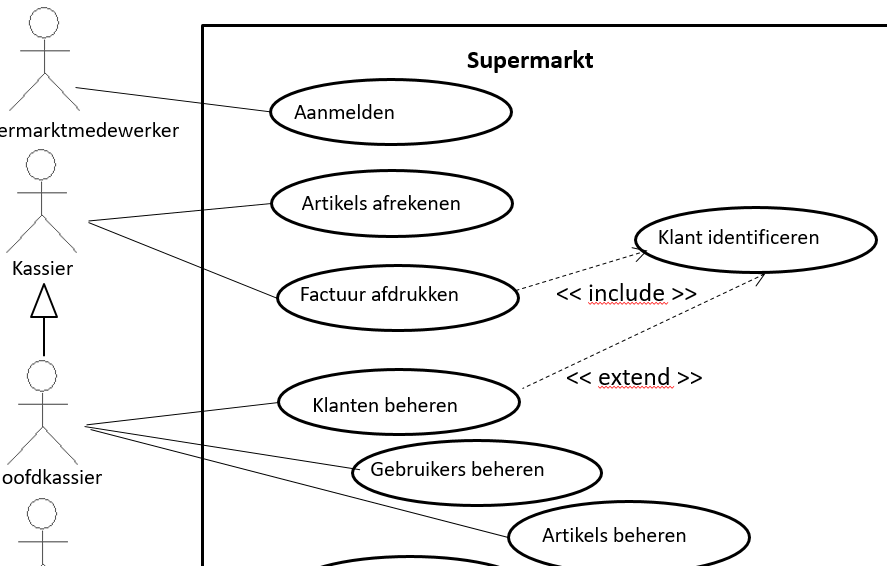


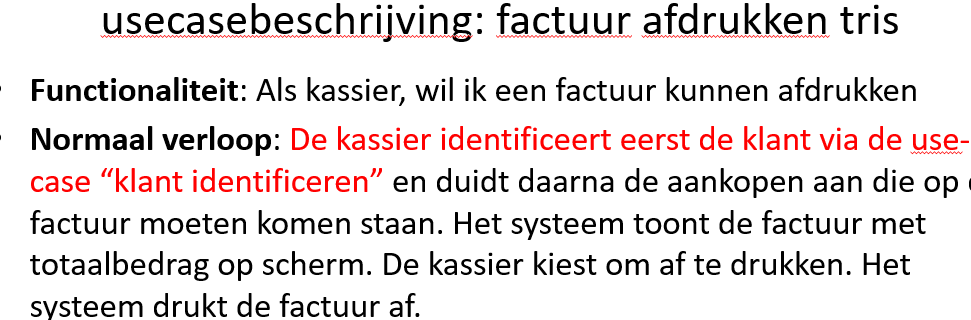


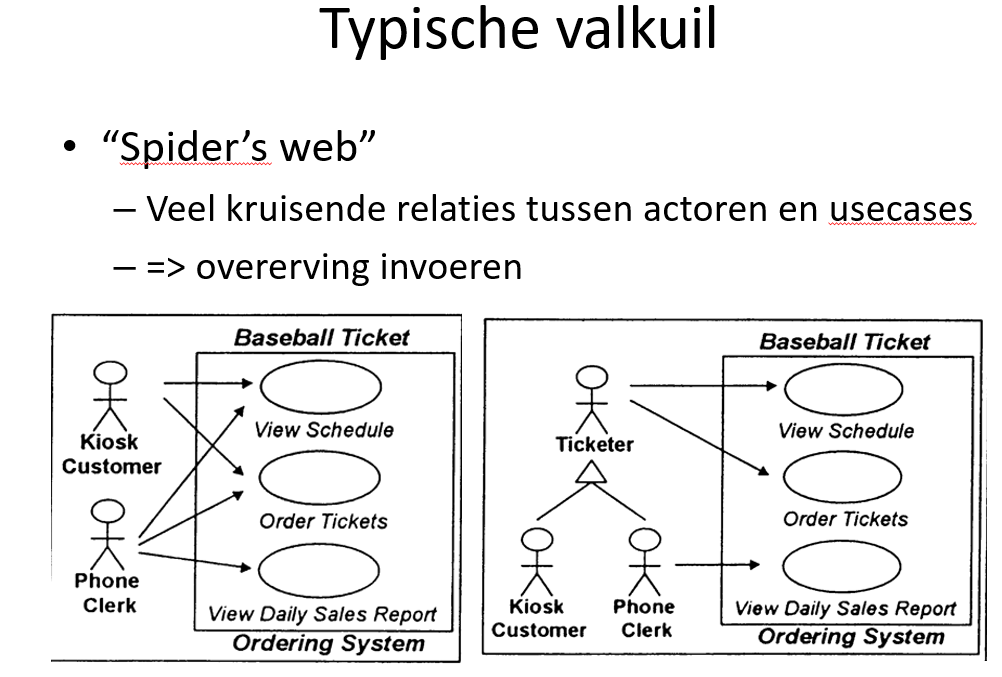
### Include-relatie

Geeft aan dat ene usecase gebruikt maakt van andere usecase waardoor gedrag van de eerste usecase verandert.

* Wordt altijd uitgevoerd
* Nieuwe usecase







## Sea-level niveau

* Opeenvolging van handelingen
* 1 persoon – 1 plaats – 1 tijdstip

## Kiss

* Keep it simple, stupid
* Keep it short & simple
* Keep it simple & sraightforward
* Keep it smart & simple
* Klein & eenvoudig:
  + Communicatie
  + Onderhoudbaarheid

# niet-functionele eisen

## Implementatie

Welke beperkingen op vlak van ontwerp & implemtentatie?

* Hardware/software
* Programmeertaal?
* Framework?
* Database?
* Kenmerken ontwikkelplatform?
* Kenmerken productieplatform?

## Externe interface

Externe interface nodig? Ja ->

* Welke input/output van/naar andere HW/SW systemen
* Beperkingen op formaat voor in/output?
* Beperkingen op medium voor in/output?

## Performantie

### Antwoordtijden

Hoe snel moet jouw systeem op bepaalde gebeurtenis reageren?

### Schaalbaarheid

Hoeveel gelijktijdige acties moeten kunnen plaatsvinden?

## Kwaliteitseisen

### Software

#### Beveiliging

Gericht op garanderen van:

* CIA (Confidentiality, integrity, availability) of information
* Wat zou systeem NIET mogen doen?
* Afhankelijk van kritikaliteit systeem

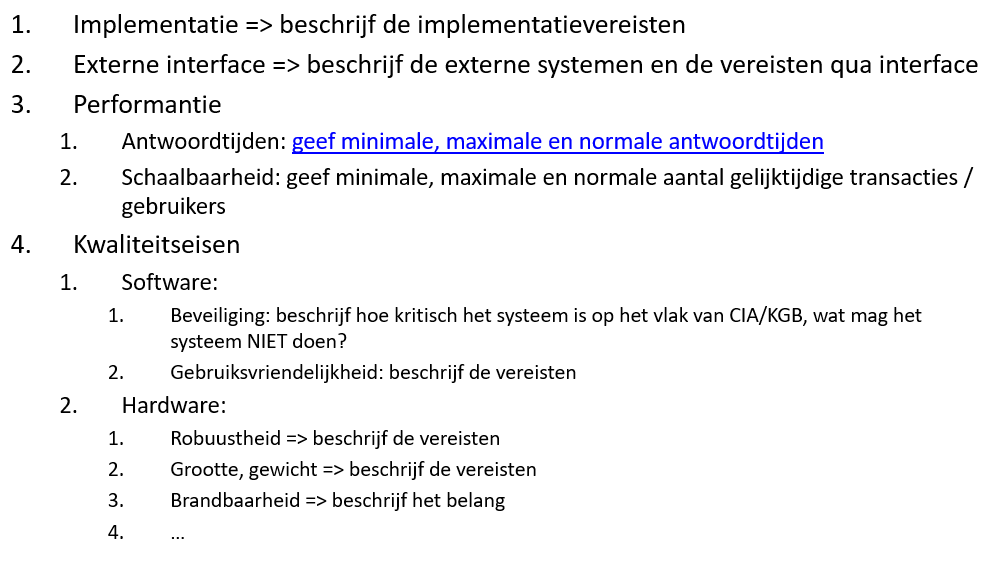
#### Gebruiksvriendelijkheid

* Menselijke factor; andersvaliden; gebruiksvriendelijkheid
* Welke “help” ondersteuning?

### Hardware

* Robuustheid
* Grootte, gewicht
* Brandbaarheid
* …

## Checklist



# website vs web applicatie

## Doelen

|  |  |
| --- | --- |
| Web site | Web applicatie |
| Inhoud | Taak gebaseerde invulformulieren |
| Browsen aanmoedigen | Snelle eenvoudige methodes |
| Zoeken aanmoedigen | Zeer weinig tekst |
| Ontdekken aanmoedigen |  |
| Bezoekers aantrekken |  |
| Bezoekers geïnteresseerd houden |  |

## Analyse gebruikers en taken

|  |  |
| --- | --- |
| Website | Web applicatie |
| Inhoud | Taak gebaseerde invulformulieren |
| Welke info nodig -> onderwerpen | Hoofdtaken |
| Welke info belangrijk -> home + hoofdpagina’s | Welke taken maken deel uit van andere takenb- |
| Hoofd – en subtopics -> navigatie | Welke taken worden hoe uitgevoerd |
| Geralteerde onderwerpen -> hyperlinks | Welke assistentie hebben gebruikers nodig |
|  | Hoe organiseren taken |

## Ontwerp

|  |  |  |
| --- | --- | --- |
|  | Website | Web applicatie |
| Bezochte links | Verschillende kleur | Zelfde kleur |
| Frames | Nadelen | Usability testing |
| Zoeken | Navigatie + verwijzingen op elke pagina | Snelle navigatie tussen verschillende formulieren |
| Pagina titels | Uniek per pagina | Zelfde voor elke pagina |
| Hyperlinks | Aanrader | Om snelle navigatie te ondersteunen |

## Functies

|  |  |  |
| --- | --- | --- |
|  | Website | Web applicatie |
| Scrolling | ??? | Usability testing |
| Verschillende browsers | Ondersteun alle | Keuzevrijheid |
| Browser buttons | Ja | Onverwachte resultaten -> alternatief voorzien |
| Pulldown navigatie menu’s | Ja | SNAFU vermijden |

## Home page

|  |  |
| --- | --- |
| Website | Web applicatie |
| Beschrijft verschillende inhoudsgebieden | Main menu |
| Nieuwe verhalen | Product overview of help voor beginners |
| What’s new/recent | Splash screen zonder toegevoegde waarde |
| Reclame voor producten/diensten | Statistieken, live data, download tijd |
|  | Niets/geen home page |

## Help

|  |  |
| --- | --- |
|  |  |
| FAQ | Traditionele online help |
| Welkom/overzicht | Info over applicatie |
| Site map | Context-sensitive help |
| Hoe website gebruiken | Systeemberichten |
|  | Links naar meer info |

# prototyping

## Definities

Prototyping = proces om prototypes te maken

Prototype = experimentele en onvolledige ontwerpen die snel ontwikkeld kunnen worden en niet duur zijn

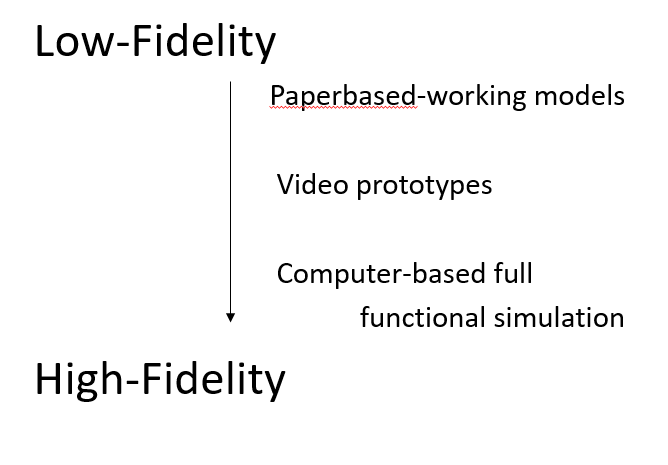
## Doel

* Visies en ideeën visualiseren om goede feedback van de gebruiker te krijgen (vroegere fase)
* Nieuwe mogelijkheden en oplossingen beschouwen in gemeenschappelijke taal

## Voordelen

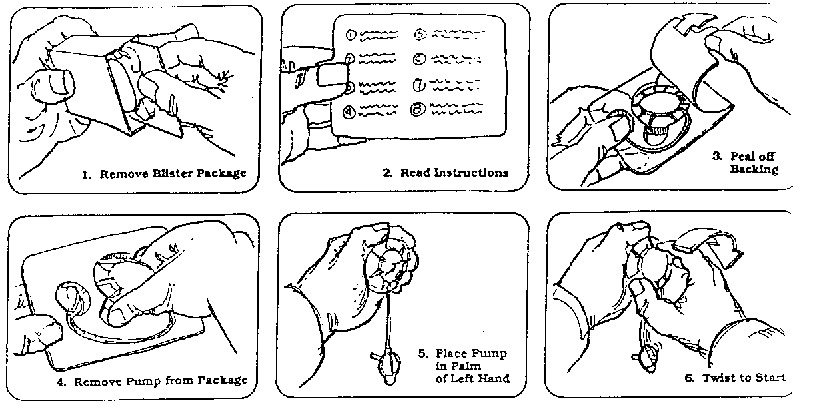
* Stimuleert vroegtijdige terugkoppeling
* Vermindert risico ingrijpende wijzigingen naderhand
* Vergroot aantal iteraties -> risico verlagend

## Soorten



## Storyboard

* Snapshots interface op belangrijke punten in interactie
* Schetsen die tonen hoe gebruiker bepaalde taak kan uitvoeren
* Vaak in combinatie met geschreven scenario
* Film-wereld -> acteurs helpen beeld te krijgen



## Hi-fi vs lo-fi

|  |  |  |
| --- | --- | --- |
| Type | Voordelen | Nadelen |
| Lo-fi | * Minder tijd + lage kost * Meerde design concepten geëvalueerd * Nuttig communicatiemiddel | * Beperkt van nut bij testing * Beperkingen flow en navigatie * <>user driven * Niet gedetailleerd genoeg -> programmeren |
| Hi-fi | * Gedeeltelijke/volledige functionaliteit * Interactief * User-driven * Navigatieschema gedefinieerd * Kan gebruikt worden voor testing * Instrument voor marketing en verkoop | * Meer tijd in beslag * Inefficiënt voor proef-of concept-design * Maakt gebruikers blind voor grote fouten * Management zou denken dat het echt is |

## Best practices hifi prototyping

* Nadruk functionaliteit
* Echte voorbeelden tonen
* Nummer/benoem schermen -> verwijzen
* 1 scherm tonen 2 mogelijke acties 3 scherm dat er op volgt
* KISS
* Uitleg in tekst indien niet vanzelfsprekend

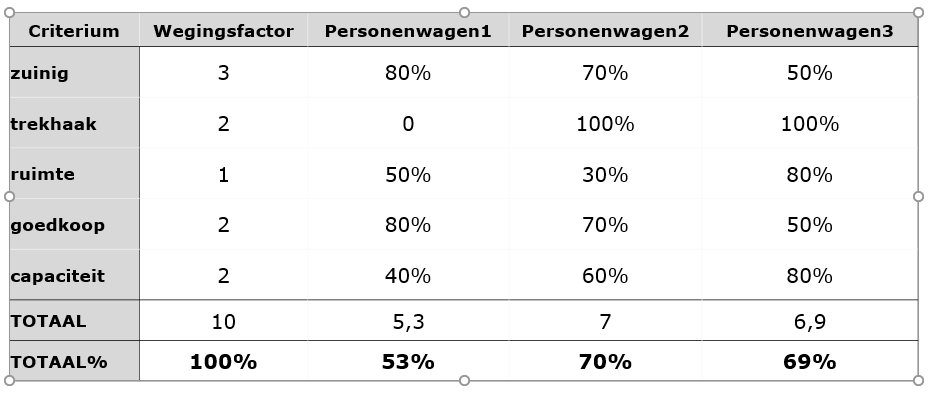
# WRM

## Wat?

* Weighted Ranking Method
* Objectieve manier keuze maken tussen gelijkaardige oplossingen

## Werkwijze

1. Identificeer eisen waaraan oplossing moet voldoen
2. Geef elk criterium wegingsfactor
3. Geef elke oplossing punten voor elk criterium
4. Voor elke oplossing: wegingsfactor X score -> som
5. Bereken percentage per tool
6. Formuleer een besluit



# datamodellering basis

## Soorten gegevens

### Samengestelde gegevens

* Gegevens die in één kolom worden samengevoegd
* Best splitsen

### Procesgegevens

* Berekend gegeven
* Niet bijhouden in tabel

### Primary key

* Één kolom of cominatie meerdere kollommen
* Uniek
* Minimaal
* Zonder inhoud

### Surrogate key

* PK die we invoeren omdat geen unieke PK aanwezig
* Autonummering

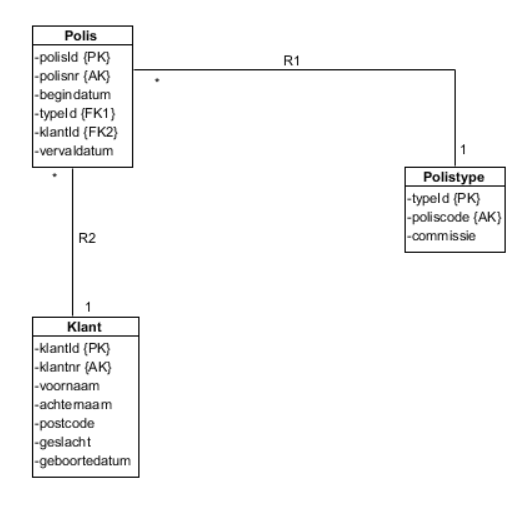
### Alternate key

* Één kolom van entiteit
* Uniek
* Minimaal
* Wel betekenis -> geen PK

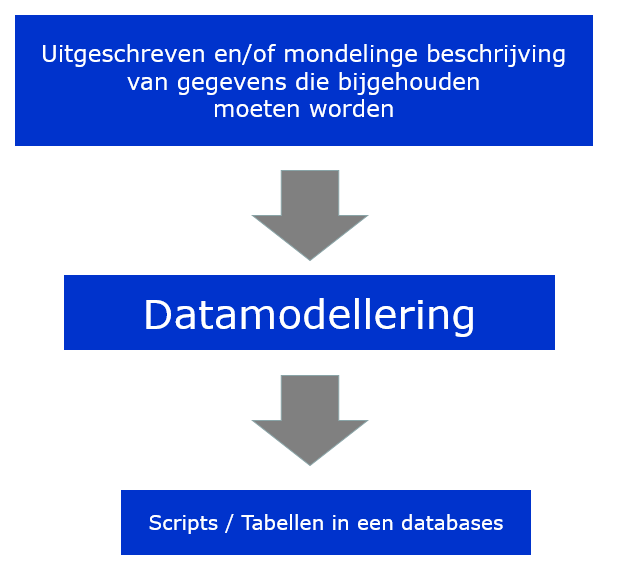
### Foreign key

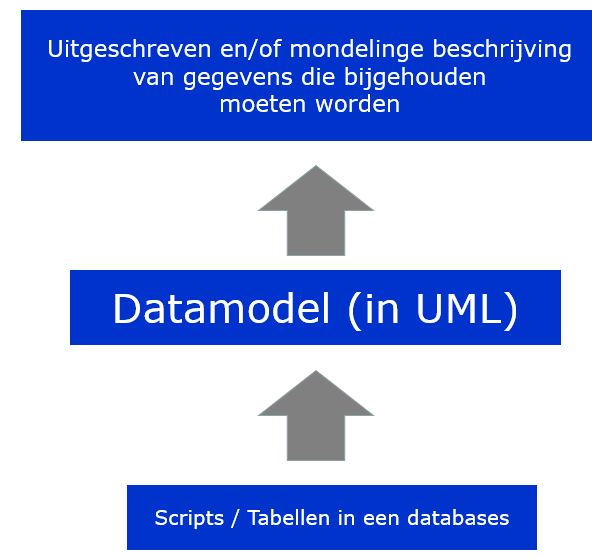
* Kolom in tabel die in andere tabel PK is
* Legt relatie tussen 2 tabellen

## Relaties tussen tabellen



## Datamodellering





### ERD

* = Entity Relationship Diagram
* Entiteiten met attributen, primary keys, foreign keys en relaties

Entiteit = tabel in database

Attribuut = kolom in tabel

Relatie = foreign key in tabel

### Goed datamodel

* Volledig
* Geen redundancy
* Forcen bedrijfsregels
* Herbruikbaarheid gegevens
* Stabiliteit en flexibiliteit
* Eenvoud en elegantie

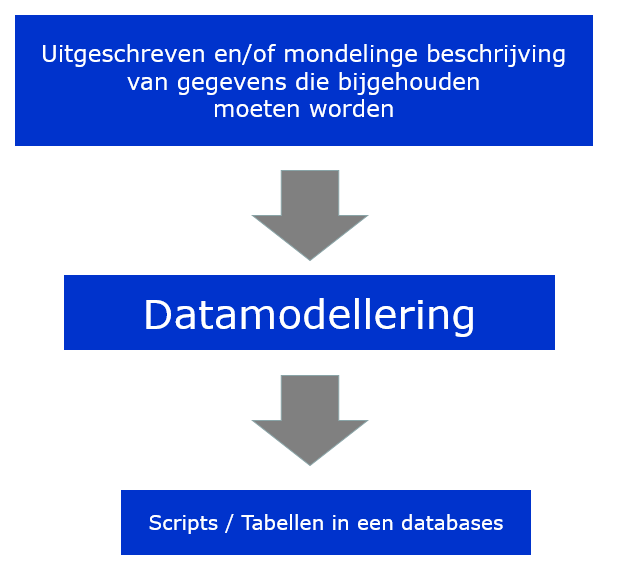
### Wanneer?

* Klassieke “functie gedreven aanpak”
* Klassieke “data gedreven aanpak”
* OO ontwikkeling

### UML notatie

* Naamgeving:
  + Entiteit: enkelvoud + Hoofdletter
  + Attribuut: enkelvoud + kleine letter
* Primary key: “PK” of onderlijnd
* Foreing key: fkx
  + X: sequentieel nummeren
  + Overeenkomstige relatie -> zelfde nummer
* Alternate key: AK
* Multipliciteit:
  + \* = meerdere, waar FK staat
  + 1 aan andere kant
* Relatie beschrijven in 2 richtingen: eerst van links -> rechts of aanduiden 
  + Altijd met “één” beginnen en multipliciteit andere kant aflezen

# synthesemethode



Model om, startend van beschrijving, ERD te maken -> omzetten in database

## Werkwijze

* Bepaal kernentiteiten met attributen
* Bepaal associatie-entiteiten met hun attributen (inclusief fk) en kies PK
* Bepaal overige relaties
* Herhaal stappen tot ontwerp af is
* Documenteer

## Toewijzen

### Kernentiteit

* Onafhankelijk
* Bestaat op zichzelf
* Datgene waarover we gegevens willen bijhouden

### Primaire sleutel

* Één attribuut of combinatie van meerdere
* Uniek binnen groep
* Minimaal
* Zonder inhoud/betekenis

### Alternate key

* = PK
* MAAR heeft inhoud/betekenis -> niet PK

### Surrogate key

* PK die we invoeren omdat:
  + Geen PK aanwezig die uniek, minimaal en zonder inhoud
  + PK bestaat uit meerdere attributen die verder als FK gebruikt worden
* Autonummering

### Associatie-entiteit

* Vervangt M op N koppeling tussen twee entiteiten
* Bevat FK ’s die verwijzen naar zijn entiteiten

### Foreign key

* Attribuut of combinatie van die in andere entiteit PK vormt
* Legt relatie tussen 2 entiteiten

### Karakteristieke entiteit

* Andere entiteit kwalificeren, karakteriseren of iets meer over vertellen
* =bestaansafhankelijk van entiteit die ze beschrijft
* Bevat FK die verwijst naar entiteit die ze beschrijft

# specs voor datamodellen

## Specificaties attributen

### NA/NNA

* Veld moet altijd ingevuld worden -> NNA; anders NA
* Niet elk veld dat ‘nuttig’ is, is verplicht

### Datatypes

* Hangt af van DBMS
* We weten enkel welk soort data we willen opslaan
* Voorlopige datatypes (int, float, string, …)

## Specificaties FK

### Optionaliteit relatie

* Andere kant relatie = 1..1 -> NNA
* Andere kant relatie = 0..1 -> NA



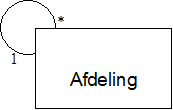
### DTR/DTC/DTN

* DTC:
  + Delete of Target Cascades
  + PK verwijderen -> record FK verwijdert mee
* DTR:
  + Delete of Target Restricted
  + Record PK kan niet verwijderd worden zolang die ergens anders record met FK heeft
  + Handmatig eerst FK’s verwijderen
* DTN:
  + Delete of Target Nullifies
  + PK verwijderen -> FK op Null gezet
  + Kan enkel als relatie 0..1 is

# special topics

## Recursieve koppeling

* Entiteit heeft relatie met zichzelf



0,1

0..\*

1

## Tijdsaspect

Elk gegeven -> afvragen of enkel actuele waarde of waarde uit verleden moet bijgehouden worden

### Enkel actuele waarde

* Rechtstreeks + enkelvoudig attribuut
* Bv actuele\_prijs in product

### Ook waarden verleden

* Rechtstreeks + meervoudig attribuut
* Nieuwe karakteristieke entiteit

## Standaardwaarden en uitzonderingen

### Enkel standaardwaarde

* Wanneer uitzondering heel uitzonderlijk is
* Situatie doet zich voor -> beschouwen als nieuwe registratie

### Enkel uitzondering

* Uitzondering toch niet zo uitzonderlijk
* Standaardwaarde geldt toch -> voor elke uitzondering zelfde standaardwaarde

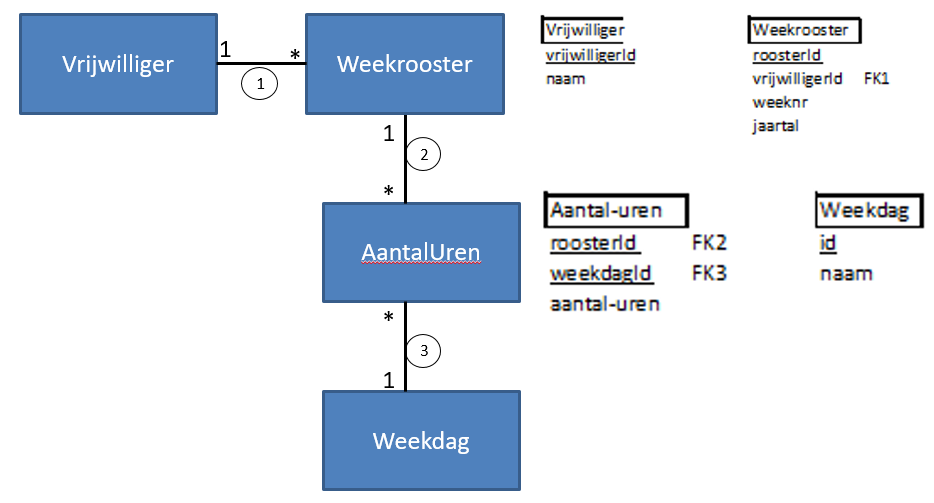
### Beide

* Standaard + uitzondering bijgehouden in geval van tussenliggende situatie
* Uitzondering toegepast indien ingevuld

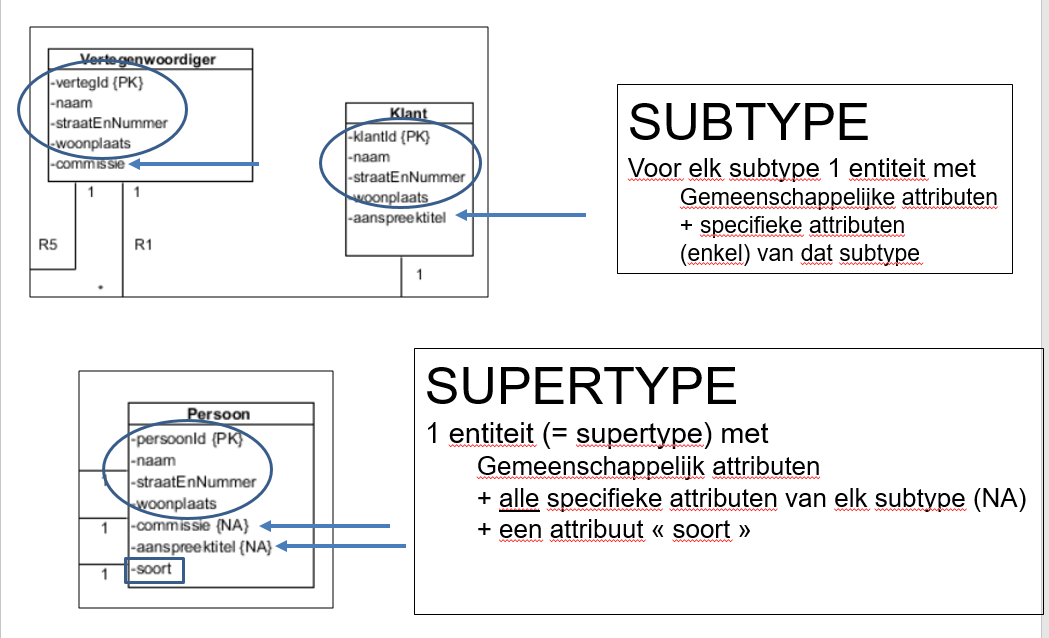
## Vast aantal coderingen



Wordt



# super- en subtypen



## Supertype

* 1 entiteit
* Gemeenschappelijke attributen
* + alle specifieke attributen elk subtype (NA)
* + attribuut “soort”

### Voordelen

* Flexibeler
* Eenvoudiger
* Geen redundante attribuutdefinities

### Nadelen

* Bedrijfsregels niet geïmplementeerd
* Lege attributen

## Subtype

* Elk subtype -> 1 entiteit
* Gemeenschappelijke attributen
* Specifieke attributen enkel van dat subtype

### Voordelen

* Bedrijfsregels vastgelegd
* Geen lege attributen

### Nadelen

* Minder stabiel/flexibel
* Redundante attribuutdefinities
* Doorzoeken alle voorkomens vereist doorzoeken meerdere entiteiten
* Disjunct en exhaustief