[Jaar]

Eisenanalyse

ROBBERT

1 INHOUD

2 INLEIDING

2.1 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?

2.2 SOORTEN EISEN

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

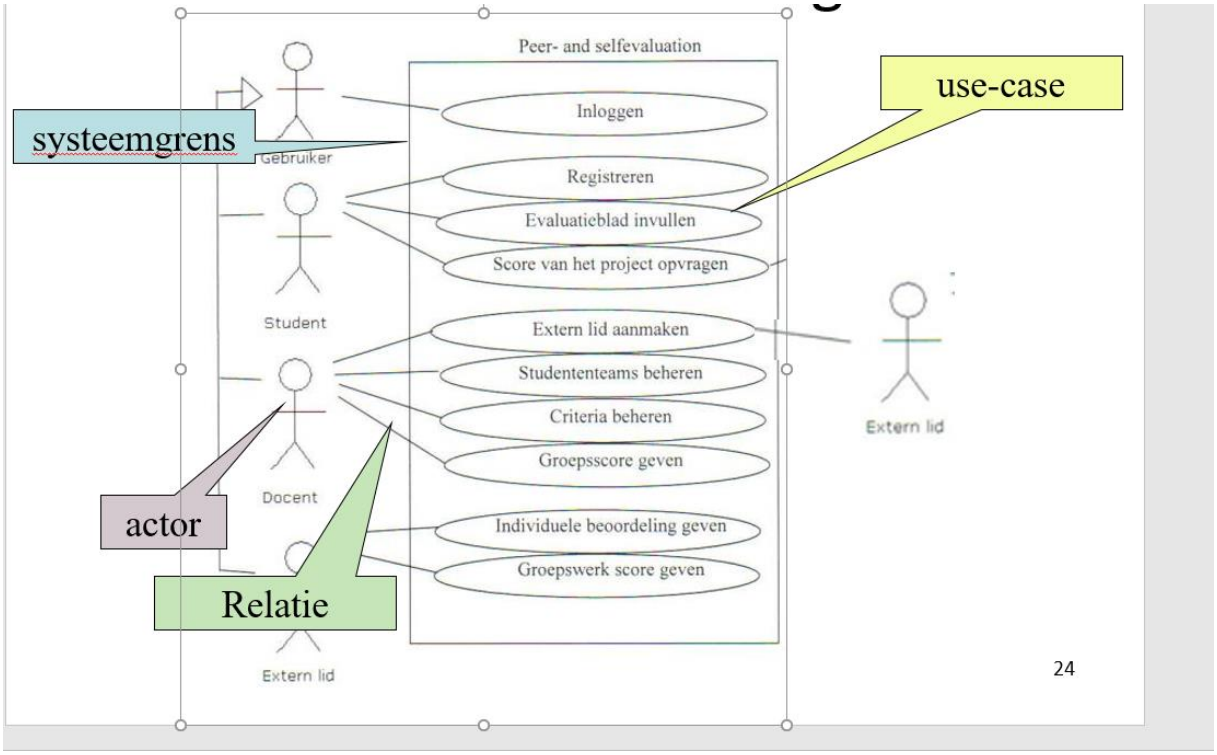
2.3 HET USECASEMODEL

= 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

2.3.1 Doelstellingen

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

2.3.2 Onderdelen

- Systeemgrens
- Actoren
- Usecases
- Relaties
- Beschrijving

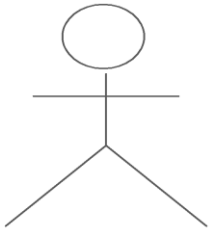
2.3.2.1 Systeemgrens

=scope

- Systeem = rechthoek met systeemnaam
- = start analyse

2.3.2.2 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?

2.3.2.3 Usecase

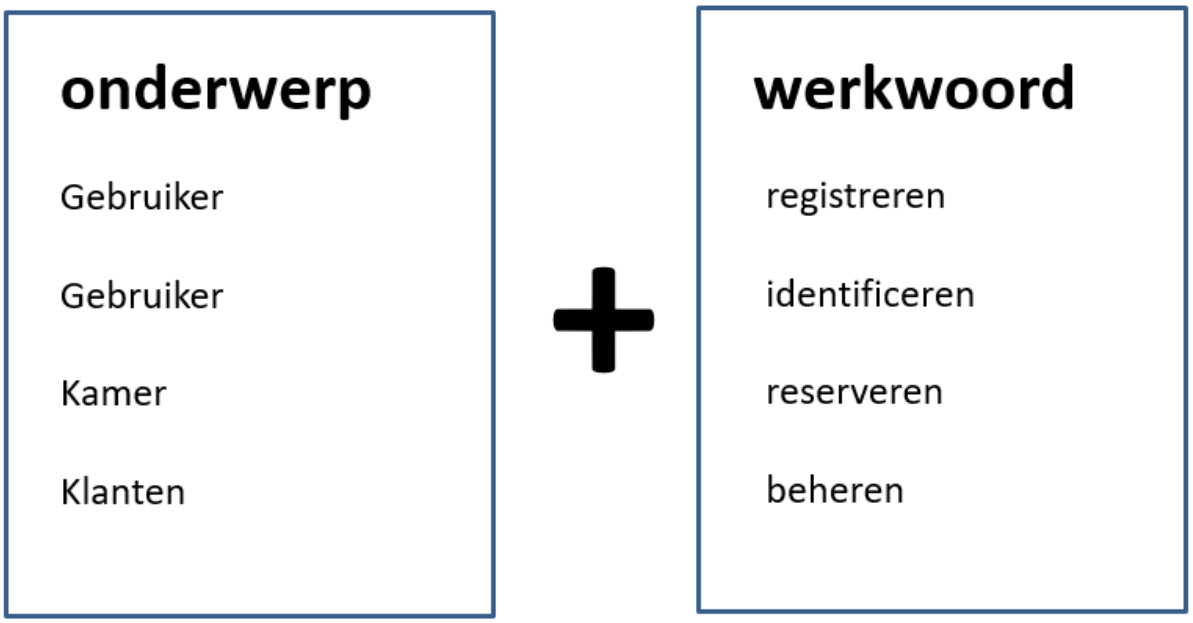
- Geval/situatie waarvoor gebruiker systeem wil gebruiken
- Doelstelling waarvoor >=1 actoren systeem willen gebruiken
- Opeenvolging van interacties tussen >=1 actoren en het systeem met welbepaald doel

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



2.3.2.4 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

2.3.3 Stappenplan usecasemodellering

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

2.3.4 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?

3 EISENANALYSE – UITGEBREID

3.1 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?

3.2 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?

3.3 USE CASES ZOEKEN (ACTOR BASED)

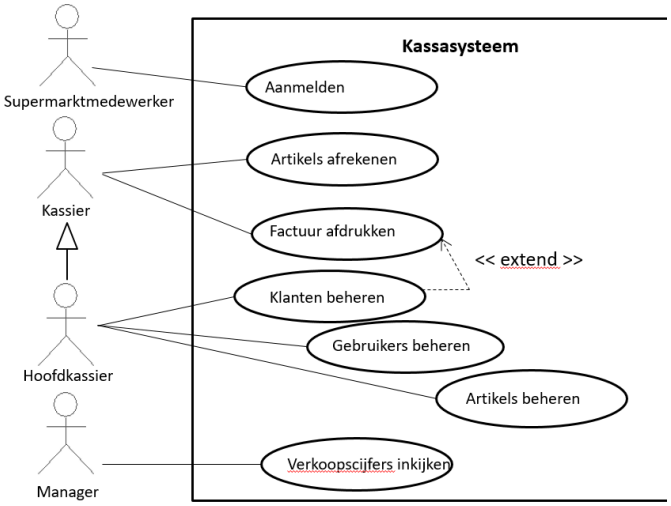
- Belangrijkste taken die systeem voor actor moet uitvoeren?
- Zal actor data aanmaken, opslaan, wijzigen, verwijderen of lezen?
- Moet actor systeem informeren over externe wijzigingen?
- Moet actor geïnformeerd worden over bepaalde gebeurtenissen?

3.4 RELATIES TUSSEN USECASES

3.4.1 Extend-relatie (met conditie)

Geeft aan dat aan gedrag usecase extra gedrag wordt toegevoegd

- Wordt optioneel uitgevoerd
- Nieuw gedrag in nieuwe usecase

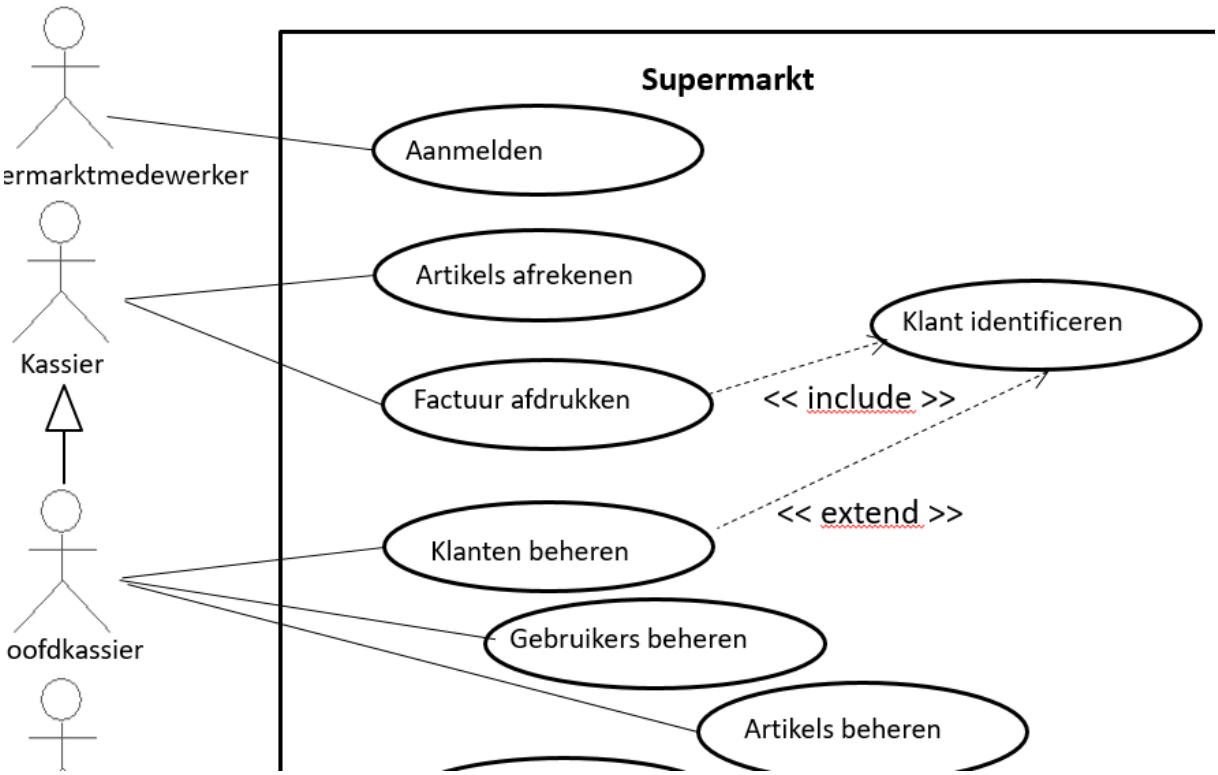


Alternatieven:
[a] indien de kassier er niet in slaagt de barcode van de factuurkaart te scannen heeft hij de mogelijkheid om de code manueel in te voeren
[b] Indien de klant geen klantenkaart heeft kan de hoofdkassier de usecase "klanten beheren" starten waar hij een nieuwe klant kan creëren en vervolgens voor hem artikels afrekenen.

3.4.2 Include-relatie

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

- Wordt altijd uitgevoerd
- Nieuwe usecase

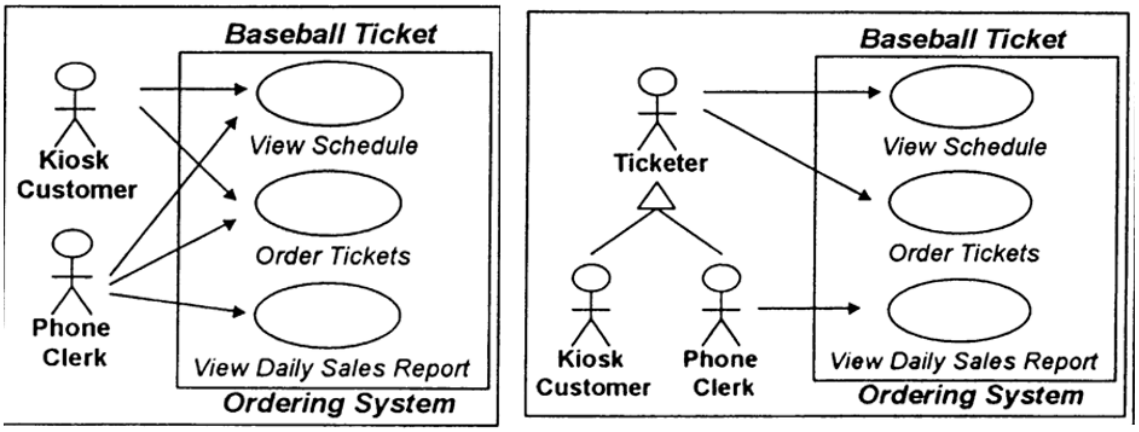


usecasebeschrijving: factuur afdrukken tris

- **Functionaliteit**: Als kassier, wil ik een factuur kunnen afdrukken
- **Normaal verloop**: De kassier identificeert eerst de klant via de use-case "klant identificeren" en duidt daarna de aankopen aan die op factuur moeten komen staan. Het systeem toont de factuur met totaalbedrag op scherm. De kassier kiest om af te drukken. Het systeem drukt de factuur af.

Typische valkuil

- "Spider's web"
  - Veel kruisende relaties tussen actoren en usecases
  - => overerving invoeren



3.5 Sea-level niveau

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

3.6 Kiss

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

4 NIET-FUNCTIONELE EISEN

4.1 IMPLEMENTATIE

Welke beperkingen op vlak van ontwerp & implemtentatie?

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

4.2 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?

4.3 PERFORMANTIE

4.3.1 Antwoordtijden

Hoe snel moet jouw systeem op bepaalde gebeurtenis reageren?

4.3.2 Schaalbaarheid

Hoeveel gelijktijdige acties moeten kunnen plaatsvinden?

4.4 KWALITEITSEISEN

4.4.1 Software

4.4.1.1 Beveiliging

Gericht op garanderen van:

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

4.4.1.2 Gebruiksvriendelijkheid

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

4.4.2 Hardware

- Robuustheid
- Grootte, gewicht
- Brandbaarheid
- …

4.5 CHECKLIST

1. Implementatie => beschrijf de implementatievereisten
2. Externe interface => beschrijf de externe systemen en de vereisten qua interface
3. Performantie
    1. Antwoordtijden: geef minimale, maximale en normale antwoordtijden
    2. Schaalbaarheid: geef minimale, maximale en normale aantal gelijktijdige transacties / gebruikers
4. Kwaliteitseisen
    1. Software:
        1. Beveiliging: beschrijf hoe kritisch het systeem is op het vlak van CIA/KGB, wat mag het systeem NIET doen?
        2. Gebruiksvriendelijkheid: beschrijf de vereisten
    2. Hardware:
        1. Robuustheid => beschrijf de vereisten
        2. Grootte, gewicht => beschrijf de vereisten
        3. Brandbaarheid => beschrijf het belang
        4. ...

5 WEBSITE VS WEB APPLICATIE

5.1 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	

5.2 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

5.3 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

5.4 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

5.5 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

5.6 HELP

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

6 PROTOTYPING

6.1 DEFINITIES

Prototyping = proces om prototypes te maken

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

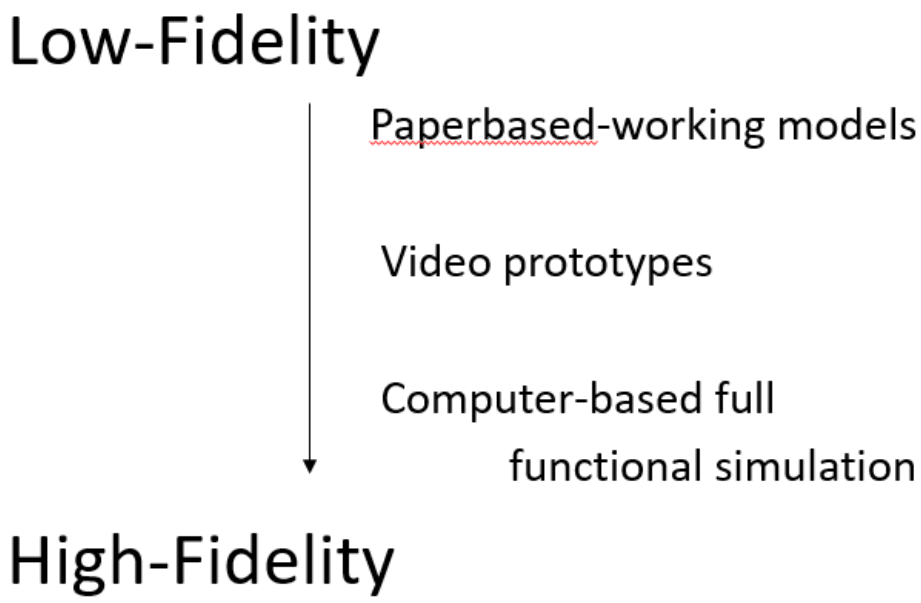
6.2 DOEL

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

6.3 VOORDELEN

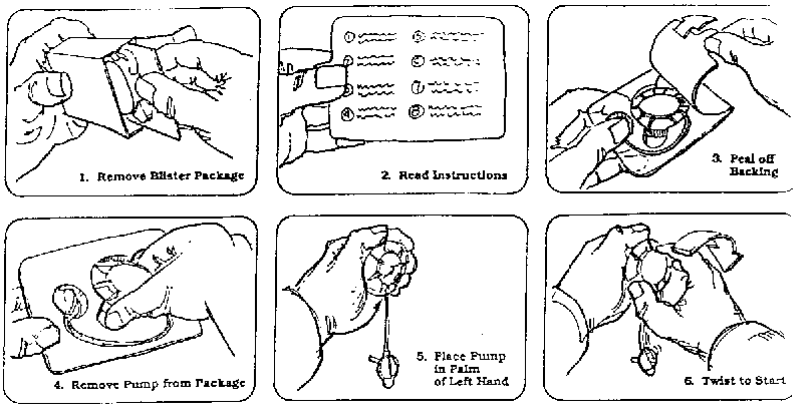
- Stimuleert vroegtijdige terugkoppeling
- Vermindert risico ingrijpende wijzigingen naderhand
- Vergroot aantal iteraties -> risico verlagend

6.4 SOORTEN



6.5 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



6.6 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

6.7 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

7 WRM

7.1 WAT?

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

7.2 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

Criterium	Wegingsfactor	Personenwagen1	Personenwagen2	Personenwagen3
zuinig	3	80%	70%	50%
trekhaak	2	0	100%	100%
ruimte	1	50%	30%	80%
goedkoop	2	80%	70%	50%
capaciteit	2	40%	60%	80%
TOTAAL	10	5,3	7	6,9
TOTAAL%	**100%**	**53%**	**70%**	**69%**

8 DATAMODELLERING BASIS

8.1 SOORTEN GEGEVENS

8.1.1 Samengestelde gegevens

- Gegevens die in één kolom worden samengevoegd
- Best splitsen

8.1.2 Procesgegevens

- Berekend gegeven
- Niet bijhouden in tabel

8.1.3 Primary key

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

8.1.4 Surrogate key

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

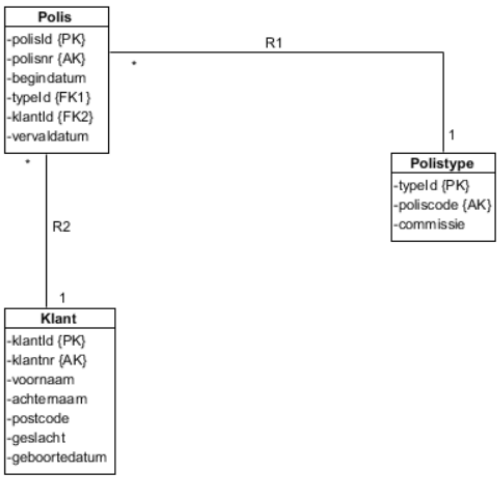
8.1.5 Alternate key

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

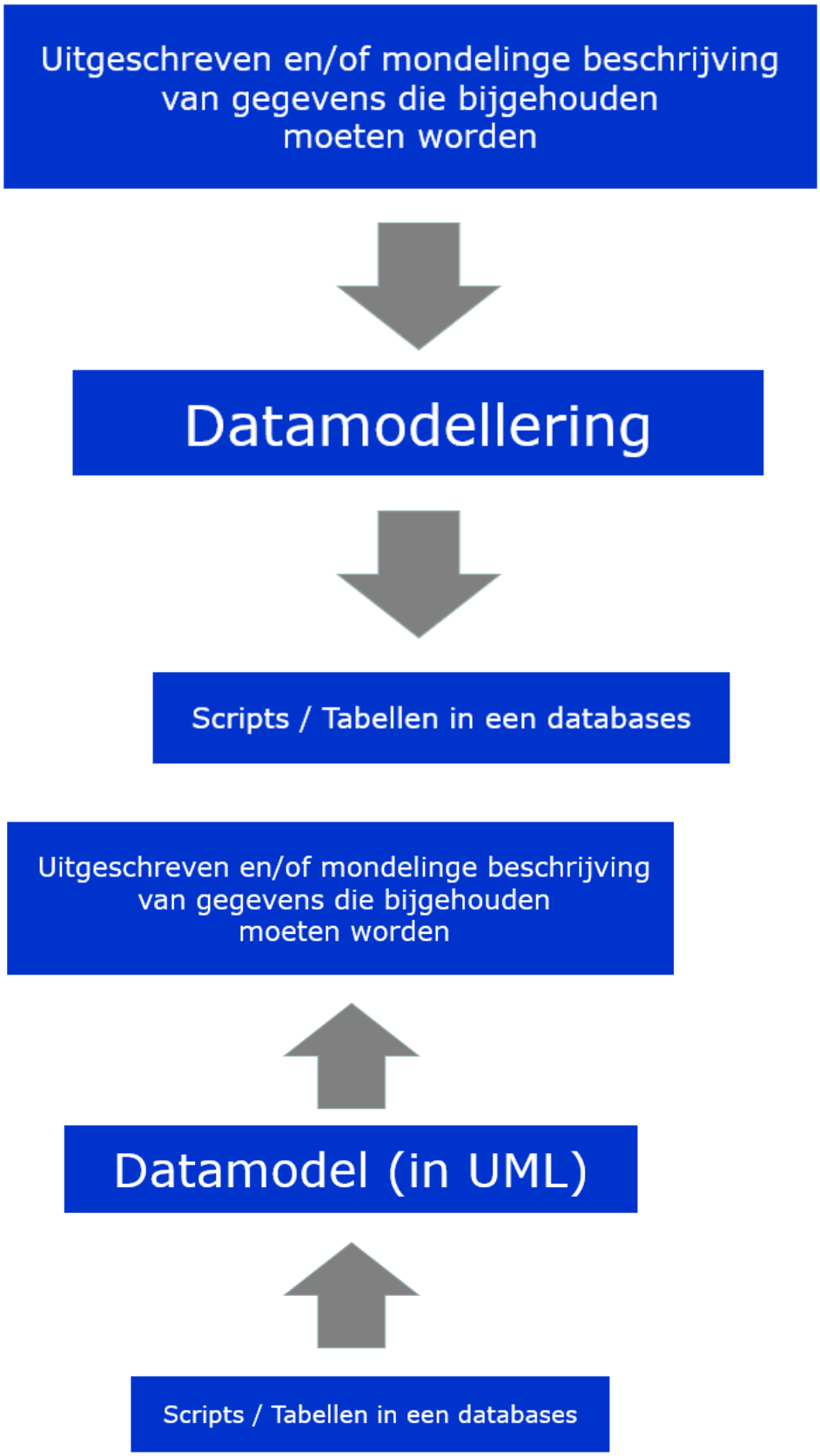
8.1.6 Foreign key

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

8.2 RELATIES TUSSEN TABELLEN



8.3 DATAMODELLERING



8.3.1 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

8.3.2 Goed datamodel

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

8.3.3 Wanneer?

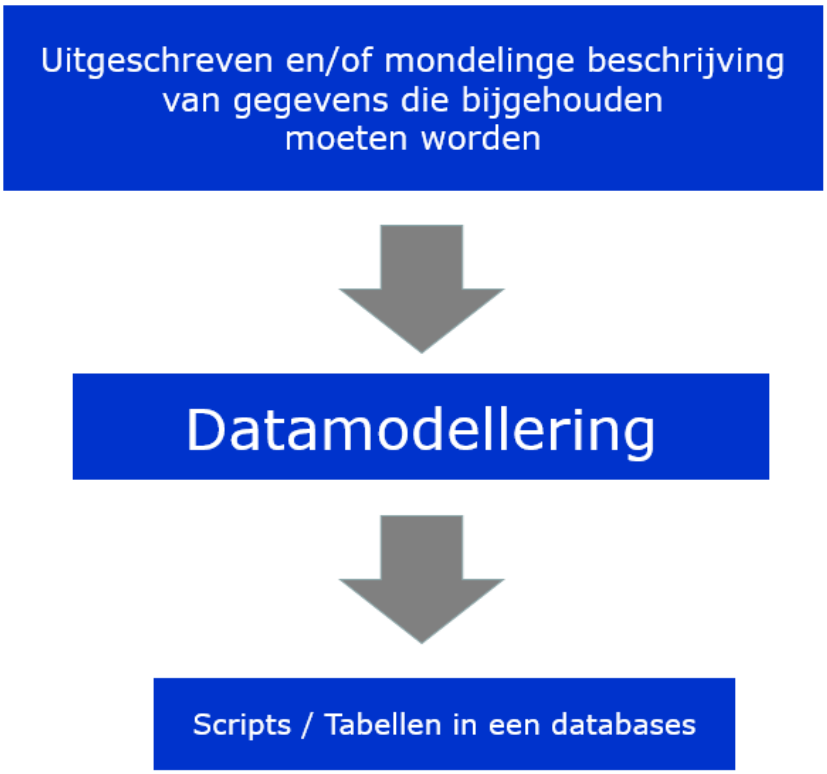
- Klassieke “functie gedreven aanpak”
- Klassieke “data gedreven aanpak”
- OO ontwikkeling

8.3.4 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

9 SYNTHESEMETHODE



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

9.1 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

9.2 TOEWIJZEN

9.2.1 Kernentiteit

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

9.2.2 Primaire sleutel

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

9.2.3 Alternate key

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

9.2.4 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

9.2.5 Associatie-entiteit

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

9.2.6 Foreign key

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

9.2.7 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

10 SPECS VOOR DATAMODELLEN

10.1 SPECIFICATIES ATTRIBUTEN

10.1.1 NA/NNA

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

10.1.2 Datatypes

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

10.2 SPECIFICATIES FK

10.2.1 Optionaliteit relatie

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

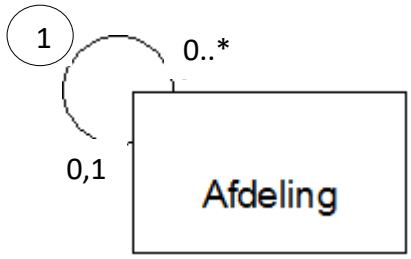
10.2.2 DTR/DTC/DTN

- 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

11 SPECIAL TOPICS

11.1 RECURSIEVE KOPPELING

- Entiteit heeft relatie met zichzelf



11.2 TIJDSASPECT

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

11.2.1 Enkel actuele waarde

- Rechtstreeks + enkelvoudig attribuut
- Bv actuele_prijs in product

11.2.2 Ook waarden verleden

- Rechtstreeks + meervoudig attribuut
- Nieuwe karakteristieke entiteit

11.3 STANDAARDWAARDEN EN UITZONDERINGEN

11.3.1 Enkel standaardwaarde

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

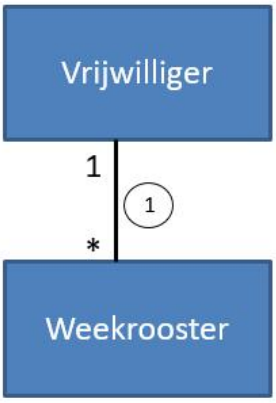
11.3.2 Enkel uitzondering

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

11.3.3 Beide

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

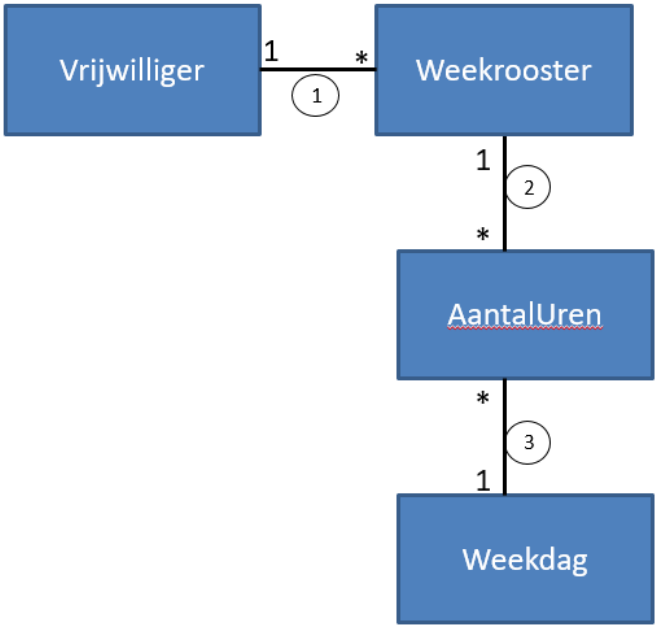
11.4 VAST AANTAL CODERINGEN



Vrijwilliger
vrijwilligerId
naam

Weekrooster
roosterId
vrijwilligerId FK1
weeknr
jaartal
aantal-uren-op-maandag
aantal-uren-op-dinsdag
aantal-uren-op-woensdag
aantal-uren-op-donderdag
aantal-uren-op-vrijdag

Wordt



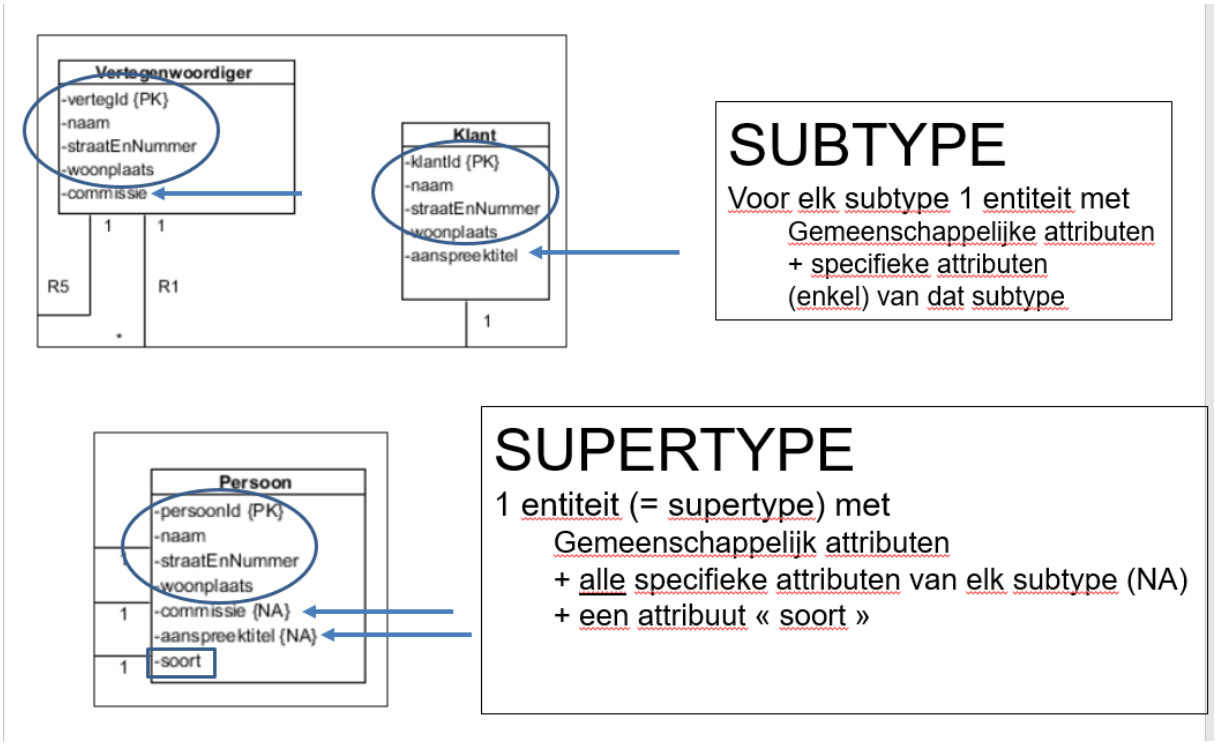
Vrijwilliger
vrijwilligerId
naam

Weekrooster
roosterId
vrijwilligerId FK1
weeknr
jaartal

Aantal-uren
roosterId FK2
weekdagId FK3
aantal-uren

Weekdag
id
naam

12 SUPER- EN SUBTYPEN



12.1 SUPERTYPE

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

12.1.1 Voordelen

- Flexibeler
- Eenvoudiger
- Geen redundante attribuutdefinities

12.1.2 Nadelen

- Bedrijfsregels niet geïmplementeerd
- Lege attributen

12.2 SUBTYPE

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

12.2.1 Voordelen

- Bedrijfsregels vastgelegd
- Geen lege attributen

12.2.2 Nadelen

- Minder stabiel/flexibel

- Redundante attribuutdefinities
- Doorzoeken alle voorkomens vereist doorzoeken meerdere entiteiten
- Disjunct en exhaustief