|  |  |
| --- | --- |
|  |  |
| 7/4/2017 | **BIS FO Reader** |

[Inleiding 3](#_Toc486945745)

[1 Documenteren van processen volgens BPMN en IPO 4](#_Toc486945746)

[1.1 BPMN level 1 4](#_Toc486945747)

[1.2 IPO 6](#_Toc486945748)

[1.3 Tips & trucs / ‘the art of modeling’ 7](#_Toc486945749)

[2 Requirements 9](#_Toc486945750)

[2.1 Wat zijn requirements? 9](#_Toc486945751)

[2.2 Requirements beschrijven 9](#_Toc486945752)

[3 Wireframe’s 11](#_Toc486945753)

[4 Use Cases 16](#_Toc486945754)

# Documenteren van processen volgens BPMN en IPO

## BPMN level 1

Het modelleren van bedrijfsprocessen, oftewel het maken van een procesmodel, is niet gemakkelijk. Dit wordt bevestigd door het onderzoek van Michael Rosemann uit 2006, waarin hij 22 valkuilen identificeert. Dit onderzoek is bijgevoegd als bijlage en kun je beschouwen als een stuk achtergrondinformatie bij het modelleren van bedrijfsprocessen. (het is geen toetsstof)

Bij het modelleren van bedrijfsprocessen is het essentieel om te komen tot procesmodellen van hoge kwaliteit. Twee 2 aspecten spelen hierbij een belangrijke rol: de syntax en de semantiek. De kwaliteit van de syntax, oftewel Correct-by-design, resulteert in procesmodellen, die overeenkomen met de regels van de techniek waarin deze zijn gemodelleerd. De kwaliteit van de semantiek, oftewel Truthful-by-design, resulteert in procesmodellen die valide en compleet zijn. Compleet betekent hier dat procesmodellen geen belangrijke aspecten missen, die nodig zijn voor het behalen van de doelstelling van het procesmodel.

**Syntax – notatiewijzen - BPMN**

De syntax is terug te vinden in de verschillende notaties / notatiewijzen. Enkele voorbeelden van notatiewijzen zijn:

* Business Process Modeling Notation (BPMN);
* Event Driven Process Chain (EPC);
* Unified Modeling Language (UML);
* IDEF (Integrated DEFinition) methods;
* Petrinetten.

In BIS zullen wij een bedrijfsproces modelleren a.d.h.v. de notatiewijze ‘BPMN’. Meer informatie over de notatiewijze ‘BPMN’ is te vinden op de officiële BPMN specificatie website van de Object Management Group (OMG). Directe URL: <http://www.omg.org/spec/BPMN/>

**Semantiek – doel procesmodel – BPMN Level 1, 2 en 3**

Procesmodellen worden binnen organisaties gebruikt voor diverse doeleinden. Vaak ligt hierbij de nadruk op communicatie tussen belanghebbenden en het creëren van een gezamenlijk begrip van het bedrijfsproces. Veelvoorkomende doelen van een procesmodel zijn:

* Voldoen aan wet- en regelgeving. Bijvoorbeeld de Sarbanes-Oxley (SOX) Act of de Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH).
* Certificering / kwaliteitsmanagement. Denk hierbij aan de ISO 9000 standaard, onderhouden door de International Organization for Standardization (ISO), welke documentatie en monitoring vraagt van alle kernprocessen om hierdoor zeker te stellen dat deze effectief zijn ingericht.
* Communicatiedoeleinden, zoals het een (werk)instructie voor nieuwe medewerkers.
* Configureren van een zogeheten process-aware information system (PAIS), zoals een workflowmanagement (WFM) systeem.
* Uitvoeren van een modelmatige procesanalyse en simulatie.

Er zijn twee soorten notaties om een bedrijfsproces te modelleren, waarbij er een relatie aanwezig is tussen het soort notatie en het doel van het procesmodel. In de eerste plaats zijn er de intuïtieve, grafische notaties, zoals EPC, die voornamelijk worden gebruikt voor het begrijpen van het bedrijfsproces en voor discussie over business vereisten en procesverbetering. Aan de andere kant zijn er notaties, die zijn gebaseerd op wiskundige paradigma’s, zoals petri nets, die worden gebruikt voor procesanalyse, procesexecutie of processimulatie.

BPMN is een breed bruikbare notatiewijze waarmee alle 5 hierboven beschreven doelen behaald kunnen worden. Dit maakt dat per doel goed gekeken moet worden welke objecten/attributen voor dit doel benodigd zijn en gebruikt moeten worden. Sommige objecten/attributen zijn alleen nodig wanneer het doel van het procesmodel is om een PAIS/WFM te configureren. Andere objecten/attributen zijn alleen nodig wanneer het doel van het procesmodel het uitvoeren van een modelmatige procesanalyse en simulatie is.

Kortom, welke objecten binnen BPMN moet ik juist wel gebruiken of juist niet gebruiken wanneer ik een bepaald doel van het procesmodel voor ogen heb? Een korte toelichting voordat je aan de hoofdstukken uit het boek van Bruce Silver begint (samen met deze reader de toetsstof voor dit onderdeel)

Binnen de BPMN specificatie (uitgelegd door Bruce Silver in hoofdstuk 2) wordt er gesproken over Process Modeling Conformance. Hierin zijn 3 subclasses geïdentificeerd, te weten: Descriptive, Analytic en Common Executable. Elk van deze subclasses zou je kunnen zien als een doel waarvoor je het procesmodel wilt inzetten/gebruiken. De BPMN specificatie geeft de volgende omschrijving m.b.t. deze 3 subclasses:

* Descriptive is concerned with visible elements and attributes used in high-level modeling. It should be comfortable for analysts who have used BPA flowcharting tools.
* Analytic contains all of Descriptive and in total about half of the constructs in the full Process Modeling Conformance Class. It is based on experience gathered in BPMN training and an analysis of user-patterns in the Department of Defense Architecture Framework and planned standardization for that framework. Both Descriptive and Analytic focus on visible elements and a minimal subset of supporting attributes/elements.
* Common Executable focuses on what is needed for executable process models.

Vervolgens worden deze 3 subclasses gerelateerd aan de verschillende BPMN objecten en zo ontstaan de zogeheten BPMN levels van Bruce Silver. BPMN Level 1 komt hierin overeen met de subclass ‘Descriptive’, BPMN Level 2 komt hierin overeen met de subclass ‘Analytic’ en BPMN Level 3 komt hierin overeen met de subclass ‘Common Execution’.

Meer informatie over de BPMN Levels kan worden gevonden in het boek van Bruce Silver.

In BIS richten wij ons op de communicatie tussen belanghebbenden en het creëren van een gezamenlijk begrip van het bedrijfsproces als zijnde het doel van het procesmodel. Dit komt overeen met de subclass ‘Descriptive’ van de BPMN specificatie en BPMN Level 1 uit het boek van Bruce Silver.

Het doel van het procesmodel dat je opneemt in je functioneel ontwerp is een zogeheten communicatiemodel: hierin maak je helder naar je opdrachtgever toe dat je begrijpt in welke context het informatiesysteem werkt. Het doel van dit communicatiemodel is dat het je houvast geeft wanneer je met je opdrachtgever spreekt over de bedrijf)processen. Dit betekent dat je niet ieder detail hoeft op te nemen, maar belangrijke elementen als keuzemomenten in het proces evenals de overdrachtsmomenten wel. Wees je ervan bewust dat het hier niet gaat om een systeemontwerp, dat gaan we in de volgende hoofdstukken doen.

Belangrijk onderdeel van het proces is de interactie tussen klant en informatiesysteem. Binnen de theorie over het beschrijven van processen is het de norm om de klant te zien als iets waar de organisatie geen invloed op heeft en dus wordt zij niet beschreven in de procesbeschrijving. Zij wordt als een zogeheten ‘black box’ weergegeven. Wel wordt de interactie vastgelegd tussen het bedrijfsproces en deze klant als ‘black box’.

Tegenwoordig wordt de klant meer en meer gezien als een participant in het (bedrijfs)proces in plaats van een buitenstaander, waarop de organisatie geen invloed heeft. Denk hierbij aan zaken zoals bijvoorbeeld ‘customer self service’ en ‘customer journey’. Organisaties ondersteunen deze zaken door informatiesystemen, waarmee zij in staat zijn om te kunnen afdwingen op welke wijze de klant door het (klant)proces navigeert, welke keuzes er door de klant gemaakt kunnen worden en welke gegevens de klant kan en/of moet invullen. Dit maakt dat de klant tegenwoordig vaak wel wordt beschreven in een procesbeschrijving en als een zogeheten ‘white box’ wordt weergegeven, inclusief de interactie met het bedrijfsproces.

Belangrijk: Voor het doel van het procesmodel uit de casus zijn de klantkeuzes van belang. Modelleer dan ook een proces waarbij je de hele scope van het informatiesysteem, inclusief klantacties, meeneemt.

## IPO

Bij het modelleren van bedrijfsprocessen kan er aan elk object binnen een procesmodel een of meerdere eigenschappen worden toegekend en/of beschreven. Je bent dus niet klaar met alleen een ‘procesplaat’, daar hoort een beschrijving / toelichting bij.

Vanuit het eerder aangegeven doel van het procesmodel in BIS is het niet nodig om dit voor alle objecten te doen. Wij hebben de keuze gemaakt om dit alleen voor het object ‘Task’, oftewel de taken, te doen.

Daarnaast hoeven de eigenschappen het niet heel gedetailleerd te worden beschreven. Wij hebben er dan ook voor gekozen om de taken te documenteren volgens de IPO techniek.

IPO staat voor *Input*, *Process* en *Output*. De pagina wordt opgedeeld in drie kolommen. Per kolom wordt op een structureerde wijze in tekst de input, de werking van de taak en de output beschreven. Hierbij geldt de wet van behoud van gegevens “er kunnen geen gegevens zomaar ontstaan of zomaar verloren gaan”. IPO-schema’s zijn eenvoudig te vervaardigen en zijn goed leesbaar. Pas op met grote hoeveelheden van deze schema’s. De samenhang kan dan een probleem worden, omdat relaties met andere processen een zwak punt zijn van deze techniek.

*Voorbeeld IPO-schema*

In tabel 1. Zijn een tweetal processen beschreven: Controleren Bestelling en Inplannen Order. Deze taken zijn uitgewerkt in een IPO-schema.

Een “Telefonische bestelling” wordt omgezet naar een Order. Een “Bestelling Betaling” wordt omgezet naar een Order.

Tabel 1. Voorbeeld IPO-schema

|  |  |  |
| --- | --- | --- |
| Input | Proces (verwerking) | Output |
| Telefonische bestelling | Uitvragen combinatie postcode en geboortedatum.  Indien bekende klant omzetten bestelling naar Order. | Order |
| Order | Omzetten van een “Bestelling betaling” naar een Order.  Op basis van Order en planning bepalen te verwachten levertijd. | Kassabon  Verwachte levertijd |

## Tips & trucs / ‘the art of modeling’

‘Process modeling is an art’ (vrij vertaald: procesmodelleren is een kunst). Voor velen van jullie zal dit een eerste keer zijn dat je een procesmodel maakt. Gelukkig zijn er allerlei tips & trucs om je te behoeden voor de verschillende valkuilen, zoals aangegeven in het begin van dit hoofdstuk.

Een eenvoudig stappenplan wanneer je aan de slag gaat is:

1. Bepaal wie de actoren zijn. Alle actoren in de buitenwereld zijn pools. Alle actoren binnen het te modeleren proces zijn “lanes” die worden gegroepeerd tot één pool.
2. Bepaal het begin en het einde van het proces (de scope van het proces). Wat zijn de mogelijke eindstatussen?
3. Kijk wat je in je ‘top level model’ wil hebben en welke ‘sub levels’ er zijn. In hoofdstuk 5 van Bruce Silver kun je hier nog achtergrondinformatie bij vinden, ook hij reikt een stappenplan aan, waarbij hij ingaat op het begrip ‘decompositie’ en hiërarchie binnen een procesmodel. Werk in ieder geval van grof naar fijn: begin met een hoofdproces om grove brokken te identificeren. Vervolgens kun je dit op subprocesniveau verder detailleren naar taken.
4. Benoem nu per subproces de taken (gebruik daarvoor werkwoorden).
5. Verbind de taken met elkaar in een logische volgorde.
6. Bepaal waar keuzes ontstaan en modelleer deze keuzes in het proces. Beschrijf de uitgangen van de keuzes (guards on arcs).

Verder zijn er nog allerlei ‘styleguides’, en tips die je online kunt vinden, een naslagwerk is <https://camunda.org/bpmn/reference/> Ook het boek van Bruce Silver besteedt in de hoofdstukken 6 en 11 aandacht aan de regels van BPMN. Deze regels kun je zien als een set van afspraken om te komen tot een procesmodel van hoge kwaliteit in de notatiewijze ‘BPMN’. Deze afspraken gelden voor level 1, 2 en 3 elementen, dus voor BIS is maar een deel hiervan relevant.

Daarnaast zijn er zogeheten process modeling guidelines. Deze, in totaal 7 guidelines, kun je zien als een aantal algemene uitgangspunten die je kunt hanteren, en waar we bij de beoordeling ook op zullen letten:

* G1: Use as few elements in the model as possible;
* G2: Minimize the routing paths per element;
* G3: Use one start and one end event;
* G4: Model as structured as possible;
* G5: Avoid OR routing elements;
* G6: Use verb-object activity labels;
* G7: Decompose a model with more than 50 elements.

Meer informatie over deze 7 process modeling guidelines kan worden gevonden in de bijlage ‘7PMG’ (achtergrondinformatie, geen toetsstof)

# Requirements

In dit hoofdstuk wordt uitgelegd wat requirements zijn, en hoe requirements worden beschreven.

## Wat zijn requirements?

Er zijn diverse definities van het begrip requirements, afhankelijk van de context waarbinnen het begrip wordt gebruikt. Hoofdstuk 2 uit het boek ‘managing software requirements’ geeft een beeld bij het proces van het beschrijven van requirements, en de context.

Binnen BIS ligt de focus op software requirements, waarbij we twee soorten onderscheiden: functionele en niet-functionele requirements, zoals ook beschreven op <https://nl.wikipedia.org/wiki/Requirement> Een nadere onderverdeling die vaak gebruikt wordt is samengevat in het acroniem FURPS:

F: Functionality

U: Usability

R: Reliability

P: Performance

S: Supportability

<https://nl.wikipedia.org/wiki/FURPS>

## Requirements beschrijven

Requirements kunnen in natuurlijke taal worden beschreven en in conceptuele modellen. Het makkelijkste is om requirements in natuurlijke taal te beschrijven. Het voordeel van natuurlijke taal is dat deze door iedereen makkelijk te lezen is. Het nadeel is dat natuurlijke taal niet altijd eenduidig (ambigue) is.

*Een voorbeeld van ambiguïteit:*

*De man slaat de hond met de stok.*

Waarmee slaat de man de hond?

Zie voor het antwoord en uitleg (en meer voorbeelden): <https://nl.wikipedia.org/wiki/Ambigu%C3%AFteit>

Een belangrijk voordeel van conceptuele modellen is dat informatie compact wordt weergegeven. Conceptuele modellen zijn altijd maar voor één uitleg vatbaar. Een nadeel van conceptuele modellen is dat deze moeilijk leesbaar kunnen zijn voor de gebruiker.

***Natuurlijke taal***

Requirements die in natuurlijke taal geschreven zijn moeten SMART zijn. SMART staat voor:

* Specific
* Measurable
* Attainable (Achievable, Actionable, Appropriate)
* Realistic
* Time-bound (Timely, Traceable)

Zie voor toelichting en voorbeelden:

https://jessica80304.wordpress.com/2008/08/04/smart-requirements/

User stories is een techniek om requirements in natuurlijke taal te beschrijven.

Zie voor toelichting en voorbeelden:

* <https://www.reaco.nl/blog/user-stories/>
* Functioneel Ontwerp Pizza Koerier hoofdstuk 4.2 user requirements

**Conceptuele modellen**

Een conceptueel model voor het beschrijven van de requirements zijn Use Cases, zie ook hoofdstuk 4.

# Wireframe’s

Wireframe’s heb je in vele soorten en smaken en leggen de focus op drie zaken:

1. Navigatie (hoe is de navigatie geregeld van het ene naar het andere scherm)
2. Hoe ziet de layout eruit?
3. Welke content wordt op de schermen geplaatst?

Wireframe’s tonen een logische ordening. Een wireframe is geen grafisch ontwerp waarin bijvoorbeeld lettertype, kleur en stijl al definitief is uitgewerkt.

Houd bij het maken van wireframe’s rekening met de opdrachtgever. Wat is voor hem of haar herkenbaar?

Soorten wireframes zijn:

- sketches 🡪 experimenteel, goed voor het vragen van feedback

- low fidelity 🡪 content en flow

- high fidelity 🡪 gedetailleerde wireframe’s die functionaliteit en gedrag goed weergeven.

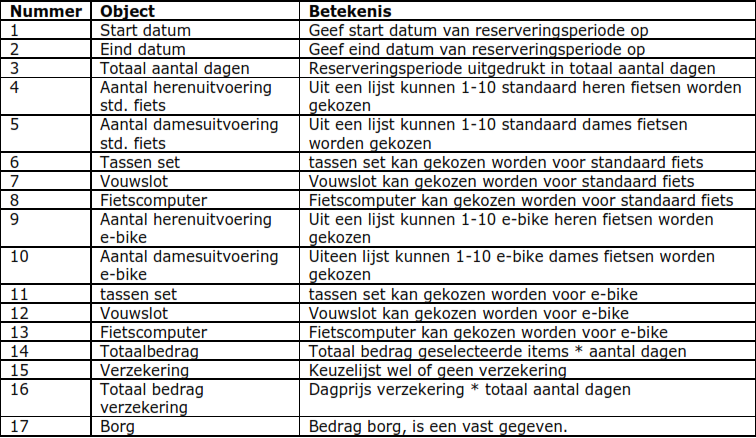
Een wireframe moet worden gedocumenteerd (een plaatje alleen is niet genoeg).

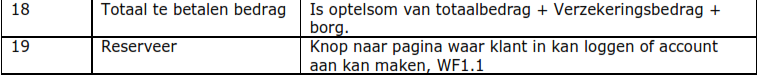
Een stappenplan voor het maken van wireframes is te vinden op: <https://webdesign.tutsplus.com/articles/a-beginners-guide-to-wireframing--webdesign-7399>

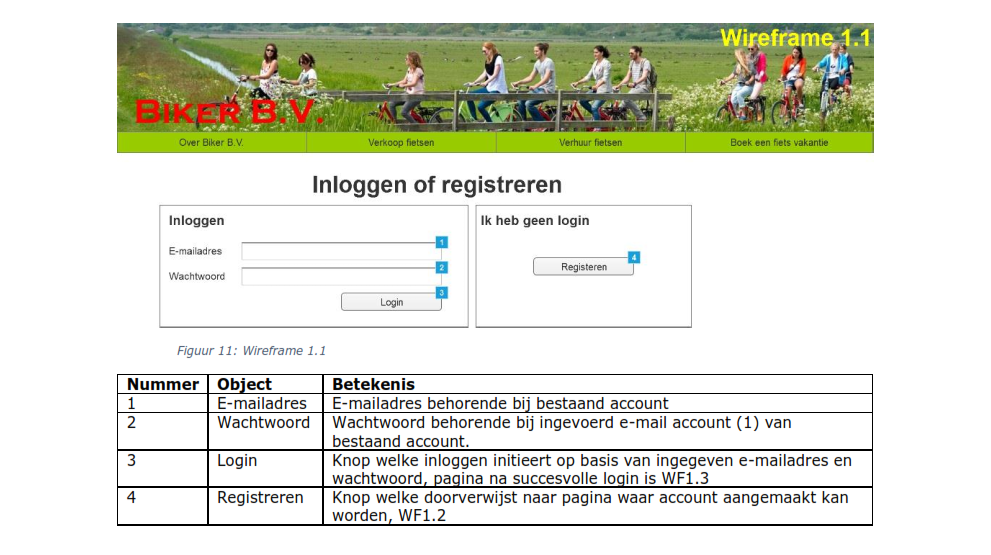
Voor de uitwerking van het functioneel ontwerp in deze module mag je zelf kiezen of je low of high fidelity wireframes uitwerkt. Het belangrijkst is dat je de flow van de interactie tussen gebruiker en systeem goed weergeeft, en dat deze aansluit bij de interactie die je in de fully dressed use cases hebt beschreven. Wanneer je kiest voor high fidelity use cases, verlies je dan niet in de details van de opmaak van het scherm, maar presenteer het zo dat het aantrekkelijk is voor de opdrachtgever, maar niet afleidt van het doel van het functioneel ontwerp. Wanneer je kiest voor low fidelity use cases, maak deze dan wel realistisch, hou rekening met bijvoorbeeld toekomstige wensen van de opdrachtgever om een logo op het scherm op te nemen, en met realistische afmetingen van invoervelden.

Onderstaand zijn een aantal high level wireframes weergegeven die zijn gedocumenteerd en waarvan in hoofdstuk 4 een voorbeeld Fully dressed usecase is uitgewerkt. Je ziet zo ook de samenhang tussen wireframes en fully dressed usecases die in een functioneel ontwerp zoals we dat bij BIS maken belangrijk is.

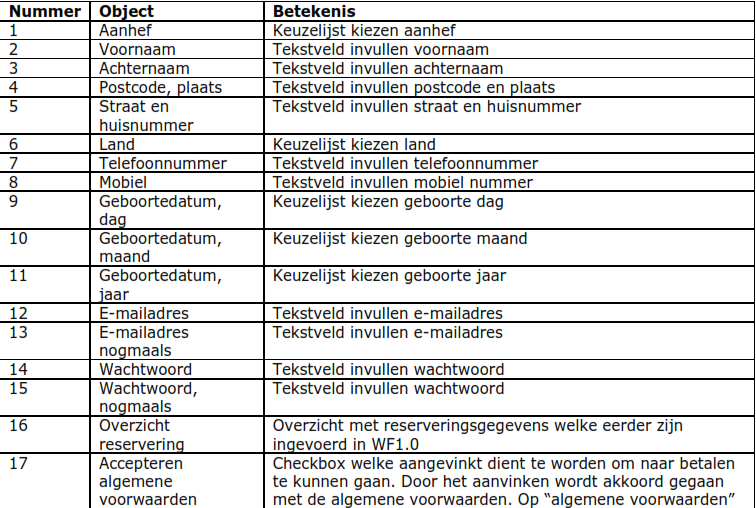


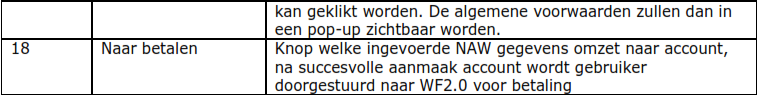




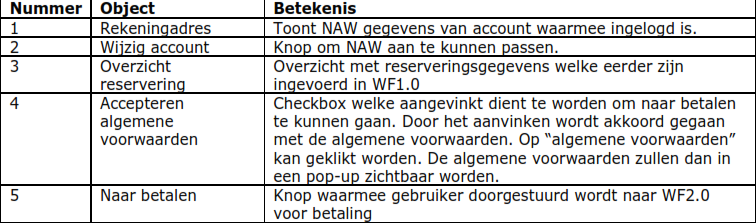












# Use Cases

In de literatuur komen verschillende templates voor het gebruik van use cases voor in meer of minder detail. De volgende templates worden voor deze course gebruikt:

* Brief use case: bestaat uit korte toelichting, die de use case samenvatten. De beschrijving wordt in de fully dressed use case weer gebruikt.
* Fully dressed use case: dit is een volledig ingevulde use case. Het is een formeel document gebaseerd op een gedetailleerd model met velden voor verschillende secties. Dit is de meest gebruikelijke betekenis van het use case concept.

Use cases worden modelmatig gepresenteerd in een Use Case Diagram. In dit diagram worden de relaties tussen de Use Cases en de actor(s) gepresenteerd.

Zoals bij natuurlijke taal komen er ook in UML veel dialecten voor. In zo’ndialect zijn er net iets andere afspraken, aangepaste lay-out of andere naamgeving van toepassing. De UML artefacten van bedrijven, organisaties e.d. zijn dan ook vaak verschillend maar met enige moeite voor ons wel te begrijpen.

Omwille van de leesbaarheid en de eenduidigheid spreken we daarom binnen BIS een aantal zaken af in een Reference Card.

**REFERENCE CARD**

**Use Cases:**

* De naam van een Use Case bevat een werkwoord. En is uniek en duidelijk. De naam is betekenisvol voor de Actor.
* Formuleer alles altijd actief en niet passief, zodat duidelijk is wie welke handelingen doet. Dus niet "Het *wordt* ingevuld", maar "De *actor* vult dit in: ..".
* We nummeren nooit dieper dan drie niveaus in een stappenplan. Als dit nodig blijkt, klopt de complexiteit van de Use Case niet en moet deze gesplitst worden.
* Meerdere pre- of postcondities scheiden we altijd met het woord *'of'* of *'en'*.
* De pre/postcondities zijn relevant voor het systeem. Dus geen dingen als "Er moet een printer aanwezig zijn".
* Het einde van een Use Case geef je aan met het woord "Stop". Dit kan na een (niet geslaagde) voorwaarde zijn of aan het einde van de use case, maar dit laatste is triviaal dus mag weggelaten worden.
* In het Use Case Diagram spreken we van een of meerdere Actors en niet van Acteurs.
* Voor het aanroepen van een secundaire Use Case gebruiken we:   
  "Voer uit **Invoeren NAW**", waarbij 'Invoeren NAW' een secundaire use case is.

Fully dressed Use Cases (ook wel: Use Case Outline / Use Case beschrijving):

* Het stappenplan in een Use Case beschrijving is altijd genummerd.
* We onderscheiden apart de Basic Flow en de Alternative Flow
* Ook de Alternative flows, uitzonderingen en faalscenario’s beschrijven we (indien mogelijk) in 2 kolommen. Maar duidelijk waar deze flows “inhaken” op de Basic Flow.
* We geven links in een tabel de acties van de gebruiker en rechts de acties van het systeem.
* De Basic Flow start altijd met een actie van de gebruiker en eindigt met een actie van het systeem.
* Zijn actief geschreven, zodat duidelijk is wie welke handelingen doet. Dus niet "Het *wordt* ingevoerd", maar "De *actor* voert “bla” in: ..". Vermijdt dus de werkwoorden worden en zijn.
* Trigger moet beschreven zijn in de eerste stap
* Het is altijd duidelijk welke Actor de stap uitvoert.
* Alle mogelijke postcondities zijn toegevoegd.

**Voorbeeld**

Onderstaande Fully Dressed Use Case toont de werking van een reservering. De bijbehorende schermen zijn weergegeven in het hoofdstuk Wireframes.

