Software Analyse Samenvatting

Requirement = eis of vereiste

# Waarom requirements?

Omdat de klant vaak niet weet wat hij wilt, de juiste vragen stellen om zo het probleem volledig vast te stellen en zonder ondubbelzinnigheden. Ook het begrijpelijk maken voor de klant zodat deze tevreden is. Dit zorgt voor de basis van:

* De project planning
* De trade-off
* De risico analyse

Succes projecten zijn gebaseerd op de betrokkenheid van mensen, de support en de duidelijke requirements.

Gefaalde projecten zijn gebaseerd op een slechte betrokkenheid, veranderende requirements en onduidelijke requirements.

Vaak hebben projecten ook last van incomplete requirements, en onrealistische verwachtingen.

Het grootste probleem is hoe een goede requirement op te stellen, bij bedrijven die minder dan 5% hieraan besteden lopen de projecten vaak 80-200% duurder uit.

Wie wilt de requirements weten? -> De Stakeholder!

* Een persoon met interesse of bezorgdheid in een organisatie / project

Requirements hebben dus een directe invloed op het succes van het project

# Wat zijn requirements?

Requirements zijn specificaties van wat er geïmplementeerd moet worden. Ze zijn een beschrijving van hoe een systeem moet werken. Ze zijn dus een beperking op de ontwikkeling van het project.

Requirements zijn condities opgelegd door een gebruiker/stakeholder om een probleem op te lossen en ervoor te zorgen om iet te bereiken.

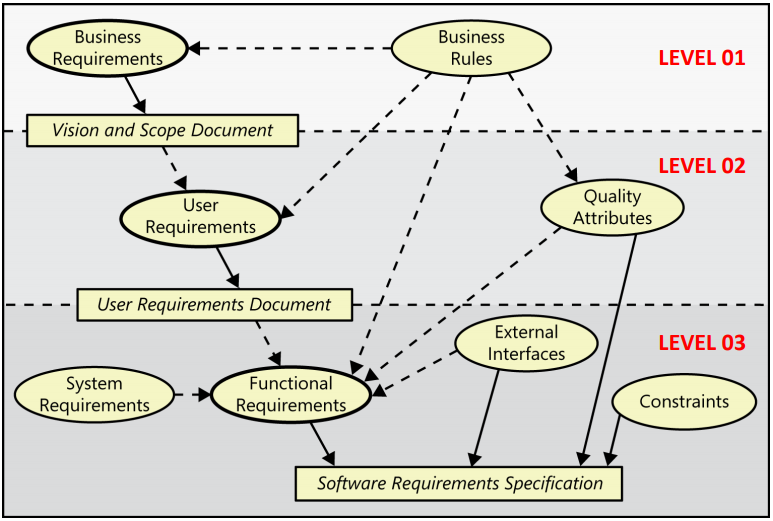
# Level & type requirements

* Functional requirement
  + Functie aangeboden door het systeem
* Quality requirement
  + Kwalitatieve eigenschap van het systeem
* Constraint
  + Een vereiste (functionele eisen en kwaliteitseisen

NON-Functional requirements

# 

* User Requirements
  + De gebruiker zal kunnen
* System Requirements
  + Het product zal
* Detailed Requirements
  + Zijn dit nog requirements?



Wat wilt de sponsor bereiken? Wat moet de gebruiker kunnen en wat moet de software doen?

# Internationale Standaard



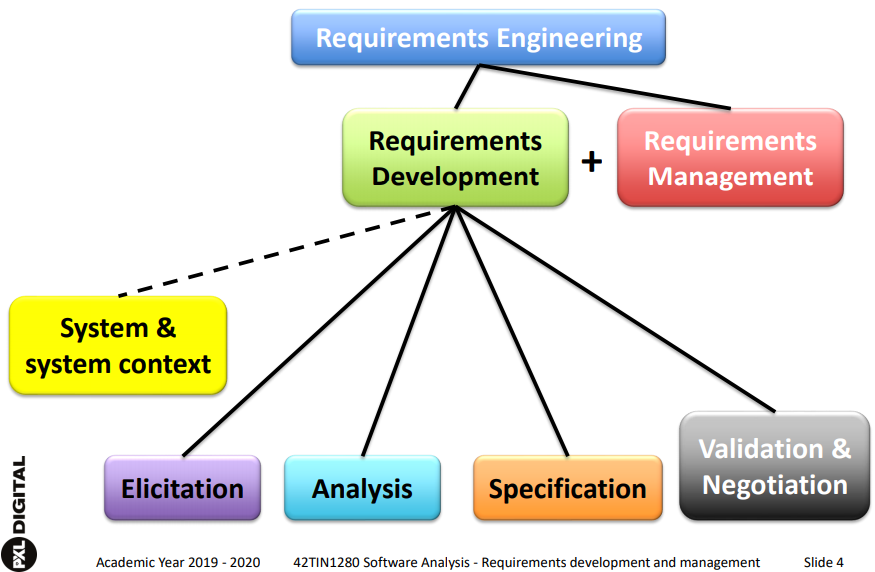
* Portability
* Efficiency
* Reliability
* Functionality
* Usability
* Maintainability
* Security

# Smart Requirements



* S: Een requirement moet duidelijk zijn over wat er gevraagd wordt
* M: Een requirement moet bevestigd worden als er aan voldaan wordt
* A: Een requirement moet haalbaar zijn
* R: Een requirement moet realistisch zijn
* T: Een requirement moet haalbaar zijn binnen een tijdslimiet

# Requirements development & management (RE)



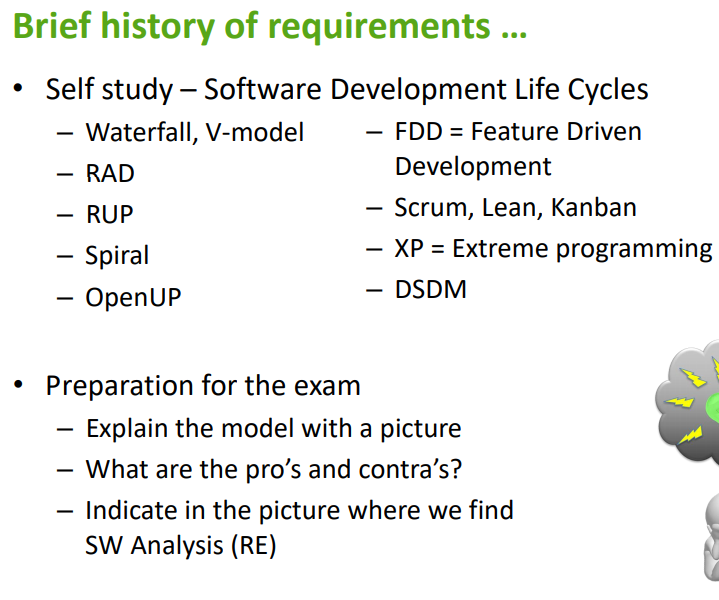
# Requirements Engineering

Een systematische en gedisciplineerde aanpak op de specificatie en management van requirements met de volgende doelstellingen:

Relevantie bepalen en documenteren van de requirements om zo het risico te beperken.

## Vaak voorkomende requirement problemen

Onduidelijk en missende requirements door communicatie problemen of verkeerde veronderstelling van de stakeholders



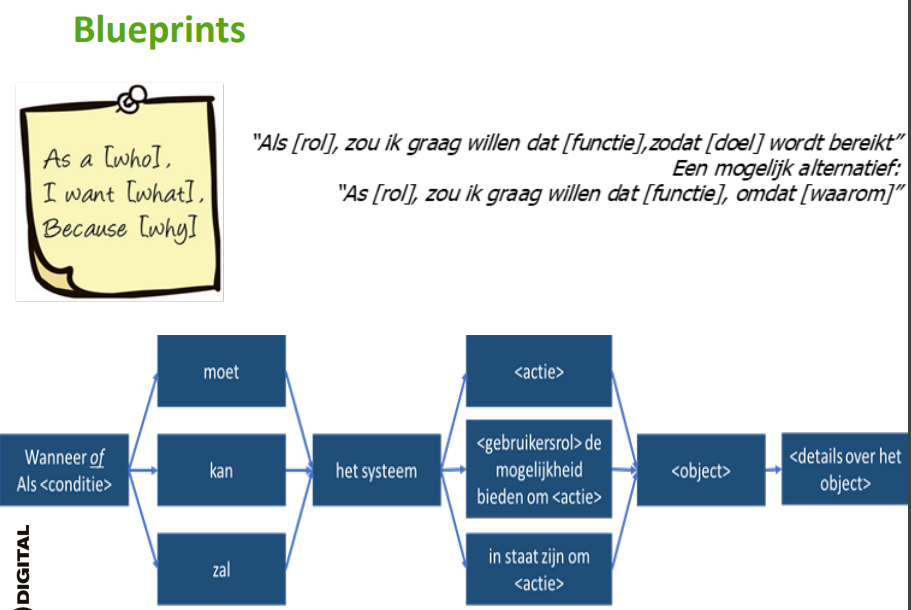
# De business analyst

Een business analyst is een persoon die business analytische activiteiten uitvoert.

* Analyseren van grote hoeveelheid data
* De echte benodigdheden van stakeholders bepalen
* Communicatie analyseren

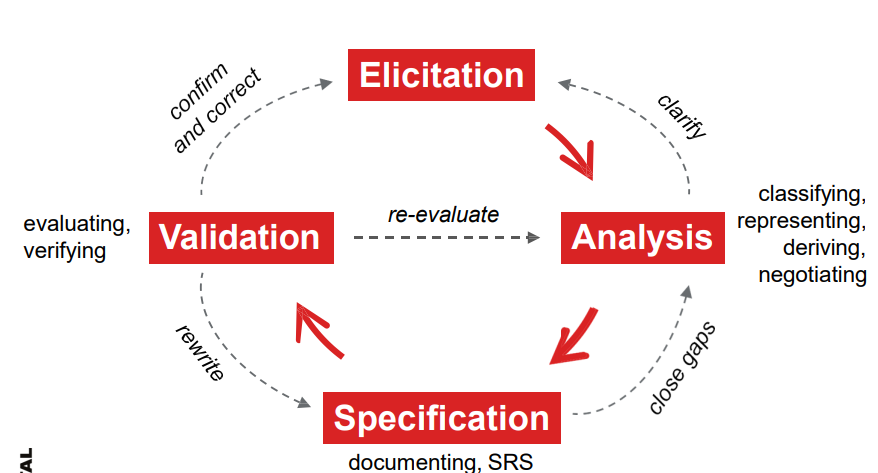
## Bekwaamheden

* Analytisch denken en problemen oplossen
* Geloofwaardige karakteristiek
* Business kennis
* Geschreven en mondelinge communicatie
* Interactieve skills
* Software applicaties.



# Subdisciplines van Requirements Engineering

Requirement Development process framework



# System en system context

“Als je niet kunt beschrijven wat je doet als een proces, Dan weet je niet wat je aan het doen bent.”

Bepalen wie de stakeholders zijn en wat hun doelen zijn. Hoe gaan ze deze doelen bereiken binnen het bereik van het project.

1. Stakeholders bepalen
2. Bereik van het project bepalen
3. De doelen bepalen
4. Hoe de doelen bereiken?

Als je stakeholders vergeet dan vergeet je direct requirements!!!

System context is de bron van de requirements van het systeem. Potentiele aspecten:

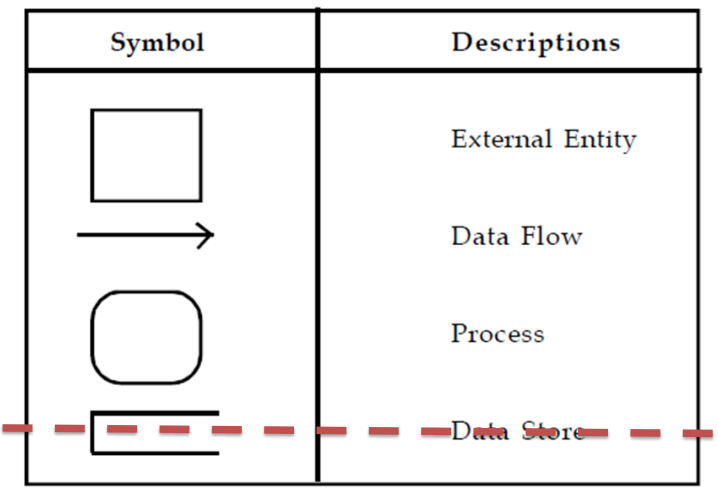
* Personen (Stakeholders)
* Systemen (Technische systemen, software en hardware)
* Processen (Technische, fysische of business processen)
* Evenementen (Technische of fysiek)
* Documenten (Standaards en systeem documentaties)

## Documentatie

* **Context diagrammen**
* **Business use case diagrammen**
* **Domein modellen**

Een context diagram toont welke data er vloeit tussen de buitenwereld en het informatie systeem. Ook beschrijft het de grenzen

wat het systeem moet zijn en niet wat het systeem moet hebben.



# Externe entiteit

* + Elke entiteit die data voorziet of krijgt van het systeem
* Data flow
  + Geeft de richting van de data aan. Input of output
* Proces
  + Acties uitgevoerd op ingevoerde data om een output te geven
* Data store
  + Geen onderdeel van level 0

### 7 regels!!

1. Pijlen mogen niet kruisen
2. Vierkanten en cirkels moeten namen hebben
3. Datastromen moeten in evenwicht zijn
4. Vierkanten en cirkels kunnen niet dezelfde naam hebben
5. De data flows moeten buiten het diagram
6. Betekenisvolle namen gebruiken
7. Passwoorden en validatie requirements zijn niet toegestaan in een context diagram

# UML = Unified Modeling Language

Dient om een standaard manier te creëren zodat iedereen het systeem visueel begrijpt en weet hoe het werkt. UML modellen zijn grafische voorstellingen in de vorm van diagrammen of informatie systemen.

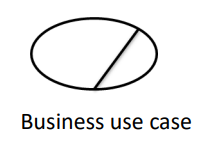
# Why & when business use case models?

* Voor simpele business situaties
  + Een context diagram alleen kan hier al voldoende zijn
* Als dit niet voldoende is
  + Business use case model of proces model opstellen.

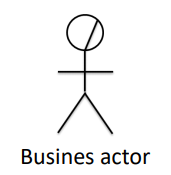
# Business versus system use case

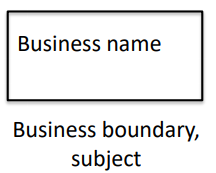
**Business use case:** Een manier waarop een klant of ander geïnteresseerde persoon/systeem gebruik kan maken van de business om zo het resultaat te krijgen wat zij willen. Alle activiteiten die een klant of andere actor wilt kunnen doen moet aanwezig zijn om een compleet proces te vormen van zijn standpunt.

**System use case:** Een manier waarop een klant of computer systeem gebruik kan maken van het systeem om zo het resultaat te krijgen wat zij willen.

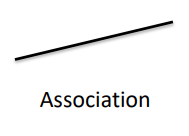
Beschrijft een **business proces** via inter(acties) tussen een actor en een werker die wilt zorgen dat de actor krijgt wat hij wilt.

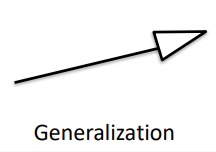
De **Business use case** beschrijft een proces dat waarde biedt aan de actor en beschrijft wat het proces doet.

Een actor representeert een business rol dat samenwerkt met het business proces of de omgeving

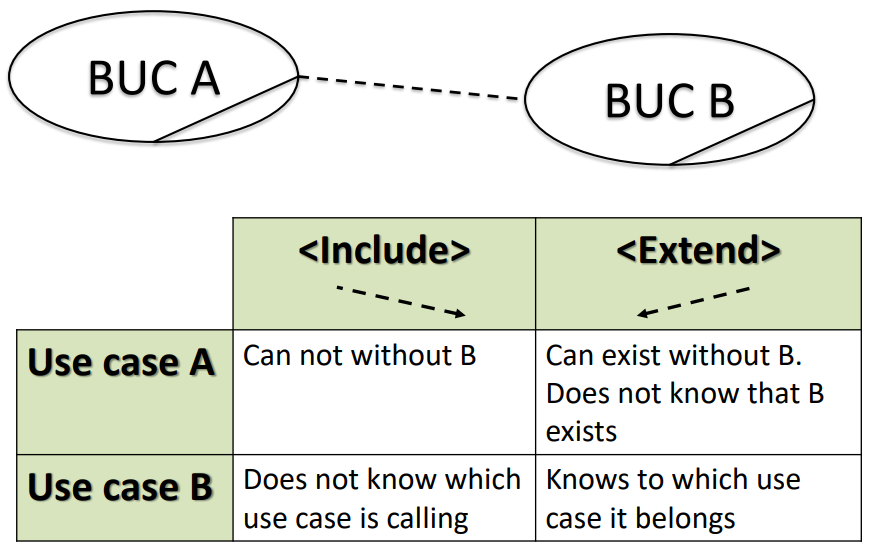
Een onderwerp van een use case definieert en vertegenwoordigt de grenzen van het bedrijf

Associatie is een relatie tussen classificaties

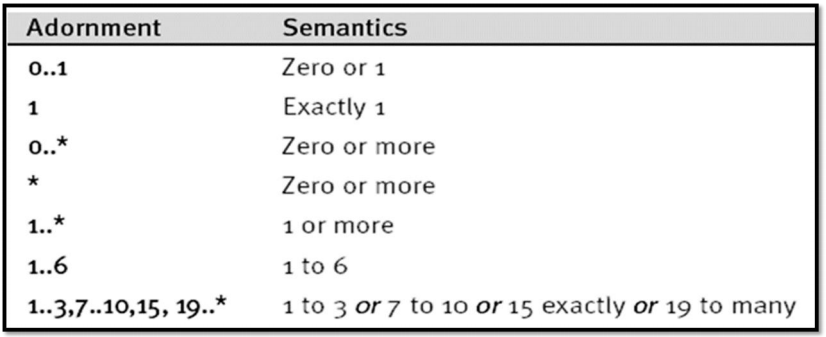




Een generalisatierelatie is een relatie waarin het modelelement (het kind) is gebaseerd op een ander modelelement (de ouder). Het kind ontvangt alle attributen, bewerkingen en relaties die zijn gedefinieerd in het bovenliggende item.



Multipliciteiten

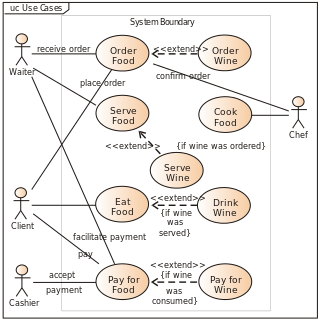


# Stappenplan

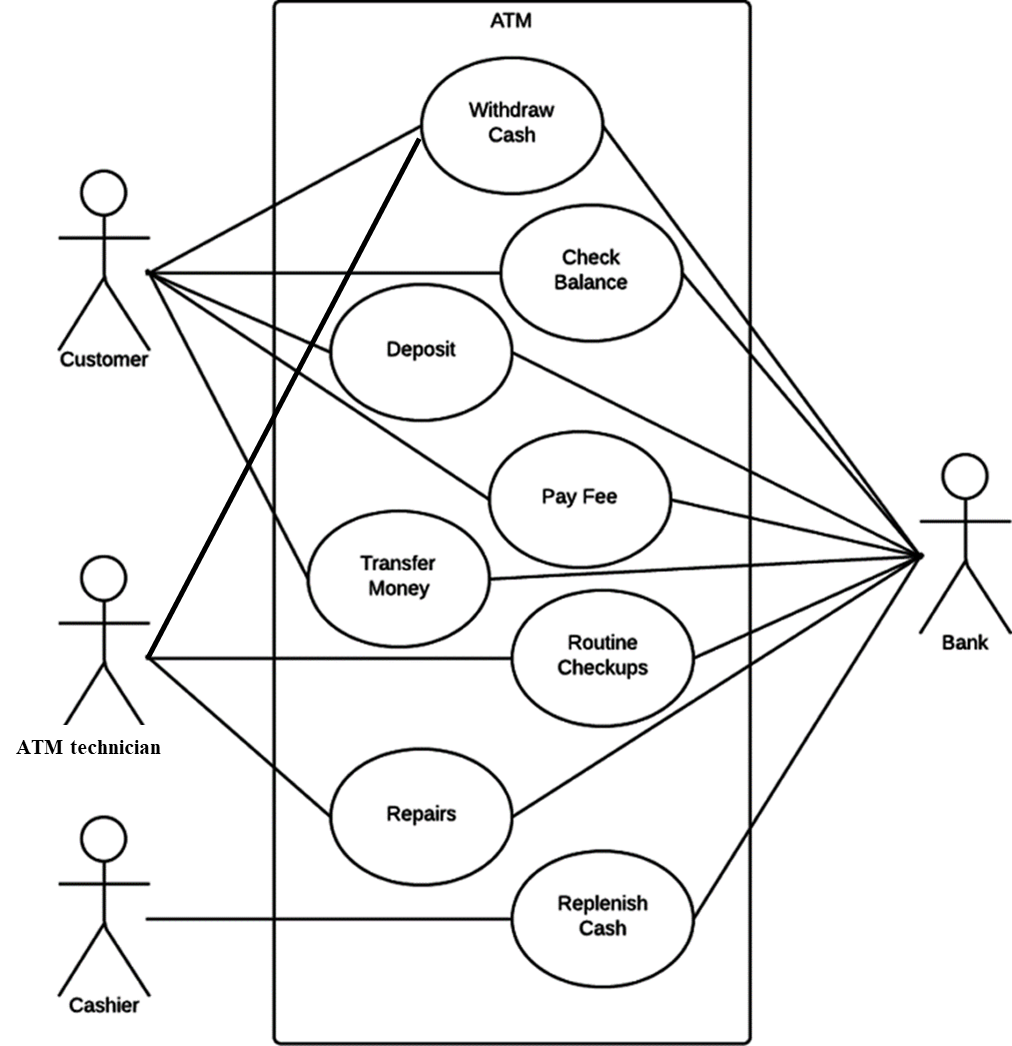
1. Context Diagram en nuttige informatie
2. Identificeer de actoren -> iemand of iets dat interactie heeft met het systeem of het gebruikt om een doel te bereiken
3. Leg het doel vast -> wat willen we doen en bereiken
4. Leg de pre-condities vast -> Wat moet er gebeuren voor de Use Case start
5. Leg de post-condities vast -> Wat is het resultaat of het proces van deze Use Case.
6. Beschrijf de flow -> Resultaat of het succes van de uitkomst van de Use Case
7. Beschrijf de verwachtingen -> Dingen die miss kunnen gaan, wat als het mis gaat
8. Beschrijf alternatieve flows -> variaties van de main flow, soms, misschien & hangt er vanaf

## BUC -> Heeft geen systeem (werknemers die die taak uitvoeren)

## SUC -> komt later in het BUC



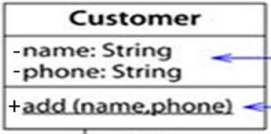
# Bij een System Use Case gaat alles naar het systeem.



# System & System Context – Domain Model

## Definities

**Object:** Een object is een zelfstandige entiteit met een goed gedefinieerde kenmerken (eigenschappen met attributen) en gedragen (activiteiten)



## Domein model

Belangrijkste type objecten in een systeem. Beschrijft de objecten en hoe ze samen hangen met elkaar.

Voorbeeld: Het uitlenen van een boek

Het domein bestaat uit alle dingen, data & regels dat te maken heeft met het uitlenen van een boek

## Karaktereigenschappen en voordelen van een domein model

Toont de associatie en relaties tussen concepten en geeft informatie visueel weer van de verschillende concepten. Hierdoor kunnen ze het probleem van het domein makkelijker oplossen.

## Class Diagram

Een diagram dat de structuur van het systeem weergeeft door klassen, attributen en operaties en de relaties tussen de verschillende klassen.

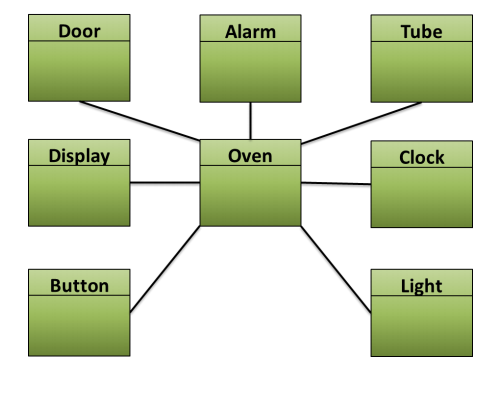
# The process of domain modeling

## Why domain modeling?

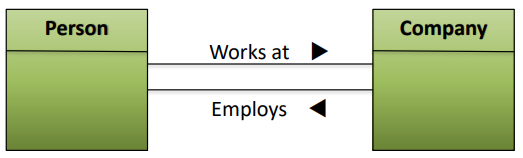
* Je kent het domein niet zo goed
* Je wilt de key concepten niet vergeten
* Je wilt overeenkomen met bepaalde termen

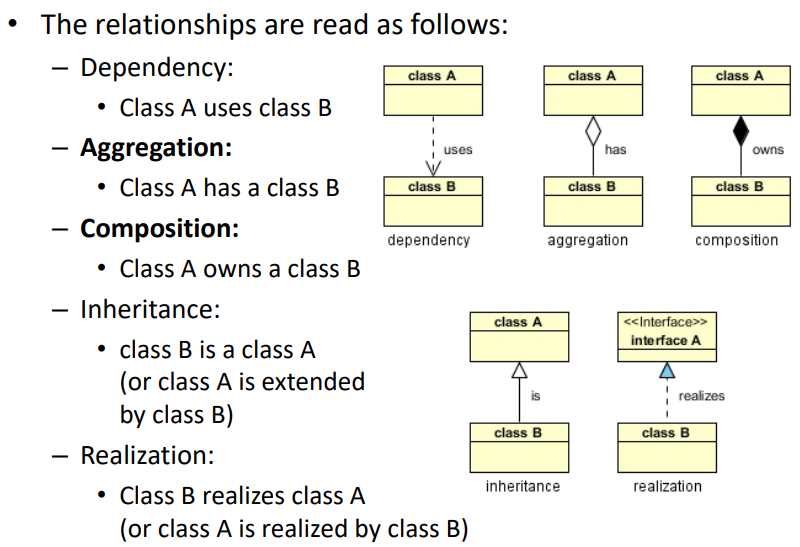
# General steps in domain modeling

1. Maak een probleem statement voor het gebruikte systeem
2. Identificeer de concepten (de klassen & objecten)
   1. ONDERLIJN DE VOLGENDE:
      1. Zelfstandige naamwoorden) -> woorden achter “de”, “het” & “een”
      2. Werkwoorden
      3. Bijvoeglijk naamwoord -> woorden die meer vertellen of het ZN
3. Maak een bekende vocabulaire, woordenboek en woordenlijst
   1. Alfabetische lijst
   2. Tel hoe vaak ze voorkomen
   3. Maak een woordenlijst van de termen -> domein klassen
   4. Maak het eerste domein klassen diagram

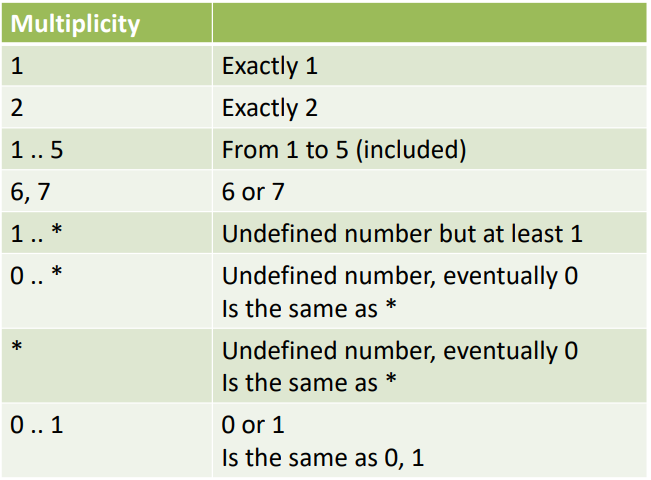


1. Identificeer de associaties tussen de verschillende concepten





1. Geef de concepten attributen
2. Controleer op multipliciteiten en voeg deze toe aan het domein model



1. Herhaal en verfijn het model

**EEN HEEL BELANGRIJK VOORBEELD:**



Elk schaakbord bestaat uit 64 velden

Elk veld hoort bij precies één schaakbord

# Natuurlijke taal

**Elke stakeholder kan het "gemakkelijk" begrijpen ... maar is vaak dubbelzinnig en interpreteerbaar**

Voorbeelden:

* “The chicken is ready to eat” Is de kip klaar om te eten of gaat de kip eten
* “They are hunting dogs” Zijn de honden aan het jagen of jagen ze op de honden
* “Wie wast tuur” Wie is tuur aan het wassen, welke persoon wast tuur

**Wat wordt er gezegd? En wat wordt er bedoeld?**

# Meest relevante voorkomens

* Normalisatie
  + “After a system breakdown a restart shall automatically be initiated”
    - Wat kan er leiden tot een breakdown
    - Wat gebeurt er bij een restart
* Naamwoorden zonder referentie
  + “The data will be shown on the terminal to the user”
    - Welke data, welke terminal en welke gebruiker?
    - Wel juist: “The system shall show the billing data on the terminal where the registered user is signed on”
* Universele kwantificatoren
  + “nooit”, “altijd”, “geen” & “iedereen”
* Onvolledig gespecificeerde voorwaarden
  + “Als”, “wanneer” & “hangt af van”
* Onvolledig gespecificeerde proceswoorden
  + “To transfer”: from where, where to?
  + Passieve zinnen -> actieve zinnen

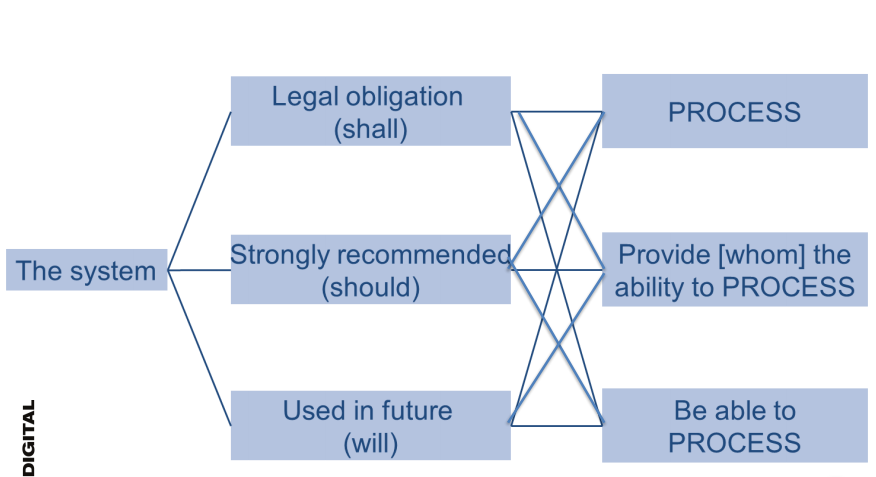
# Dealing with transformations

1. Bepaal wettelijke verplichting / relevantie

➔ werkwoordconventie

* Shall (Een erkende eis)
* Should (aanbeveling of advies)
* Must (wettelijke of regelgevende vereisten)
* Will (vereiste geacht buiten deindividuele reikwijdte van h het systeem)

1. Bepaal de kern van de eis
2. Karakteriseer de activiteit



### De eerste 3 stappen

Ons geplande systeem moet een afdrukoptie bieden

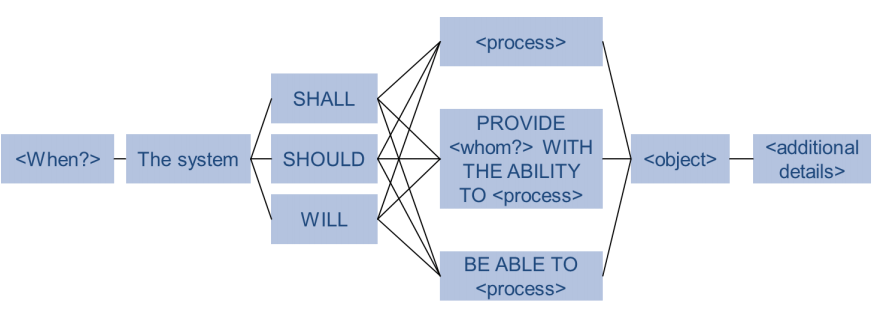
* Onmisbare optie
* “Afdrukken” is het proces woord
* Het systeem geeft de administrator de optie om
* Requirement N1, versie 1:
  + Het systeem biedt de beheerder de mogelijkheid om te printen

1. Voeg objecten toe (“wat”, “waar”

* Requirement N1, versie 2:
  + Het systeem biedt de beheerder de mogelijkheid om de foutmelding af te drukken naar de netwerkprinter

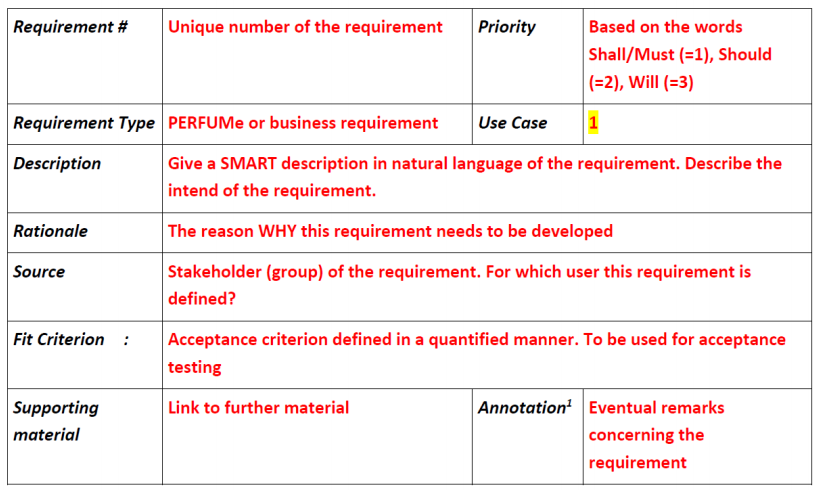
1. Bepaal de omstandigheden

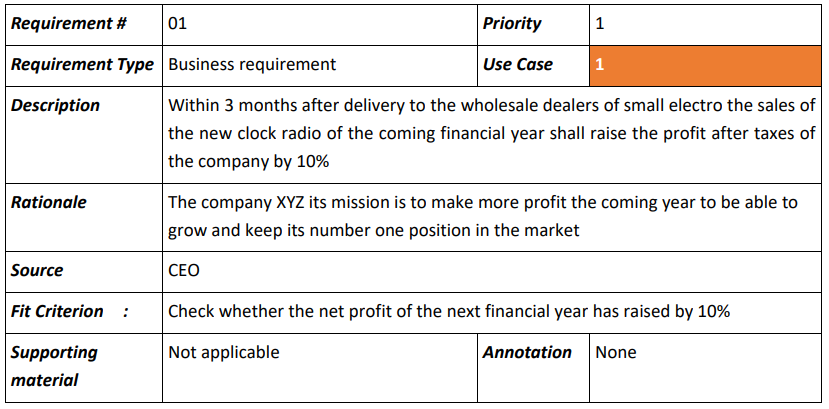
* Requirement N1, versie 3:
  + Als er een foutmelding is gegenereerd, Het systeem biedt de beheerder de mogelijkheid om de foutmelding af te drukken naar de netwerkprinter

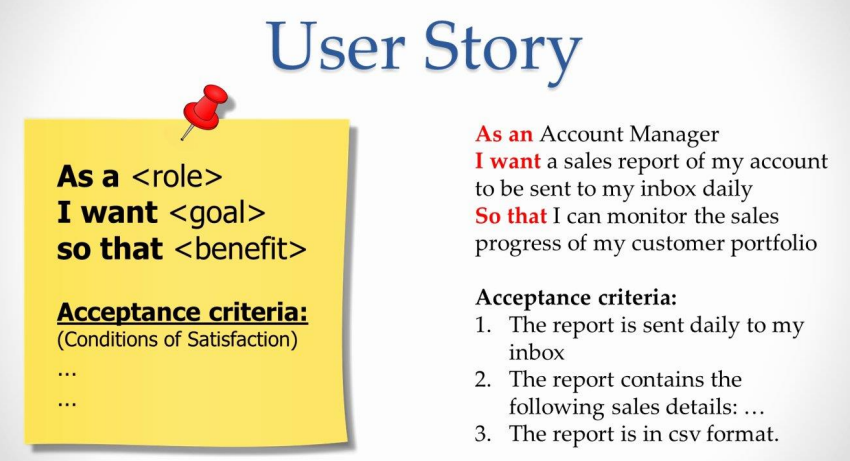


# Regels voor goede requirements

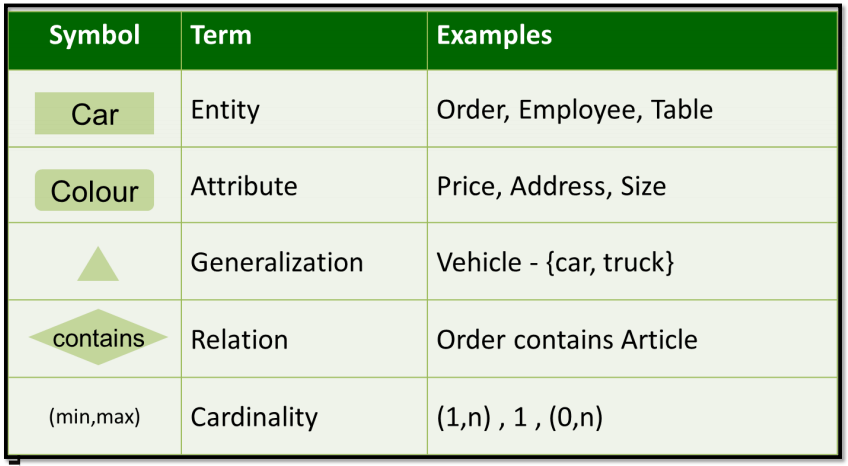
1. Ze moeten ondubbelzinnig duidelijk zijn voor de beoogde lezer.
2. Zij specificeren een maatschaal om het concept te definiëren.
3. Ze zullen complexe concepten opsplitsen in een set van meetbare concepten.
4. Om "relatieve" termen zoals "hoger" te definiëren, zullen ze specificeren op ten minste twee referentiepunten op de gedefinieerde schaal.
5. Ze zullen precies aangeven wanneer een kwaliteitsniveau is bereikt.
6. Ze combineren geen ontwerpideeën in de specificatie van doelstellingen.
7. De procesinvoer of 'bron' (zoals contract, standaard,marketingplan) van de eis moet worden gegeven.

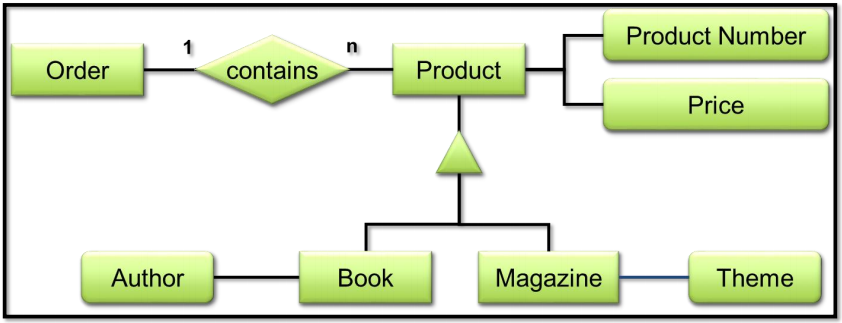


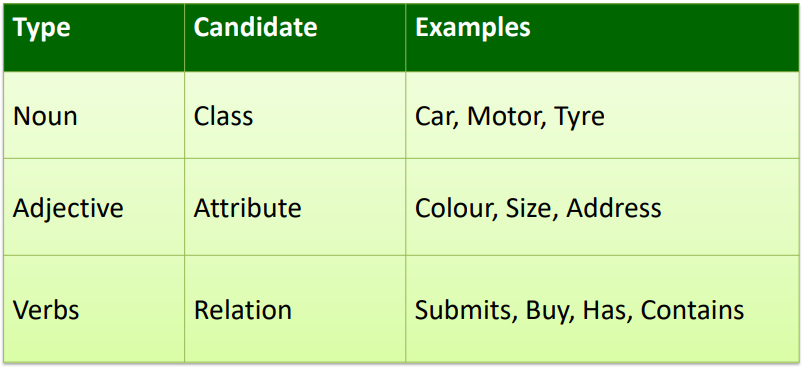




# Entity Relationship Model



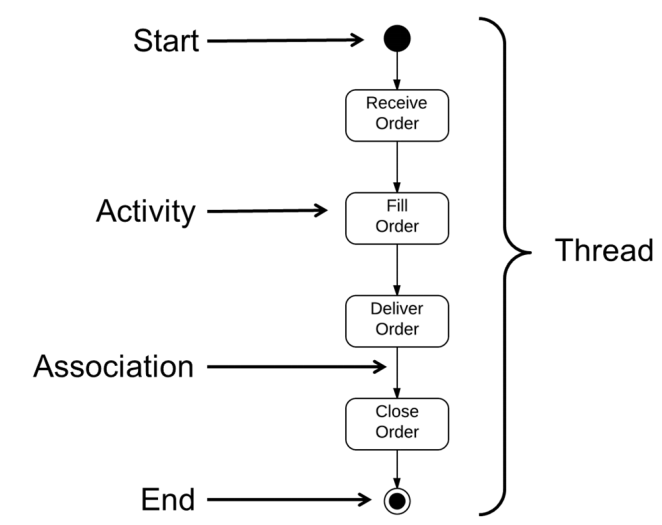




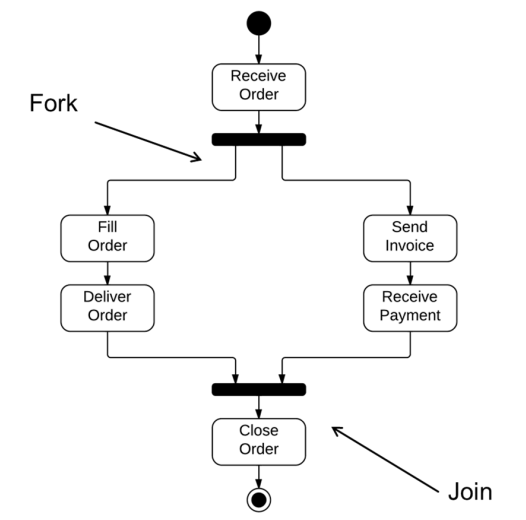
# Activity Diagram

* Activiteiten van de processen en hun relatie
* Volgorde van de activiteiten
* Parallel of conditioneel

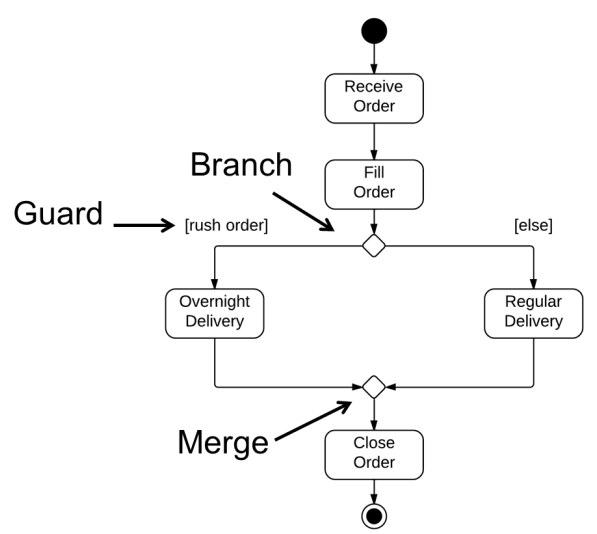
## Activity Diagram - model elements



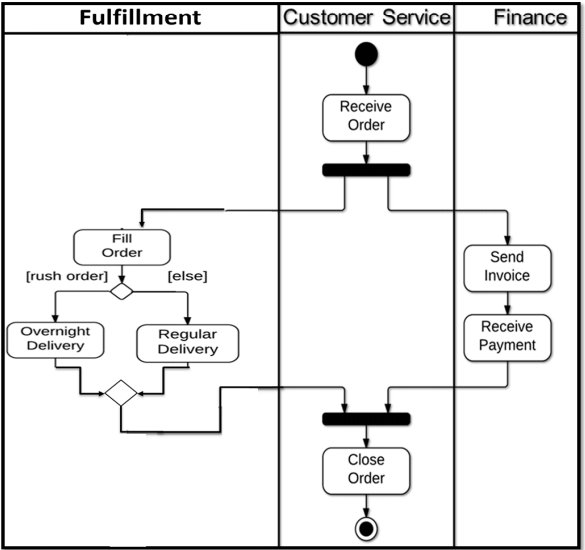
## Activity Diagram - parallel behavior



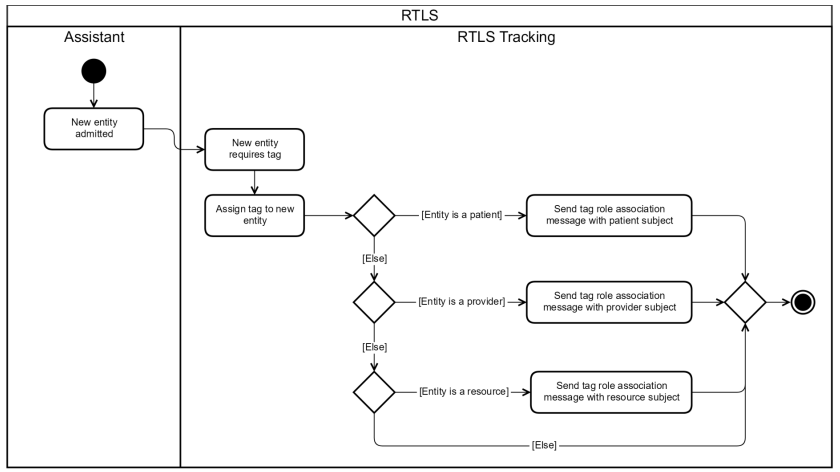
## Activity Diagram - conditional behavior



## Activity Diagram – Swim Lanes



## Activity Diagram – condition ➔ switch case



## Activity Diagram – how to draw

* Een flow chart van de activiteiten uitgevoerd door het systeem
  + **Een activity diagram is niet een exacte flow chart (lanes, branching, parallel & etc)**
* Identificeer de volgende elementen
  + Activiteiten
  + Associaties
  + Condities
  + Beperkingen
* Teken het Activity diagram

## Activity Diagram - pro’s and cons

* Gemakkelijk te leren
* Veel gebruikt
* Kan complexe materie tonen
* Te veel details
* 50% van de activiteiten zou bijna genoeg zijn
* Weinig praktische informatie in UML handleiding

# Models

## System Use Case Diagram (+ Description)

➔ Geeft de interactie tussen de persoon en het systeem weer

## Activity diagram

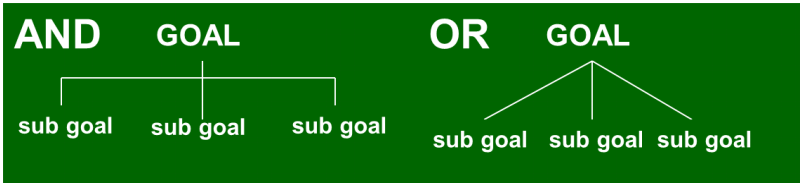
➔Toont de flow van controle

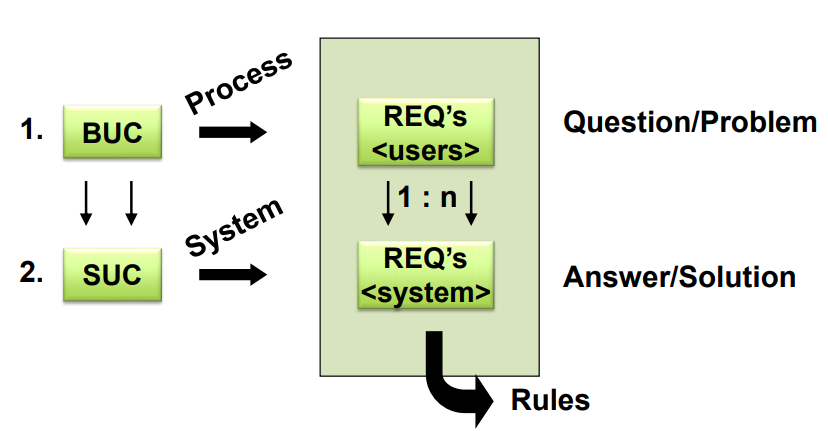
## State chart

➔Toont het gedrag

## Goal Models

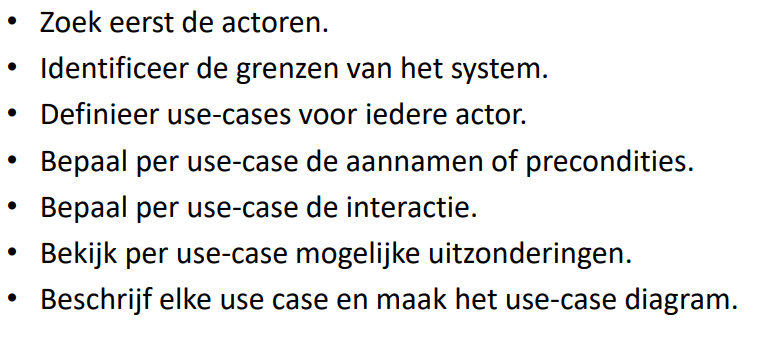
Een doel is een opzettelijke beschrijving van één of meer stakeholders over een gezochte karakteristieke eigenschap van het systeem



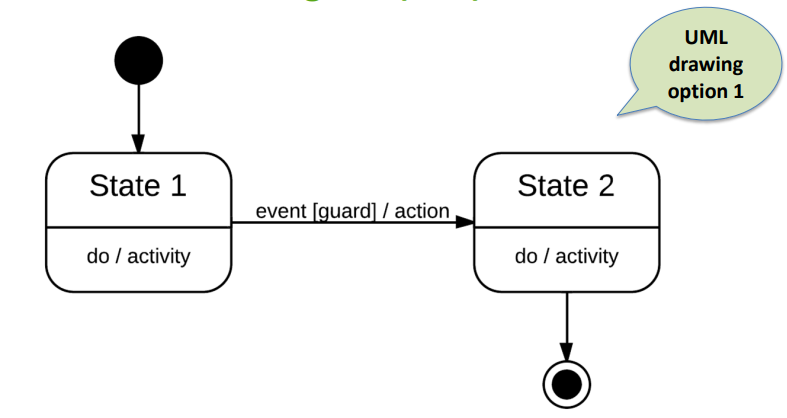


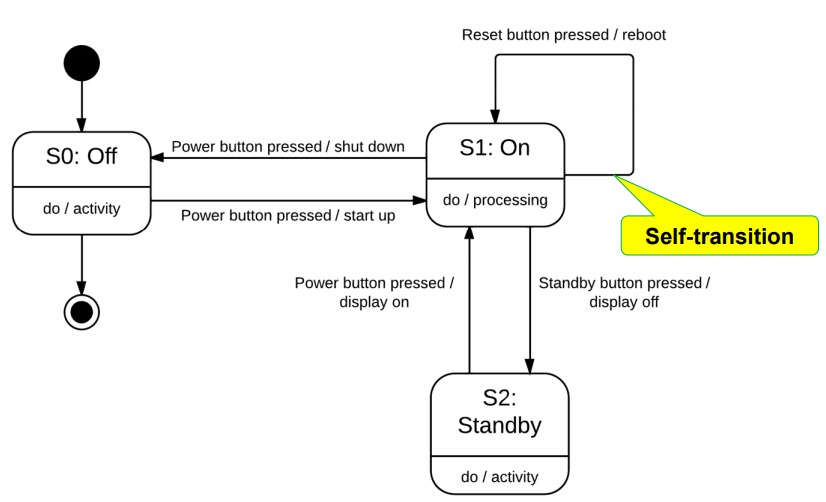
**System Use Cases:** helpen bij het onderzoeken en documenteren van functionaliteit vanuit het perspectief van gebruikers van het systeem

# SUC diagram – eenvoudig stappenplan



# State Transition Diagram (STD)





## STD - how to draw?

1. Definieer status
2. Beschrijf de status
3. Teken transities
4. Definieer events / condities
5. Definieer acties
6. Maak een state transitie tabel

## STD - pro’s and cons

* Beschrijft het gedrag van het systeem
* Toont alle mogelijke statussen van het systeem
* Veranderd van status door event
* Acties zijn korte reacties of transities
* Bruikbaar voor modelling interfaces
* Bruikbaar voor simulatie en testen
* Moeilijk te lezen voor buitenstaanders

