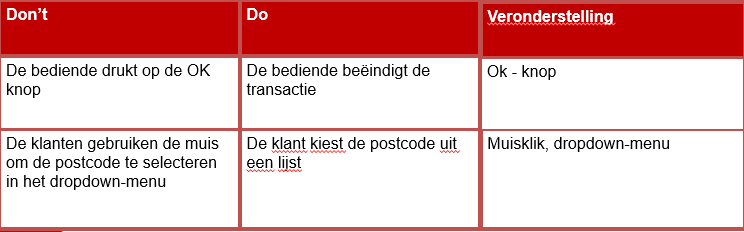
**Software analysis**

Use Case diagrams

* Analyse ( maak Use Case)
* Design (implementeer Use Case)
* Testing (Verifieer Use Case)

**Waarom ?**

* Begrijpbaar voor gebruikers
* Eenvoudig als controle
* Om te scopen
* Geen specifieke taal
* Laat toe om scenario’s uit te werken
* Slechte alternatieven



**Te vermijden :**

* Specifieke personen
* Specifieke afdelingen in organisatie
* UI kenmerken (knoppen, menu’s…)‏
* Veronderstelling over zaken die fysiek gebeuren
* IF THEN ELSE
* Pseudocode

**Wanneer ?**

* Requirements gathering
  + Verstaanbaar voor ontwikkelaar & gebruiker‏
  + Volledig en precies
    - Basis kunnen dienen voor design,testen en acceptatie
* Open
  + Ruimte tot mogelijke oplossing laten
  + Enkel functionele requirements

**1. Functionele en niet-functionele systeemeisen**

**1.1 Functionele systeemeisen**

* Functionaliteit van het systeem beschreven
* Opties / mogelijkheden

**1.2 Niet-functionele systeemeisen**

* Responsetijden, snelheid, geheugengebruik,…
* Bestaande omgeving (software, hardware, databases, standaard systeemcomponenten, standaard gebruikersinterface, …)
* Belangrijke rol bij inschatten benodigde middelen
* Aan bod gekomen in haalbaarheidsstudie
* Spelen rol in latere stadia systeemontwikkeling: applicatie- en implementatiediagrammen

**2. Use Cases in het ontwikkelingstraject**

* Beschrijving van een bepaalde wijze waarop systeem gebruikt kan worden
* Hulpmiddel communicatie gebruikers/opdrachtgevers: niet te formeel
* Use Cases worden later uitgangspunt voor sequentiediagrammen
* **Use Case** als **test case** voor het system

**3.Gebruikersgeoriënteerd werken**

* Systeem beschouwen als zwarte doos
* Systeem = zwarte doos
* Interactie tussen gebruiker en systeem
* Functionele decompositie op hoog niveau

**Concepten**

**Actor**

* Entiteit buiten te analyseren systeem
* Communiceert direct met system
* Is systeem of mens
* 1 mens kan verschillende rollen hebben / actoren zijn‏

**Actor overerving**

* Ik doe alles wat die actor kan en nog meer!‏

**Use Case**

* Beschrijving reeks interacties tussen één of meer actoren en systeem
* Vanuit gezichtspunt van de gebruiker
* Beschreven in natuurlijke taal
* Start altijd bij actor, dan eventueel naar andere

**Relaties**

Actor 🡪 Use Case

* Slechts 1 relatie tussen actor en Use Case
* Actor triggert Use Case

**Include-relatie**

* Tussen 2 Use Cases
* Sub-case is deel van Use Case die op zijn beurt weer Use Case is
* Kan door verschillende Use Cases gebruikt worden
* Meestal bij herbruikbaarheid van vaak voorkomende Use Case
* Target Use Case mag niet afhankelijk zijn van source
* Target Use Case moet op zichzelf kunnen bestaan

**Extends relatie**

* Verbinding tussen Use Case en tweede Use Case
* Tweede Use Case kan **optioneel** worden uitgevoerd binnen eerste Use Case
* Uitbreidingspunten (extension points)‏

Werkwijze

**Stappenplan**

* 1.Identificeer grens systeem en vind actoren
* 2.Vind Use Cases voor iedere actor
* 3.Bepaal (per Use Case) de aannamen
* 4.Bepaal (per Use Case) de interactie
* 5.Bekijk (per Use Case) mogelijke uitzonderingen
* 6.Splits de veelvoorkomende sub-cases uit
* 7.Maak het Use Case diagram

**1. Identificeer de grens van het systeem en vind de actoren**

* Wat behoort wel /niet tot verantwoordelijkheden systeem?
* Zoek mensen / systemen buiten ons systeem die met systeem communiceren
* Bepaal rollen die ze spelen
* 1 rol = 1 actor

**2. Vind Use Cases voor iedere actor**

* Bedenk voor iedere actor functies die hij met systeem kan uitvoeren
* Zoek taken die elke actor m.b.v. systeem moet kunnen uitvoeren
* Use Case is globale beschrijving van systeemgebruik
* Use Cases die op elkaar lijken 🡪 samenvoegen
* Twee woorden met werkwoord: rekening openen, hypotheek afsluiten…
* Hiërarchisch opdelen Use Cases

**3. Bepaal de aannamen**

* Bepaal voorwaarden Use Case uitvoering
* Welke gegevens dienen bekend te zijn, toestand van systeem, …

**4. Bepalen van de interactie**

* Bepaal gewenste interactie tussen de actor en het systeem
* Bepalen gebeurtenis die Use Case initieert en wanneer Use Case is afgelopen
* Pincode invoeren, …
* In natuurlijke taal, normale gang van zaken

**5. Bekijk mogelijke uitzonderingen**

* Apart bekijken van alle mogelijke uitzonderingen, die loop Use Case direct beïnvloeden

**6. Splits veelvoorkomende sub-cases uit**

* Zoeken naar delen Use Cases die identiek zijn. Uit bestaande Use Cases halen en sub-cases definiëren

**7. Maak Use Case diagram**