

Analyse

Kwaliteitsvolle ICT projecten opleveren

- Tijd
- Communicatie: Team-verband
- Geld

Meer mensen betekent niet betere software. Communicatie neemt ook toe waardoor communicatie moeilijker wordt. Neem polynomiaal toe.

Tamme Problemen	Wicked Problemen
(Goed gedefinieerd + makkelijke oplossing)	(Moeilijke oplossing)
Budget op	Weten wat de klant wilt
Tijd op	
Werknemer ziek	
Netwerk probleem	
Hardware	
Eisen v.d. klant wijzigen	

Software ontwikkelingsproces

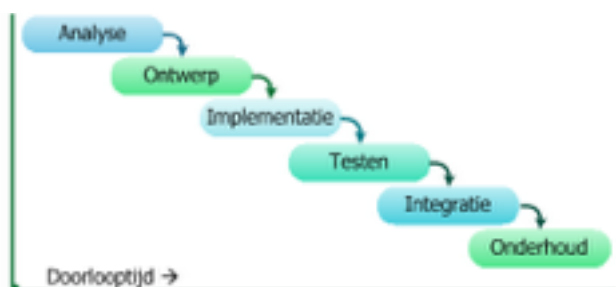
Een kader dat vastlegt hoe een softwareproject wordt aangepakt.

Een methode om de activiteiten in verband met creatie, oplevering en onderhoud van softwaresystemen te organiseren.

Watervalmethode

Opeenvolgend proces.

1. **Analyse:** Requirements achterhalen. Taal verstaanbaar voor klant & programmeur
2. **Ontwerp:** Onderdelen van het te bouwen systeem (Classes & Objects)
3. **Implementatie:** Coderen & organiseren, unit testen schrijven
4. **Testen:** Requirements testen (use cases)
5. **Integratie:** Samen werken met andere systemen (API's)
6. **Onderhoud:** features & optimalisatie.



Robin Malfait

Simpel process (Voor kleinere software projecten)

—> Problemen:

- Weinig flexibiliteit, houdt geen rekening met veranderende eisen van de klant.
- Verlies van informatie: gespecialiseerde personen verlaten het project na hun respectievelijke fase)
- Testen op het einde: fouten worden pas op het einde ontdekt.

Oplossing:

Iteratieve en incrementele ontwikkelmethode

—> Per iteratie een milestone (= documentatie)

Rational Unified Proces (RUP)

Iteratie:

- Software opdelen in time boxes waarbij elke iteratie het proces Analyse, Ontwerp, Implementatie, Testen, Integratie doorlopen en dus uitbreiden.
- We gaan domeinmodel per iteratie (en dus Use Case) opbouwen.

Incrementeel: functionaliteit dat werkt (een werkend geheel)

- Iteratieve, incrementele ontwikkeling
 - think big, develop small: werk in iteraties
- Een iteratie bevat steeds dezelfde activiteiten
- De tijdsbesteding aan iedere activiteit zal gaandeweg tijdens het project veranderen (in het begin meer analyse...)
- Iteraties duren meestal 2 tot 6 weken.

==> **Agile (Een voorbeeld van een RUP)**

Agile manifesto:

Individuals and interactions (Mensen en hun onderlinge interactie)	over (boven)	processes and tools (processen en hulpmiddelen)
working software (werkende software)		comprehensive documentation (allesomvattende documentatie)
customer collaboration (samenwerking met de klant)		contract negotiation (contractonderhandelingen)
responding to change (inspelen op verandering)		following a plan (het volgen van een plan)

OOA/D

Object georiënteerde Analyse/Design

Klasse:

- parameters (attributen)
- methodes

Object:

- Audi A4 (Instantie)

Robin Malfait

Use cases != object georiënteerde analyse/design (maar wordt gebruikt)

UML

Unified Modelling Language (Notatie wijze)

Voordelen:

- Visualisatie
- Communicatie (klant <—> programmeurs)
- Transformatie

Sequentiediagram:

Dynamische levenslijnen



Use Case vs verhaal

verhaal:

- geen structuur
- irrelevante informatie
- warrig verteld

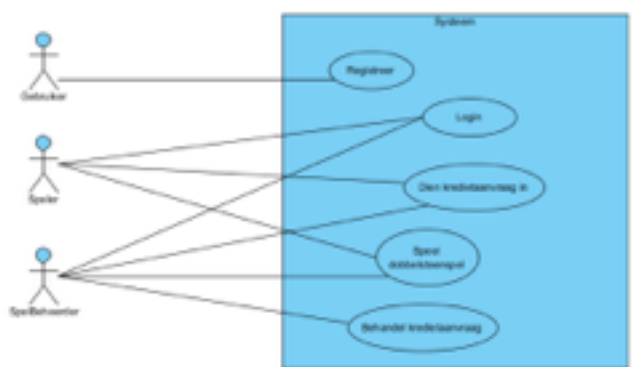
Use Case Diagram:

- UML Diagram
- Alle taken die het systeem uitvoert
- Actor (Persoon **OF** Extern Systeem)
 - De 'starter' is de Primaire Actor

Wat is een Use Case

- tekstuele beschrijving van een taak
- beschrijft gebruik (doel)
- Begrijpbaar voor alle partijen
- Contract
- **Functionele vereisten**

Wat het doet en niet hoe hij het doet



Voorbeeld van een use case:

(Hoe het er uit moet zien)

Use Case: Naam van de use case

Situering: Korte beschrijving

Primaire Actor: de entiteit (belangrijke partij!) die de use case instanties

Pre Conditie:

- Voorwaarden voldaan voor een succes (Voor de use case)
- nodige voorwaarden

Post Conditie: Voorwaarden voldaan na een succes

Normaal verloop: (Main Succes Scenario)

- Lijst van acties die het systeem moet doen om tot een succes te geraken
- Wie / Wat

Alternatieve Verlopen: uitbreidingen/uitzonderingen + referentie naar de stap in het main succes scenario

Domeinregels: Beperkingen/voorwaarden die geldig moeten zijn in het systeem (VB.: Wachtwoord moet min 6 tekens zijn)

Voorbeeld:

Use Case: Registreer

Situering: Een gebruiker wenst zich te registreren

Primaire Actor: een (willekeurige) gebruiker

Pre Conditie: Systeem met databank beschikbaar.

Post Conditie: Actor is geregistreerd en kan nu inloggen

Normaal verloop:

1. De actor wenst zich te registreren
2. Het systeem vraagt naam, voornaam, e-mail, geboortedatum, wachtwoord en bevestiging wachtwoord.
3. De actor geeft de gegevens in
4. Het systeem valideert (Alles verplicht + DR-wachtwoord + DR-email
5. Het systeem maakt de speler aan en geeft de speler een krediet van 5 euro.
6. Het systeem meldt dat de actor is geregistreerd en kan inloggen

Alternatieve Verlopen:

4A. Het systeem detecteert dat niet alle gegevens ingevuld zijn.

1. Het systeem geeft een gepaste melding
2. Terug naar 2

4B. Het systeem detecteert dat er reeds een speler bestaat met opgegeven e-mailadres.

1. Het systeem geeft een gepaste melding
2. Terug naar 2

4C. Het systeem detecteert dat wachtwoord en wachtwoord bevestiging niet gelijk zijn aan elkaar.

1. Het systeem geeft een gepaste melding
2. Terug naar 2

4D. Het systeem detecteert dat actor jonger is dan 18 jaar.

1. Systeem geeft een gepaste melding
2. Use Case eindigt zonder bereiken van de postconditie

4E. Het systeem detecteert dat het e-mailadres niet voldoet aan de regels van een geldig e-mailadres.

1. Het systeem geeft een gepaste melding
2. Terug naar 2

Robin Malfait

4F. Het systeem detecteert dat het wachtwoord niet voldoet aan de regels van een wachtwoord (zie DR-wachtwoord).

1. Het systeem geeft een gepaste melding
2. Terug naar 2

Domeinregels:

DR-wachtwoord:

Wachtwoord bestaat uit minstens 6 tekens, waarvan minstens 2 cijfers.

DR-email

Het e-mailadres moet uniek zijn.

Use case: structuur

Use case = stappenplan voor een bepaalde taak van het systeem.

Een use-case geeft een overzicht van:

- pre condities
- normaal verloop
- alternatieve wegen
- postcondities

FURPS+

FURPS	
Functionality (Features, mogelijkheden, beveiliging)	Functional requirements
Usability (Consistentie, Documentatie)	Non-functional requirements
Reliability (Faalimpact, recoverability, accuracy)	
Performance (Snelheid, Efficiëntie)	
Supportability (Testbaarheid, uitbreidbaarheid)	
+	
Ontwerp requirements	
Implementatie requirements	
Interface requirements	
Fysieke requirements (hardware)	

Includes vs Extends

Includes:

- Verplichte uitvoering van een deeltaak
- Voorbeeld: “withdraw cash” includes “check balance”.
—> Use Cases die lang genoeg zijn om op zichzelf te staan, maar toch verplicht is om een succesvolle use case te bekomen.

Extends:

- Een afwijking van het normale verloop
- Voorbeeld: “withhold card” extends “login” (indien actor 3 x verkeerde code ingeeft)
—> Use Cases die lang genoeg zijn om op zichzelf te staan, die in het alternatieve verloop passen.

Tip: vermijd te moeilijke use cases, vermijd zoveel mogelijk includes & extends

Use Case vs Use Case Scenario:

Use Case: Beschrijving van alle scenario's samen.

Scenario: unieke opeenvolging van acties

Scenario1: 1, 2, 3, 4, 5, 6

Scenario2: 1, 2, 3, 4, 4A, 2, 3, 4, 5, 6

Scenario3: ...

Scenario4: ...

—> Belangrijkste scenario's gaan beschrijven —> basis testen

Hoofdscenario

Het normale verloop is het hoofdscenario.

Eenheid van planning

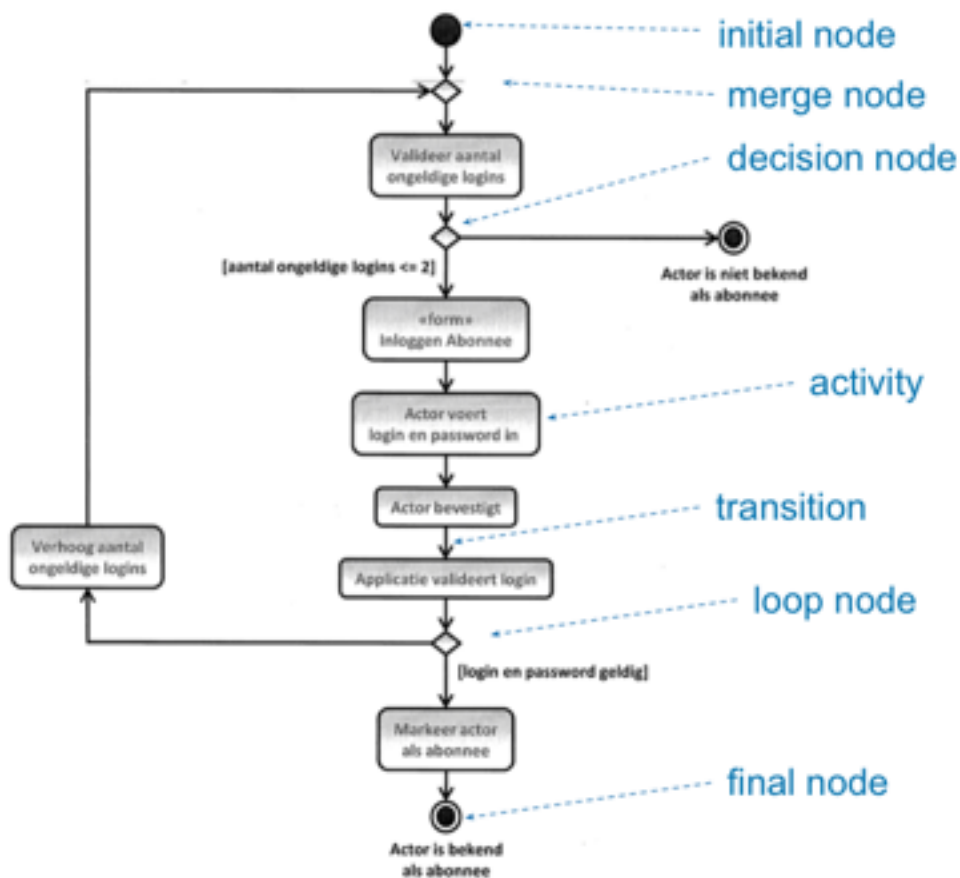
Stap 1: opdelen in use cases

Stap 2: opdelen in juiste prioriteit (sorteren)

Stap 3: Schat ontwikkeltijd

Stap 4: lever incrementeel op (2 - 3 weken)

Activity Diagram



Domeinmodel

Klassendiagram (geen technische specificaties)

Voordeel:

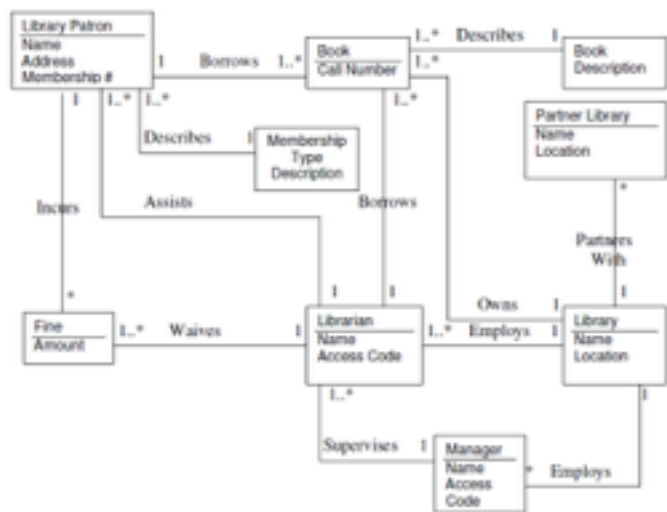
- verstaanbaar voor alle partijen
- eenduidig formuleren van het probleem

“Verhaal”

- > Use Cases (functionele vereisten)
- > domeinmodel -> statische structuur van het programma

Klassendiagram:

- UML - diagram
 - Conceptuele klassen (enkel concepten, geen implementatie)
 - relevante eigenschappen: attributen
- Verbanden tussen instanties zijn associaties.
AssociatieNaam met hoofdletter, werkwoord



Multipliciteiten: 0 1 *

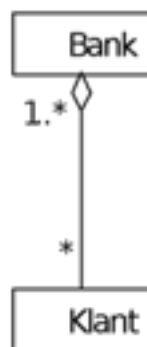
Associatieklasse

Verbonden met de associatie via een streepjeslijn

Aggregatie

een deel/geheel-relatie tussen gelijken
gelijken = het ene object is niet belangrijker dan het andere in de relatie
Belangrijk —> de objecten kunnen onafhankelijk van elkaar bestaan

Voorbeeld:



Compositie

een deel/geheel-relatie tussen niet gelijken
het ene kan niet zonder het andere. Multipliciteit aan de ‘geheel’ kant is altijd 1.



Afhankelijkheden:

Dit wordt afgeraden, als je in A iets veranderd moet je in B ook iets veranderen, als C dan afhankelijk is van B krijg je een hele reeks.

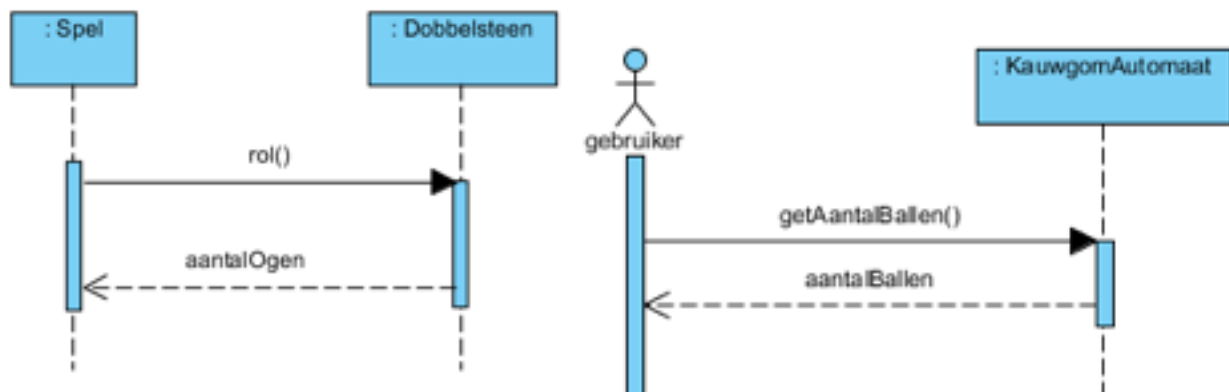
Stappenplan:

1. identificatie van kandidaat klassen
2. selecteren van conceptuele klassen
3. identificatie van associaties
4. identificatie van attributen

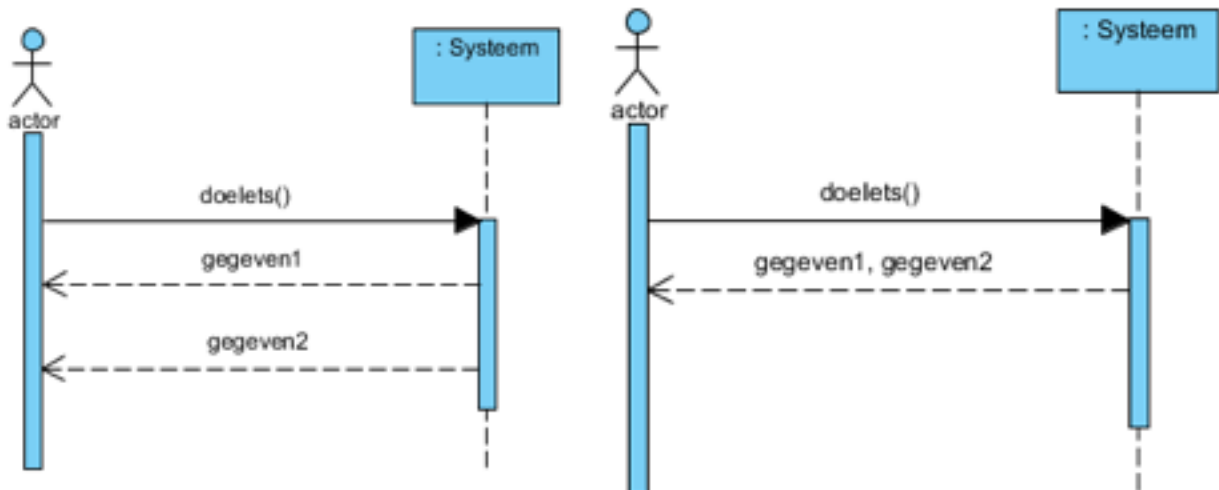
SSD (Systeem Sequentie Diagram (=black box))

UML-diagram, toont aan hoe objecten met elkaar samenwerken in de loop van de tijd om een bepaalde functionaliteit te realiseren.

Voorbeelden van Sequentie Diagrammen:



Voorbeelden van Systeem Sequentie Diagrammen:



Operation Contract

Elke operatie die iets wijzigt in het systeem: maak een contract:

1. Begin met het neerschrijven van de Operation: naam operatie en de parameters
2. Cross References: opsomming van de use cases waarin deze systeemoperatie gebruikt wordt
3. Zijn er niet triviale pre condities? Beschrijf het scenario
4. Leg de postcondities vast.
 1. Condities die bereikt worden nadat de operatie is uitgevoerd.
 2. Beperk de postcondities tot categorieën
 1. creatie van instantie / verwijdering van instantie
 2. creatie van een associatie/verbreking van een associatie
 3. wijziging van attributen
 - Druk uit in de verleden tijd
5. Beschrijf niet hoe de postcondities werden bereikt.

Voorbeeld:

Operation	registreer(naam, voornaam, e-mail, geboortedatum, wachtwoord, bevestiging wachtwoord)
Cross references	Registreer
Preconditions	
Postconditions	<ul style="list-style-type: none"> - Een object van Speler s werd gecreëerd met doorgegeven parameterwaarden. - Attribut krediet van s kreeg de waarde 5.

Sequentiediagramm ⇔ Klassendiagramm

Klassendiagramm:

- statisch
- overzicht klassen met attributen en methoden
- op type-(klasse) niveau
- grondplan van applicatie

Sequentiediagramm:

- dynamisch
- interactie, samenwerking van objecten
- op instantie-(object)niveau
- applicatie in werking

UX (= User eXperience)

Usability:

Effectiviteit, efficiëntie en voldoening waarmee gebruikers een systeem, product of service gebruiken.

UX:

de perceptie en de reacties van de gebruiker ten aanzien van het gebruik (of de intentie tot gebruik) van een systeem, product of service

UX Design:

methode om usability en UX in de ontwikkeling van een systeem, product of service te integreren.

Dit betekent niet dat je de gebruiker de gebruikersinterface laat ontwerpen.

Dit betekent wel dat je de gebruiker een centrale plaats geeft in het volledige ontwikkelingstraject zodat de ontwikkeling meer gebruikersgericht gebeurt. Know your users.

“Everything should be made as simple as possible, but not simpler.” ~ Albert Einstein

Wat maakt iets simpel of complex?

Niet het aantal knopjes of functionaliteiten, maar hoe de gebruiker er mee omgaat of hoe het werkt.


UX Strategy

Lang geleden

Jaren 70

- Ontwikkeling en verbetering van business-processen via software-ontwikkeling.
- Ontwikkeling is gericht op
 - reduceren van tijd
 - reduceren aantal fouten
- Focus voor gebruikers:
 - Snelheid
 - Efficiëntie
- Gebruikers zijn in de eerste plaats technici
—> Weinig nood aan 'usability'-inspanningen


Jaren 80

- PC op de markt (Apple )
- Opkomst van productivity tools (tekstverwerkers, rekenbladen, grafische tools...)
- Gebruikers van software worden minder technisch
—> Belang van usability neemt toe

Jaren 90

- Windows 3.0 bereikt het grote publiek (1990)
- Opkomst van het internet (1994)
- Kritische gebruikers vragen bruikbare systemen

Jaren 2000-2010:

- Tot 21ste eeuw: computer ~ Desktop PC
- Brede waaier van technologische producten
- Apple  leads the way ... again (#WIN)
- Embedded technology: automotive, home, entertainment and automation

Komende jaren:

- Generation Z is here!
- Nooit in een wereld zonder internet geleefd
- Hoge verwachtingen van technologie
- apps vs mobile sites; native vs html5, fluid, responsive, tablets
- NFC, geolocation, augmented reality, voice, exoskeleton, ...
- The Social Web
 - conversatie met de consument
 - gepersonaliseerd internet 'dankzij' vrienden, eigen historiek, online gedrag
- Designing the Product —> Designing the Service
- E-commerce
 - omnichannel experience
 - comfort
- All in the Cloud!
- Tevreden gebruikers
- Relevante technologie
- Innovatieve producten
- Betere service
- Maar ...
 - Excellente user experience aanbieden is vandaag nog steeds voor de meeste bedrijven dansen op een slappe koord...

Frustraties genoeg



Focus is een werkwoord

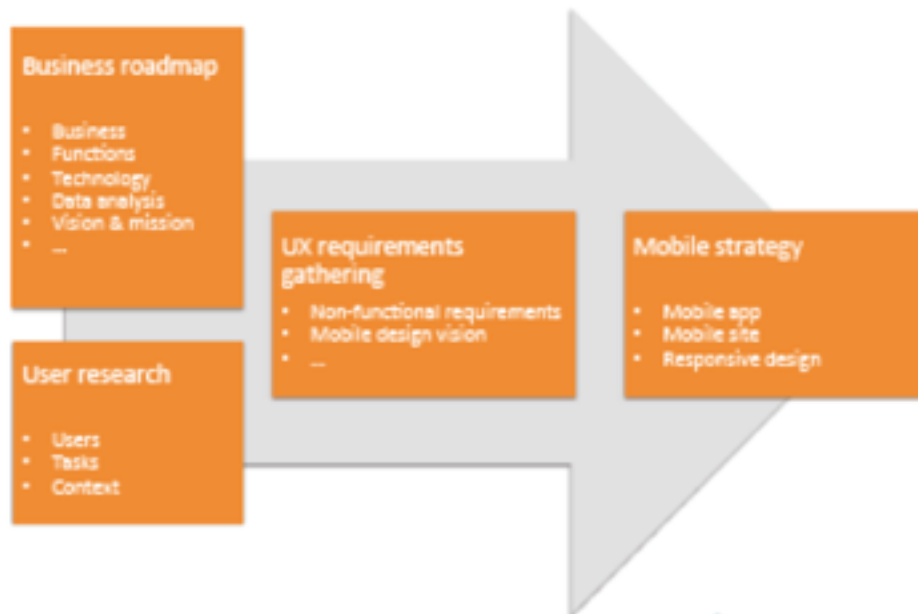
- **Funcities, performantie en betrouwbaarheid:**
 - > onmiddellijk effect op verkoop
 - Nadruk op deze 3
 - Relatief goedkoop
 - Makkelijk om een breed klant segment te bedienen
 - Klanten zijn geneigd om het product met meeste funcities te kopen
- **Nadruk op usability en FJ&D (Fun Joy & Delight):**
 - effect pas duidelijke na verloop van tijd
 - > Minder effect tijdens verkoop, maar positief effect op klanttevredenheid en repeat business
 - wanneer een product faalt voor één of meerdere van de 5 factoren: tevredenheid en verkoop daalt

Focus = succes

- Wanneer de focus op evenwicht in alle factoren ligt, creëer je toegevoegde waarde.
- Gevoel van tevredenheid, 'graag' gebruiken
- Love Mark

Design-visie

- ontwerp-oplossing voor een probleem uit de realiteit van de gebruiker
- gelinkt aan de visie en de missie van de organisatie
- High-level requirement
 - Kader voor ontwerpbeslissingen vergemakkelijkt het verzamelen van requirements
- Resulteert in een user interface concept en een interactiemodel



HoGent

UX Research

Gebruikers, taken en context

- **To build a successful product you must understand:**
 - The people who will use your product (users)
 - The work they do (their goals and tasks)
 - The situations they work in (their physical, social and cultural environments)

De Gebruiker:

- **Begrip verwerven in**
 - welke **doelstellingen** hebben gebruikers?
 - wat **doen** ze om hun doel te bereiken?
 - welke **persoonlijke** kenmerken hebben ze?
 - wat is de **invloed** van voorkennis en ervaring?

Robin Malfait

De taken

- **Gebruiker heeft een doel**
 - een incident coördineren
- **Gebruiker heeft de keuze tussen één of meerdere middelen om doel te bereiken**
 - incident management software, telefoon, radio, ...
 - motorkapoverleg, ...
- **Gebruiker voert één of meerdere taken uit om zijn doel te bereiken:**
 - map tekenen in incident management software tijdens een telefoongesprek

De context

- **Een remote control gebruiken in een verduisterde woonkamer**
- **met een maaidorser rijden**
- **je domotica configureren**
- **geld afhalen**

Requirements verzamelen

- **Requirement**
 - wat een product moet doen
 - welke eigenschap een product moet hebben
- **Functionele requirements**
 - zinvolle acties die met een product uitgevoerd moeten kunnen worden
 - VB.: tijdens een lopend incident moet een tweede incident opgestart kunnen worden
- **Niet functionele requirements**
 - eigenschappen of kwaliteiten die in een product vervat moeten zitten
 - vb.: de pronto remote control moet zonder formele opleiding gebruikt kunnen worden
- **Beperkingen (Constraints)**
 - waar een product aan moet beantwoorden
 - VB.: "My Arval moet bruikbaar zijn op een 3G netwerk"
- **Gebruikers- taak- en contextanalyse maken voornamelijk deel uit van niet-functionele requirements opsporing**

Observaties & interviews

- > Geen vragen stellen in focus groups
- > Geen surveys en questionnaires
- > Geen gesprekken voeren met zogenaamde expert gebruikers
- > Wel inzicht krijgen hoe gebruikers hun dagelijkse taken uitvoeren
 - actief luisteren
 - 100% focus op het gesprek
 - uitleg van de gebruikers parafraseren
 - verwachtingen expliciteren
 - je eigen rol
 - het doel van het interview
 - vb.: geen evaluatie van het functioneren van de gebruiker

Robin Malfait

Participerende observatie

—> Fly-on-the-wall (= call center, operatiekamer, verkeerstoren, ...)

Rollenspel (focus group)

—> alternatief voor de twee vorige

—> minder efficiënt

—> beter in onvoorspelbare situaties, vb.: spoeddienst

User Experience Design

Device Experience

input

- toetsenbord
- muis
- knoppen
- jijzelf (kinect)

output

- tv
- pc scherm
- smart watch

—> Beslissing:

1. 1 oplossing voor alle toestellen —> compromis
2. of een oplossing die zichzelf aanpast aan de verschillen (Responsive design)

posture

1. Hoe de toepassing zich aan de gebruiker **presenteert**
2. Hoeveel **aandacht** een gebruiker aan de toepassing besteedt
3. Hoe de toepassing **reageert** op de acties van de gebruiker

- Lean Back: chillax, in de zetel, filmpje meepakken
- Lean Forward: ow aandacht mannen, excel
- Quick bursts: gsm'ke nekeer zien hoe laat het is

Interface & stijl

platform

Desktop, web, tablet, smartphone, embedded

Platformstijlgidsen: algemene designprincipes (= menus, toolbars, vensters, navigatie, controls)

Interactie via muis, keyboard, gestures, ///

visueel design

heel wat informatie is reeds beschikbaar!

Windows 8.1 (tiles), Apple (perfect)

Concept & navigatie

mentaal model

Hoe moet het product werken (=voorstelling in hun hoofd)

—> gebaseerd op vorige ervaring

—> Niet alle gebruikers hebben eenzelfde mentaal model (designers, users)

Robin Malfait

Conceptueel model

Interface communiceert het conceptuele model van het product
=> het model dat door het product weergegeven wordt

Mentaal/conceptueel model

- **Intuïtieve producten**

- Het conceptuele model matcht met de mentale modellen van de eindgebruikers

- **Innovatieve producten**

- Volledig nieuw conceptueel model dat (altijd/ volledig) afwijkt van het huidige mentale model

- **In plaats van het conceptuele model van het product aan te passen, pas het mentale model van de gebruikers**

- Via training of ondersteuning in UI

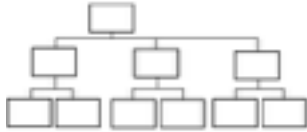
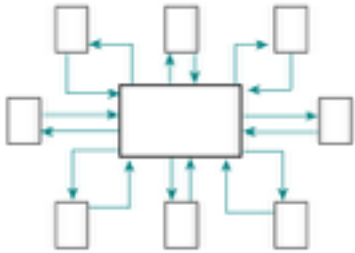
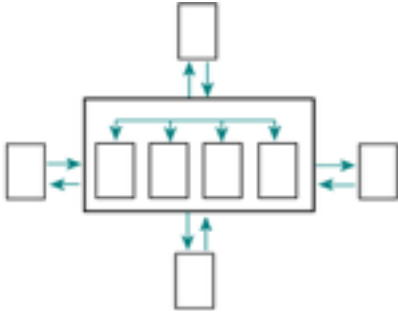

Navigatieplan

- schematische weergave van de opeenvolging van applicatie-onderdelen en de middelen om te navigeren

Doel

- inzicht krijgen in de complexiteit van de gebruikersinterface
- zicht krijgen op de omvang van de applicatie
- communicatie met ontwikkelaars en opdrachtgevers

Navigatie

Structuur	Example	Image
boom	NMBS Automaten	
Ster	Photoshop	
multi-panel	Calendar	
Wizard	Installatie programma's	

Design principles:

1. Ken je gebruiker (geen excel maken voor kinderen)
2. Beperk de mentale belasting (prop het niet vol met knoppen enz)
3. Geef de gebruiker de controle (ok, cancel, undo, history, exits..)
4. Design for error (als er errors zijn, toon geen stack trace, maar een duidelijke melding)
5. Maak het consistent (layout, kleuren, ok cancel vs cancel ok)

Maquette for the win! Zo kan je je layout schetsen zonder enige functionaliteit