Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling met

Titel: Agile development process for usable software
Richting: 2de masterjaar in de informatica - Human Computer Interaction Jaar: 2009

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

LEMMENS, Jan

Datum: 14.12.2009

Agile development process for usable software

Jan Lemmens

promotor :
Prof. dr. Karin CONINX

Eindverhandeling voorgedragen tot het bekomen van de graad
master in de informatica
Human Computer Interaction

Maastricht University

universiteit hasselt

Agile development process for usable software

Jan Lemmens

promotor:
Prof. dr. Karin Coninx

begeleider:
Mieke Haesen

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica human-computer interaction

universiteit hasselt

Abstract. Mede door de steeds uitbreidende mogelijkheden van de computer, is deze alomtegenwoordig geworden in ons dagelijkse leven. We kunnen bijvoorbeeld met behulp van onze mobiele telefoon een voetbalwedstrijd volgen, terwijl we de wasmachine programmeren met behulp van een touch screen. Deze mogelijkheden worden geïmplementeerd in software. Het ontwikkelen van deze software gebeurt tijdens een proces dat Software Engineering wordt genoemd. Software Engineering is een tak van de informatica die zich bezighoudt met het ontwikkelen en onderhouden van software. In deze thesis worden bepaalde onderdelen van Software Engineering behandeld die de focus leggen op het creëren van gebruiksvriendelijke en bruikbare software en dit op een flexibele en snelle manier. Er wordt gestart met een onderzoek naar Agile Software Engineering, gevolgd door een uiteenzetting over User-Centered Design. Een combinatie van deze twee methoden blijkt namelijk een goed uitgangspunt te zijn voor het bouwen van software die zorgt voor een grote klant- en gebruikerstevredenheid. Naast de uitwerking van dit geïntegreerd ontwikkelingsproces, wordt er ook een tool voorgesteld ter ondersteuning van dit proces.

Inhoudsopgave

Deel I:

Introductie

Hoofdstuk 1

Inleiding

1.1 Inleiding

Sinds het ontstaan van de computer in de jaren veertig, is het toepassingsgebied ervan sterk uitgebreid. Steeds meer en meer mensen maken gebruik van computers bij het uitvoeren van hun werkzaamheden in het dagelijkse leven. Het aspect van software speelt hierbij een significante rol. Zonder software is een computer niet bruikbaar. Het doel van software is ondersteuning bieden aan de gebruiker bij het uitoefenen van zijn of haar taken.

Het produceren van software is een complexe bezigheid. Geen enkel softwareproject is hetzelfde. Telkens zijn er andere vereisten, is er een andere doelgroep, wordt er gewerkt met verschillende platformen (bijvoorbeeld een handheld computer of een tablet interface) of is het domein verschillend. Het ontwikkelen van software is een creatieve en innovatieve bezigheid en is dus niet eenvoudig te automatiseren of te herhalen. Verder zijn er een groot aantal partijen betrokken in het software ontwikkelingsproces, zoals de ontwikkelaars, de gebruikers, het management, de klant, mensen uit de grafische sector, enzovoort. De communicatie tussen deze mensen is van essentieel belang voor het bouwen van succesvolle softwareproducten. Om orde te scheppen in dit proces, zijn er een aantal methodologieën in het leven geroepen. Dit zijn raamwerken die ondersteuning en structuur bieden aan het ontwikkelproces.

Het aangehaalde concept van een methodologie, probeert de kans op slagen van een softwareproject te maximaliseren. De hierboven vermelde eigenschappen van software, zijn echter een katalysator voor het falen van een softwareproject. De kans op een onsuccesvol softwareproject is zeer reëel. Volgens IT Cortex [1] zijn minder dan de helft van alle softwareprojecten succesvol. Dit probleem kent vele oorzaken [2], maar de kern van de zaak kan worden samengevat in drie punten (volgorde heeft geen betekenis):

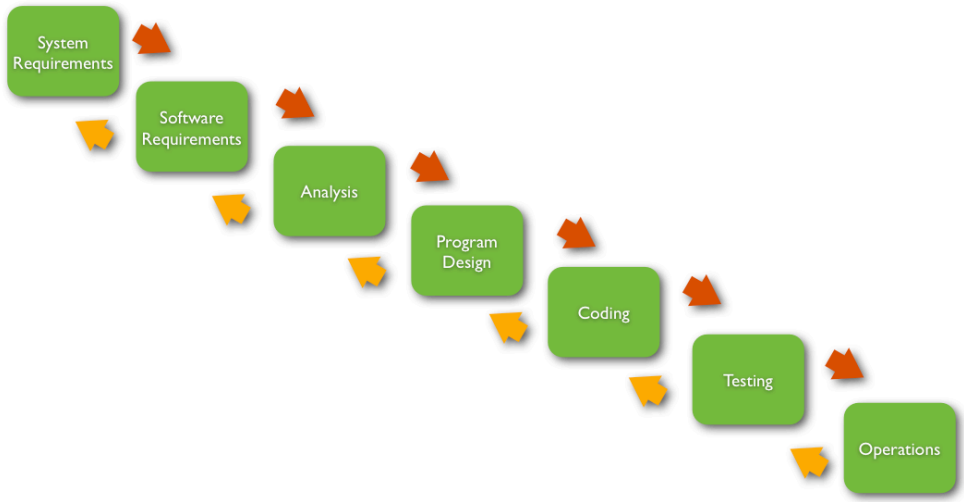
1. De software voldoet niet aan de eisen van de klant.
2. De software wordt niet op tijd afgeleverd.
3. De eindgebruikers kunnen moeilijk of niet overweg met de software.

In deze thesis worden de oorzaken van deze problemen besproken, samen met een reeks methoden en technieken die een oplossing proberen te bieden voor deze kwesties. De principes van “Agile Software Ontwikkeling” en “User-Centered Design” worden onderzocht, met als doel het beschrijven van een methode die bovenstaande problemen aanpakt.

1.2 Probleemstelling

Het Waterfall model presenteert een levenscyclus voor het ontwikkelen van software. Dit model werd bedacht in de jaren 1970 door Winston Royce en wordt tegenwoordig nog steeds gebruikt, al dan niet in verschillende versies. In figuur 1 wordt de originele versie van dit model afgebeeld. [3]

We zien dat binnen het Waterfall model het ontwikkelen van software gebeurt op een sequentiële manier doorheen afgebakende fases. Elke fase heeft een strikt begin- en eindpunt en moet voltooid zijn om met de volgende fase te kunnen starten. De vooruitgang van een Waterfall project wordt gemeten door middel van documenten. Tijdens elke fase wordt er documentatie, ook wel “deliverables” genoemd, gecreëerd die dient als invoer voor de volgende fase. Deze eigenschap maakt van het Waterfall model een document-gedreven model. [3]



Figuur 1: Het originele Waterfall model [3]

Het Waterfall model biedt enkele voordelen bij het produceren van software. Door de streng afgebakende fases is het steeds nodig dat fase x volledig voltooid wordt voordat er gestart kan worden met fase x+1. Dit zorgt voor discipline bij het uitvoeren van het project. Verder wordt er gewerkt met strikte en duidelijke deadlines, die ervoor zorgen dat de voortgang van het project nauwkeurig gemeten kan worden door de uitvoerder enerzijds en de klant anderzijds. Documentatie speelt hierbij een belangrijke rol. Doordat de analyse en het design volledig op voorhand gebeuren, verspillen de ontwikkelaars weinig tijd bij de implementatie.

Ondanks deze voordelen, heeft het Waterfall model echter ook nadelen. Deze nadelen stammen onder meer uit het verkeerd gebruik van het model door ontwikkelaars [4]. In [3], legt Royce reeds de nadruk op enkele belangrijke aspecten die in rekening moeten gebracht worden bij het gebruik van het Waterfall model. In het standaard model zoals afgebeeld in figuur 1, zien we dat er een iteratieve relatie is tussen opeenvolgende fases (pijlen naar

links). Deze relatie zorgt ervoor dat, wanneer er problemen zijn in fase x, er steeds teruggekeerd kan worden naar fase x-1 voor het verbeteren van deze problemen. Doordat er enkel een relatie is met de aangrenzende fases, is er steeds een progressieve basis waarop men kan terugvallen in geval van problemen. Hierdoor is het mogelijk om het reeds voltooide werk duidelijk af te lijnen. In de praktijk is deze werkwijze echter niet van toepassing. Tijdens de testfase wordt er namelijk voor het eerst een grondige validatie uitgevoerd op de software. Wanneer er problemen of fouten gevonden worden, is het dikwijls niet mogelijk deze op te lossen door eenvoudigweg enkele aanpassingen te maken in de code. Een terugkeer naar de designfase is dan de enige mogelijkheid. Bij het aanpassen van het design zal men in vele gevallen stuiten op problemen die te wijten zijn aan de requirements die vooropgesteld werden. Kortom, wanneer er tijdens het testen problemen gevonden worden, is de kans groot dat er een aanpassing moet gebeuren van de requirements. Dit betekent dat het nodig is het volledige proces opnieuw te doorlopen, wat enorme kosten en een verhoogd risico met zich mee brengt. [3]

Om dit risico te vermijden, stelt Royce vijf toevoegingen aan het standaard Waterfall model voor [3]:

1. Toevoeging van een inleidende designfase
2. Doorgedreven documentatie van het design
3. Creëren van een “pilot-model”
4. Zorgvuldig testen
5. Betrekken van de klant

Door het uitvoeren van een initieel design alvorens de analyse plaatsvindt, krijgt men de mogelijkheid de behoefte aan resources te onderzoeken voordat definitieve analyse en design plaatsvinden. Dit zorgt voor een daling van het risico op falen. Het tweede punt benadrukt het belang van documentatie als communicatiemiddel tijdens het proces. In punt drie wordt er voorgesteld het volledige Waterfall proces in een beperkte tijd en diepgang uit te voeren na de inleidende designfase. Dit is eigenlijk een simulatie van het proces waarin er precies gecontroleerd kan worden hoeveel tijd de verschillende stappen in beslag zullen nemen en welke resources er allemaal nodig zullen zijn. Nadat de implementatie voltooid is, volgt de testfase. Deze fase moet op een grondige manier plaatsvinden. Documentatie speelt hierbij een belangrijke rol en laat het toe externe testpersonen in te zetten. Ten slotte is het van belang de klant continue te betrekken bij het ontwikkelproces. Tijdens de inleidende designfase, het definitieve design en de testfase kan de klant door zijn inbreng zorgen voor de feedback. [3]

Ondanks de nadruk die Royce legde op deze factoren, wordt het Waterfall model in de praktijk meestal verkeerd toegepast [4]. Doordat het model gezien wordt als een “single-pass” sequentieel proces, gaat de iteratieve relatie tussen de fases verloren wat zorgt voor een groot risico door het gebrek aan feedback. Niet enkel het verkeerde gebruik, maar ook de structuur van de cyclus zorgt ervoor dat projecten falen. Het Waterfall model stelt het ideale ontwikkelproces voor, waarbij de requirements vooraf eenduidig vastgesteld kunnen worden en dat deze, samen met de wereld, niet veranderen. Echter dit is in de praktijk zelden zo. [4] Bij de aanvang van een project is het namelijk moeilijk voor de klant om zich een beeld te vormen van al zijn requirements, zonder dat hij enige notie heeft over hoe de software eruit zal zien. In de loop van het project krijgt hij een beter inzicht op eventueel ontbrekende functionaliteit en geeft feedback door communicatie met de ontwikkelaars. Dit

kan gebeuren tijdens de designfase of tijdens de implementatie (bij het in werking zien van bijvoorbeeld de ontworpen gebruikersinterface). Wanneer het ontwikkelteam dan feedback ontvangt, is het niet triviaal voor het om hiermee rekening te houden. Men moet dan terugkeren naar de eerste fase van de ontwikkelcyclus om requirements aan te passen of toe te voegen. In het geval dat de requirements toch volledig kunnen vastgesteld worden, is het meestal zo dat tijdens de implementatie extra details naar voren komen. [5]

In de moderne Waterfall cyclus wordt het volledige softwareproduct in zijn geheel ontworpen, geïmplementeerd, getest en uitgerold. Er kan dus weinig tot geen beroep gedaan worden op de feedback van de klant, wat resulteert in een hoog risico en hoge kosten bij het opnieuw doorlopen van het proces. Reeds in 1976 maakte Harlan Mills hier volgend statement over:

> *"Software development should be done incrementally, in stages with continuous user participation and replanning and with design-to-cost programming within each stage." [6]*

Men zou zich kunnen afvragen waarom het Waterfall model zelfs tegenwoordig nog steeds aanbevolen wordt. Volgens Larman et al. [4] zijn er voor dit fenomeen volgende verklaringen:

- Het model is eenvoudig om uit te leggen. Door de jaren heen is het originele Waterfall model [3] steeds vereenvoudigd naar voren gebracht, met een verkeerde adoptie tot gevolg.
- Men presenteert het Waterfall model als zijnde een makkelijk te managen en te meten proces met duidelijke milestones.
- Het werd gepromoot als het aangewezen model voor software ontwikkeling in vele artikels en teksten en door consultanten.

Software ontwikkelen is een complex proces dat veel creativiteit, communicatie en collaboratie vereist en dat kan het Waterfall model niet bieden. Verder is het noodzakelijk dat deze software gebruikersvriendelijk en bruikbaar is voor de eindgebruikers. In deze thesis zullen we dan ook een proces proberen te definiëren dat de kans op het creëren een succesvol softwareproduct verhoogt.

1.3 Overzicht van deze thesis

In Hoofdstuk 2 wordt Agile Software Ontwikkeling besproken, als voorgestelde oplossing voor de problemen aangehaald in bovenstaande inleiding. Hoofdstuk 3 behandelt het User-Centered Design proces, een methode die streeft naar het creëren van gebruiksvriendelijke producten door het continue betrekken van gebruikers doorheen het ontwikkelproces. Deze methode zal vervolgens aangewend worden in hoofdstuk 4 ter beschrijving van een ontwikkelproces dat de methoden en technieken van Agile Software Development en User-Centered Design integreert, om zo te komen tot een flexibel proces voor het bouwen van bruikbare en gebruiksvriendelijke software. Vervolgens gaan we verder in hoofdstuk 5 met het bespreken van de StoryDraw, de softwaretool ontwikkeld bij deze thesis ter

ondersteuning van het proces uit hoofdstuk 4. In hoofdstuk 6 worden de resultaten van de uitgevoerde gebruikerstest besproken, om te eindigen met een conclusie in hoofdstuk 7.

Deel II:

Literatuurstudie

Hoofdstuk 2

Agile Software Ontwikkeling

2.1 Inleiding

Het ontwikkelen van computerapplicaties houdt een zeker risico in. De klant moet namelijk tevreden zijn met het afgeleverde product en de deadlines van het project moeten gerespecteerd worden. Meer dan de helft van alle IT projecten falen [1], onder andere omdat de ontwikkelaars de gevraagde software niet binnen de afgesproken tijdspanne kunnen voltooien, of omdat er niet aan de functionele vereisten van de klant voldaan wordt [2]. Dit hoofdstuk bespreekt een methode van softwareontwikkeling die poogt het risico van een softwareproject te verkleinen om zo de kans tot slagen te verhogen, namelijk Agile Software Ontwikkeling.

Agile Software Ontwikkeling staat voor een reeks waarden en principes die vorm geven aan het software ontwikkelproces door de wendbaarheid van het proces te verhogen en door frequent afgewerkte stukken functionaliteit te tonen aan de klant zodat deze snel feedback kan geven [7]. De term “Agile” werd voor het eerst gebruikt in 2001, toen de “Agile Alliance”[1] gevormd werd. Dit is een groepering die sterk gelooft in de principes achter Agile Software Ontwikkeling. Zij stelden het “Agile Manifesto” [8] op, waarin deze principes beschreven worden.

In sectie 2.2 volgt een bespreking van Iteratieve en Incrementele Ontwikkeling, het basisprincipe achter Agile methodes. De principes van Agile Software Ontwikkeling worden besproken in sectie 2.3. Sectie 2.4 legt de werkwijze achter Agile Software Ontwikkeling uit en bespreekt de Agile levenscyclus. In sectie 2.5 worden enkele voorbeelden aangehaald van een aantal methoden die steunen op de Agile principes. De voor- en nadelen van Agile Software ontwikkeling worden besproken in secties 2.6 en 2.7, om in sectie 2.8 te besluiten met een conclusie bij dit hoofdstuk.

2.2 Iteratieve en incrementele ontwikkeling

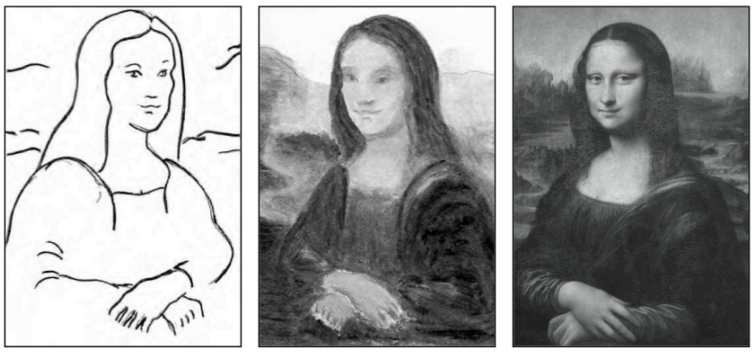
> *“A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system.”*
> *(Gall’s Law [9])*

[1] Agile Alliance, http://www.agilealliance.org/

Agile Software Ontwikkeling is gebaseerd op de principes van Iteratieve en Incrementele Ontwikkeling, ofwel Iterative and Incremental Development (IID) [4]. In deze sectie bespreken we kort enkele belangrijke eigenschappen van IID waarop Agile Software Ontwikkeling steunt. We merken op dat de termen “iteratief” en “incrementeel” vaak verward worden [10]. Daarom worden deze methoden ter verduidelijking besproken in sectie 2.2.1 en 2.2.2.

2.2.1 Iteratieve ontwikkeling

Door het gebruik van iteratieve ontwikkeling kan men (een gedeelte van) een bestaand product of systeem verfijnen, reviseren of verbeteren [10]. Deze werkwijze kan best geïllustreerd worden door een praktisch voorbeeld. In figuur 2 worden drie verschillende fases in het ontwerp van de Mona Lisa[2] getoond. We zien reeds in de linkse afbeelding een duidelijke vorm van een vrouw. Na feedback van de opdrachtgever, werd de bestaande afbeelding gereviseerd in de tweede stap om zo tot een volledig afgewerkt schilderij te komen (afbeelding rechts). Het schilderij werd dus op een iteratieve manier ontwikkeld. Dit proces kan ook toegepast worden bij het ontwikkelen van softwareproducten. Door inbreng vanwege de klant of vanwege gebruikers, kan reeds geïmplementeerde software verfijnd of verbeterd worden.



Figuur 2: Iteratieve ontwikkeling van de Mona Lisa [10]

Volgens Cockburn [10], zijn er twee tegenovergestelde strategieën voor het itereren van een systeem:

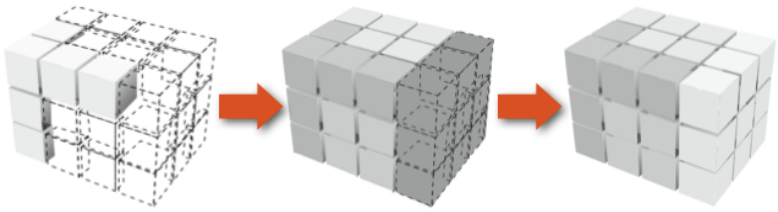
- Starten met het bouwen van een vrijwel compleet systeem, in de hoop dat bij eventuele feedback de nodige aanpassingen miniem zullen zijn en dat deze snel en eenvoudig geïncorporeerd kunnen worden.
- Vertrekken van een absoluut minimum, met het oog op het feit dat wanneer er veranderingen moeten gebeuren, er weinig tijd en kosten verloren gegaan zijn omdat er veel van het werk niet meer kan gebruikt worden.

[2] Wikipedia - Mona Lisa, http://nl.wikipedia.org/wiki/Mona_Lisa

Bij het uitvoeren van een project, is het niet noodzakelijk slechts één benadering te kiezen. Het afwisselen of combineren ervan kan gebeuren, afhankelijk van de omstandigheden en de aard van het project (bijvoorbeeld eventuele deadlines of tijd- en kostenbesparingen). Het voordeel van het gebruik van een iteratieve werkwijze, is dat het vooropgestelde probleem (doel van en vereisten voor de software) geleidelijk aan beter begrepen worden, en dus ook de oplossing voor dit probleem verfijnd kan worden. [11]

2.2.2 Incrementele ontwikkeling

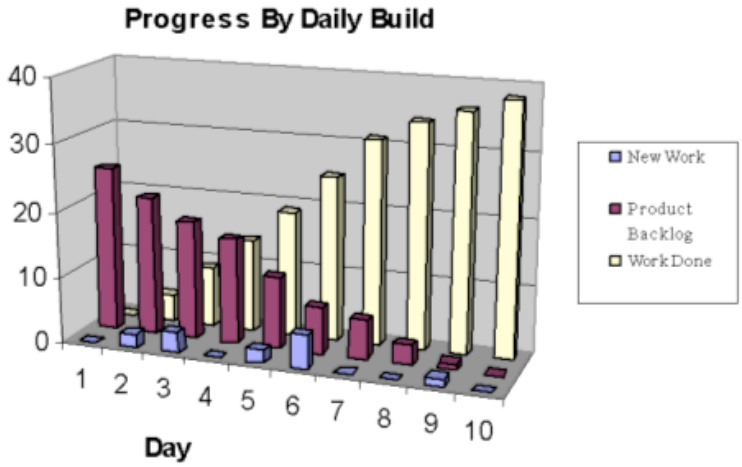
Wanneer men een systeem incrementeel opbouwt, wordt er bij elk increment een blok functionaliteit toegevoegd aan en geïntegreerd met een reeds gecreëerd product. Het totale werk dat moet gebeuren, wordt met andere woorden opgedeeld [10]. Er hoeft echter niet gewacht te worden totdat het volledige product klaar is voordat het kan afgeleverd worden aan de klant. Men kan beslissen in een bepaalde fase dat de aanwezige functionaliteit en mogelijkheden voldoende zijn om gebruik te maken van het product. Dit wordt ook wel Incrementeel en Evolutionair afleveren genoemd (zie sectie 2.2.6). Het proces van incrementele ontwikkeling wordt geïllustreerd in figuur 3.



Figuur 3: Incrementele ontwikkeling [10]

Nadat een increment voltooid is, wordt er een demonstratie van de toegevoegde functionaliteit gegeven aan de klant, wat het mogelijk maakt deze functionaliteit te inspecteren in de context van de totale applicatie. De feedback die de klant geeft tijdens deze demo, wordt vervolgens gebruikt als input voor het volgende increment.

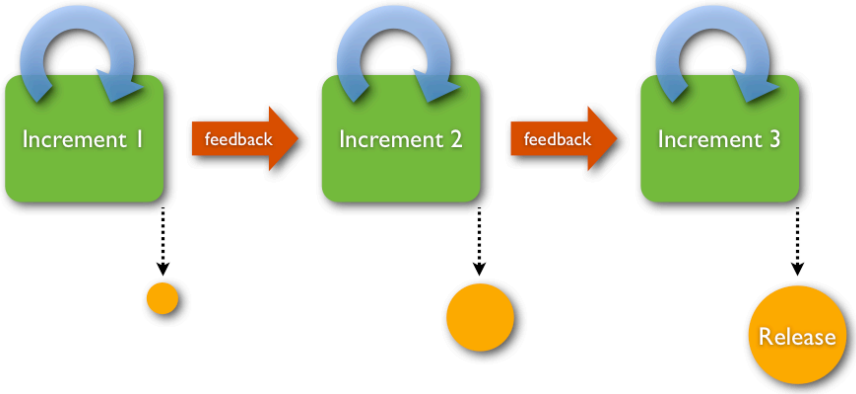
De verzameling van vooropgestelde requirements voor een project, worden bij incrementele ontwikkeling opgeslagen in een zogenaamde “product backlog” (zie ook sectie 2.5.2 over Scrum). De items aanwezig in deze backlog zijn geprioriteerd volgens bepaalde criteria (bijvoorbeeld risico, kritisch, e.d.) [12]. Het principe van een product backlog wordt getoond in figuur 4, waar de omvang ervan vergeleken wordt met het reeds voltooide werk.



Figuur 4: Werkwijze van een product backlog [11]

2.2.3 De twee methodes combineren

Zoals we verder zullen zien, steunen Agile methodes op een combinatie van iteratieve en incrementele ontwikkeling, namelijk op “Iterative and Incremental Development” (IID) [13] [14]. Deze twee methodes zijn op een voor de hand liggende manier samen te gebruiken. De richtlijnen hieromtrent zijn niet strikt gedefinieerd en er kan dus een combinatie gekozen worden die past in de context van het project. De basis hiervoor is dat, voordat een increment geïntegreerd wordt, er een reeks iteraties erover uitgevoerd worden om de kwaliteit van dat stuk functionaliteit te maximaliseren. Deze werkwijze wordt afgebeeld in figuur 5.



Figuur 5: Iteratieve en Incrementele Ontwikkeling [13]

Zo een iteratie duurt binnen IID typisch één tot zes weken. Deze korte tijdspanne zorgt ervoor dat het risico op een falend project aanzienlijk verkleint, omdat het project regelmatig

opgevolgd kan worden. Merk op dat een increment bij IID in zijn geheel, inclusief de interne iteraties, ook wel een “iteratie” wordt genoemd. [13]

Door de frequente aflevering van werkende software heeft de klant een goed zicht op de vooruitgang van het project en kan hij bijsturen door het geven van feedback. Het softwaresysteem groeit op deze manier geleidelijk aan uit tot een afgewerkt product dat kan vrijgegeven worden aan de klant.

Een increment heeft als doel een deel van de requirements van het project te behandelen en heeft dus met andere woorden zijn eigen set van requirements. Hoe deze bepaald worden, staat beschreven in sectie 2.2.4.

2.2.4 Opstellen van de requirements voor een iteratie

Bij het Waterfall model, wordt de volledige lijst van requirements vast bepaald alvorens het ontwerpen en ontwikkelen van start gaat. Dit kan ervoor zorgen dat het ontwikkelteam te maken krijgt met onrealistische verwachtingen vanwege de klant, wat kan zorgen voor frustraties bij de ondervinding dat de opgelegde requirements niet gehaald kunnen worden [11]. Een mogelijke oplossing voor dit probleem kan gevonden worden in een proces genaamd “requirements pipeline”. Bij deze werkwijze is het nodig dat er een apart requirements-team aangesteld wordt dat, naarmate het project vordert, de requirements opstelt voor de volgende iteratie. Deze methode kan echter slechts in beperkte gevallen gebruikt worden, vanwege bijvoorbeeld geografische redenen. [11]

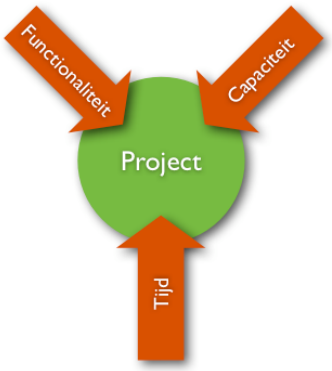
Het opstellen van de requirements kan beter ingebakken worden in het proces van iteraties. Aan het begin van elke iteratie worden dan telkens een set van requirements opgesteld, specifiek voor die iteratie. Tijdens deze werkwijze worden de requirements dus “just-in-time” bepaald. [11]

Het bepalen van de requirements voor een volgende iteratie kan op twee manieren gebeuren: door de klant zelf (client-driven), of op basis van risico (risk-driven). Een client-driven methode betekent dat het ontwikkelteam de requirements voor de volgende iteratie afleidt uit feedback van de klant tussen twee iteraties. De functionaliteit die de klant het belangrijkst vindt, wordt dus eerst geïmplementeerd. De planning van het project is dus adaptief per iteratie. Bij een risk-driven benadering daarentegen beslist het projectteam welke functies er eerst zullen worden ontwikkeld, gebaseerd op de moeilijkheidsgraad en het risico van deze componenten. Op die manier worden alle belangrijke onderdelen van het systeem vroeg in het project behandeld, wat zorgt voor een beter inzicht in de totale architectuur. [13]

Bij het opstellen van requirements is een combinatie van client-driven en risk-driven aangeraden. Zo worden componenten met een hoog risico reeds vroeg in het ontwikkelproces aangepakt, terwijl ook de noden van de klant niet uit het oog worden verloren. Om de stabiliteit van het proces te verhogen, wordt het wijzigen van de requirements door de stakeholders tijdens een iteratie niet toegelaten. Eventuele veranderingen moeten wachten tot een volgende iteratie. [13]

2.2.5 Time-management bij IID

Bij een project worden er deadlines vastgesteld die aangeven wanneer bepaalde belangrijke mijlpalen (ook wel milestones genoemd) bereikt moeten zijn. Deze deadlines moeten gerespecteerd worden om een vlotte voortgang van het project te kunnen garanderen. Het halen van een deadline hangt af van drie belangrijke factoren namelijk tijd, functionaliteit en capaciteit [15], zoals te zien is in figuur 6.



Figuur 6: Variabelen bij het halen van een deadline [15]

Wanneer een deadline nadert en de afgesproken functionaliteit niet gehaald zal worden, zijn er drie mogelijkheden [15]:

- **De deadline verschuiven naar een later tijdstip**
  Door het verschuiven van de deadline zal het verloop van het project verstoord worden en zal het project niet tijdig voltooid zijn.
- **De capaciteit van het projectteam verhogen**
  Door mensen toe te voegen aan het projectteam, zou men in principe meer vooruitgang moeten kunnen boeken, maar in de praktijk is dit zelden zo. Nieuwe mensen hebben namelijk tijd nodig om zich in te werken in het projectteam en om de context en situatie van het project te begrijpen. Dit zorgt voor een overhead die het project extra vertraagt.
- **De functionaliteit inkrimpen**
  Door functionaliteit te schrappen kan ervoor gezorgd worden dat de vooropgestelde deadline toch gehaald wordt. De tijd en de capaciteit blijven dan constant, maar het uiteindelijke product van de deadline mist dan wel een deel van zijn functionaliteit.

Bij IID is elke iteratie onderworpen aan een deadline. Bij de start van een iteratie wordt een duidelijke datum vastgelegd waarvan niet mag worden afgeweken. Wanneer de deadline dreigt niet gehaald te worden, moet het doel van de iteratie ingekrompen worden, met andere woorden bepaalde requirements zullen niet voldaan zijn bij het beëindigen van de iteratie. Elke iteratie bevindt zich in een zogenaamde “Time Box”, deze werkwijze wordt daarom ook “Timeboxing” genoemd [16].

Het afslanken van de requirements bij Timeboxing moet gedisciplineerd gebeuren. Daarvoor kan gebruikt gemaakt worden van de “MoSCoW” methode [17]. Deze werkwijze maakt gebruik van vier prioriteitsniveaus voor de requirements. De klant beslist bij de start van een iteratie de prioriteiten van de requirements voor die iteratie op basis van volgende schaal:

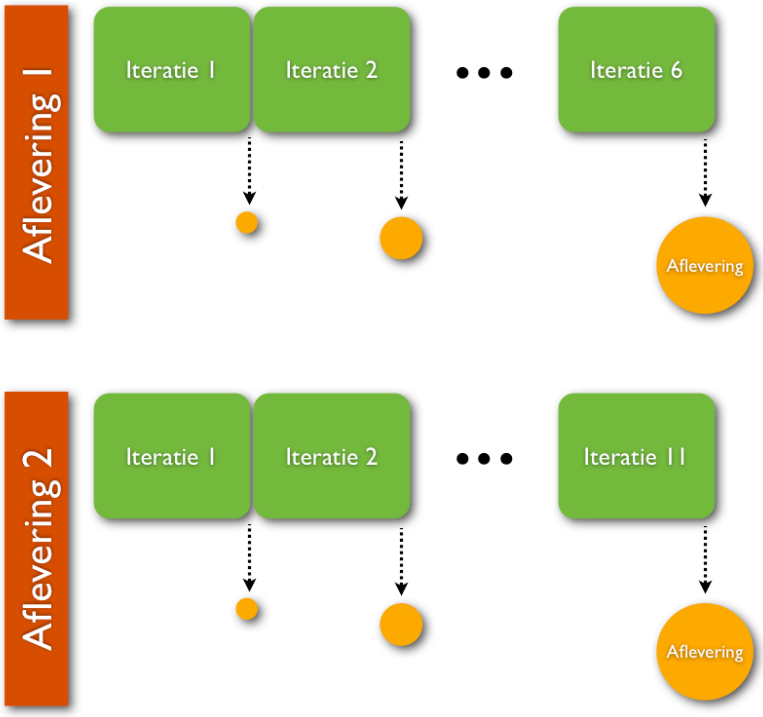
- **Must Have**
  Requirements met het label “Must Have” hebben de hoogste prioriteit. Deze requirements zijn essentieel voor de iteratie waartoe ze behoren en moeten volledig behandeld worden.
- **Should Have**
  Het prioriteitslabel “Should Have” duidt op een belangrijk requirement waaraan voldaan moet zijn aan het einde van de iteratie, maar alternatieve oplossingen of “work-arounds” zijn mogelijk.
- **Could Have**
  Wanneer een requirement niet noodzakelijk is om het slagen van de iteratie te garanderen, maar toch een nuttige feature binnen het systeem betekent, kan deze als “Could have” aangeduid worden.
- **Want to Have But Won't Have This Time Around**
  Requirements die nuttig zijn maar uitgesteld worden naar een volgende iteratie, worden gelabeld met “Want to Have But Won't Have This Time Around”.

Deze prioriteitsregels geven de ontwikkelaars de mogelijkheid om het einddoel van een iteratie in te krimpen, zonder dat er belangrijke functies ontbreken. Het stellen van prioriteiten gebeurt in samenwerken met de klant zodat de functies die de klant belangrijk vindt, reeds vroeg geïmplementeerd zullen worden.

Het is nuttig op te merken dat het niet nodig is dat alle iteraties binnen een project dezelfde lengte aannemen, sommige iteraties vergen immers meer tijd dan andere.

2.2.6 Incrementeel en evolutionair afleveren

In een standaard IID project, ontwikkelt men een softwaresysteem door te werken met iteraties die telkens functionaliteit toevoegen aan het systeem. Uiteindelijk volgt dan de aflevering van het eindproduct aan de klant die de software kan gaan gebruiken. Deze werkwijze kan uitgebreid worden met “Incremental Delivery”, of incrementeel afleveren” [13]. Bij Incremental Delivery wordt de aflevering van het softwareproduct verspreid over meerdere incrementele afleveringen (zie figuur 7). De datum van deze afleveringen zijn vooraf vastgelegd.



Figuur 7: Incrementele aflevering [13]

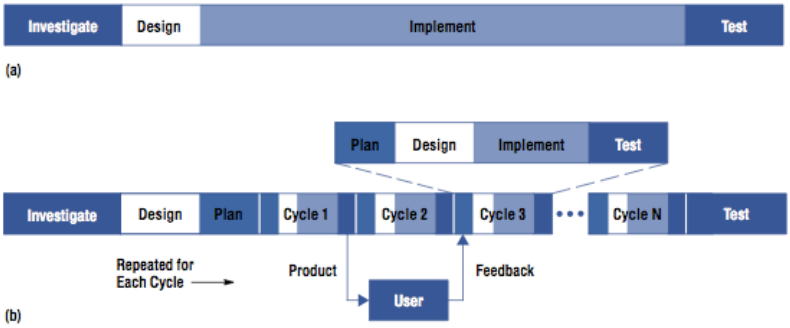
Zoals te zien is in figuur 7, is elke aflevering samengesteld uit een reeks iteraties. Het aantal iteraties per aflevering is variabel. Doordat de software reeds vroeg kan geleverd worden aan de klant, moet deze niet wachten totdat de software volledig klaar is vooraleer hij aan de slag kan. Omdat de requirements opgesteld zijn op basis van risico en in samenwerking met de klant, bevatten de eerste incrementele afleveringen reeds de kernfunctionaliteit en de functies die de klant belangrijk vindt.

De methode van Incremental Delivery kan verfijnt worden door het toepassen van “Evolutionary Delivery” oftewel evolutionaire aflevering. Evolutionary Delivery voegt het verzamelen van feedback toe aan het proces van Incremental Delivery. Deze feedback komt van de klant en eindgebruikers en bevat hun bevindingen in verband met het product. Het feit dat de software gebruikt wordt in levensechte omstandigheden en door echte eindgebruikers, maakt het mogelijk dat er meer fouten ontdekt kunnen worden dan mogelijk is bij het testen van het product door de ontwikkelaars of de klant alleen. De verzamelde feedback kan vervolgens gebruikt worden om volgende afleveringen te plannen. [13]

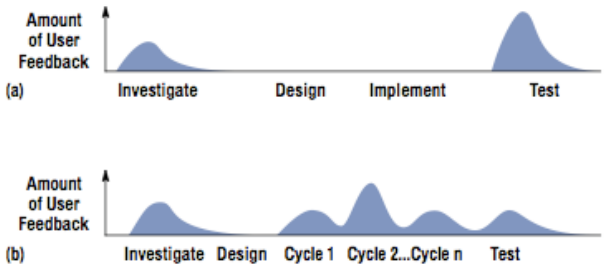
2.2.7 Voorbeelden van IID

Een eerste voorbeeld van een IID methode is “EVO” [13]. EVO maakt gebruik van iteraties met een lengte van één tot vier weken en gebruikt Evolutionary Delivery met als doel reeds vroeg in proces en op regelmatige basis feedback te ontvangen van de klant [18]. Deze methode is ontstaan in de jaren 1960. In figuur 8 wordt het EVO model vergeleken met het Waterfall model. Door de feedback van de klant op regelmatige tijdstippen, daalt het risico van het project en dalen de kosten omdat fouten reeds vroeg opgespoord - en dus ook

verbeterd - kunnen worden. Het verschil van de timing en de hoeveelheid van feedback tussen het EVO en het Waterfall model, wordt geïllustreerd in figuur 9.

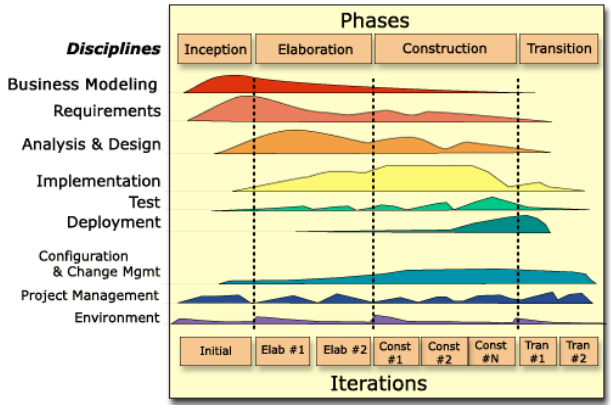


Figuur 8: Het EVO ontwikkelmodel (b) vergeleken met het Waterfall model (a) [18]



Figuur 9: Het proces van feedback bij EVO (b), vergeleken met het Waterfall model (a)

Unified Proces [19] (UP) is een tweede voorbeeld van een methode die steunt op IID. Een project gebruik makend van UP bestaat uit een cyclus van vier fases die zich op lange termijn kan herhalen. Deze vier fases dragen de namen inceptie, elaboratie, constructie en transitie en zijn elk onderverdeeld in iteraties. Een iteratie bestaat hoofdzakelijk uit analyse, design, implementatie en testen en deze disciplines kunnen variabel verdeeld worden per iteratie (zie figuur 10).



Figuur 10: Projectfases bij Unified Process [19]

Als belangrijkste artefact worden Use Cases gebruikt bij UP. Deze vatten de requirements en dus het doel per iteratie samen. Verder gebruikt UP een risk-driven methode. Hierdoor worden eerst de cruciale elementen van het project aangepakt. [19] UP ontstond midden jaren negentig.

2.3 Principes van Agile Software Ontwikkeling

Agile Software Ontwikkeling steunt sterk op de pijlers van Iterative and Incremental Development. Het maakt gebruik van iteraties en de software krijgt meer en meer vorm bij elk increment. De iteraties zijn kort en onderhevig aan Timeboxing. Agile Software Ontwikkeling valt moeilijk strikt te definiëren. Het is meer een raamwerk dat enkele principes en waarden voor flexibele softwareontwikkeling benadrukt. Verschillende methodes, zoals Scrum en Extreme Programming (zie sectie 2.5), steunen op de Agile principes [7]. Deze methodes definiëren zelf de parameters van het ontwikkelproces, zoals de lengte van de gebruikte iteraties en de gebruikte artefacten, rekening houdend met principes van Agile Software Ontwikkeling. Wanneer we dit veralgemenen, kunnen we Agile methodes opdelen volgens de hoeveelheid documentatie die gebruikt wordt en de duur van de iteraties. In sectie 2.5 zullen we een aantal voorbeelden van Agile methodes classificeren op basis van deze indeling.

Om een beter inzicht te krijgen in Agile Software Ontwikkeling, is het nuttig eerst de betekenis van het woord “Agile” te bekijken. “Agile” heeft volgens Van Dale volgende betekenis:

“vlug en beweeglijk”

In de context van softwareontwikkeling wil dit zeggen dat er gestreefd wordt naar een flexibel en wendbaar ontwikkelingsproces met als doel snel te reageren op veranderingen van bijvoorbeeld requirements, budget of beschikbare technologie.

Zoals eerder vermeld bestaat Agile Software Ontwikkeling uit een reeks principes en aanbevelingen. De vier basisprincipes staan beschreven in het zogenaamde “Agile Manifesto” [8]:

- **Individuals and interactions** over processes and tools
- **Customer collaboration** over contract negotiation
- **Working software** over comprehensive documentation
- **Responding to change** over following a plan

In volgende paragrafen worden deze principes kort uitgelegd.

Het succes van een (software)project hangt voor een groot deel af van de productiviteit van de teamleden. Het is daarom belangrijk dat deze mensen gemotiveerd zijn, vlot kunnen werken en dat voldoende opleiding voorzien is. Binnen het team is een goede samenwerking tussen de teamleden essentieel. Dit bevordert de creativiteit, wat zorgt voor oplossingen van hogere kwaliteit. In plaats van gebruikt te maken van uitgebreide documenten en complexe tools voor communicatie binnen het team, is mondelinge conversatie aangeraden. Tijdens een groepsgesprek worden er ideeën uitgewisseld en worden problemen besproken. Dit stimuleert de teamgeest. Niet enkel de ontwikkelaars nemen deel aan deze gesprekken, ook het management en de klant zijn aanwezig. Het maken van aantekeningen kan gebeuren op plaknotities of index cards met behulp van kernwoorden. Deze notities kunnen bovendien eenvoudig gesorteerd worden, wat nuttig kan zijn bijvoorbeeld bij het prioriteren van requirements. Omdat Agile Software Ontwikkeling gebruik maakt van Timeboxing, is de druk op de ontwikkelaars bij het naderen van een deadline kleiner. Overuren zijn niet toegelaten. [13][20]

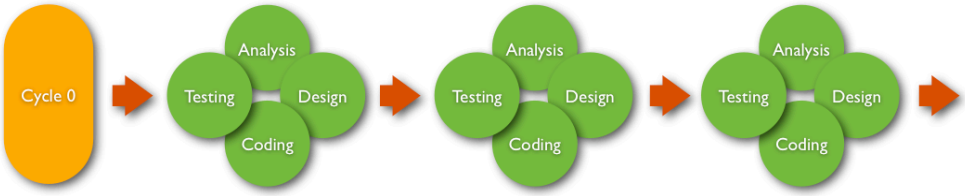
Het doel van software ontwikkelen is een oplossing bieden die voldoet aan de vereisten van de klant. Niemand beter dan de klant zelf kan zijn eisen beschrijven. Daarom is het nuttig nauw samen te werken met de klant en dit doorheen het volledige ontwikkelproces. De klant in deze context bestaat uit een vertegenwoordiger van de gebruiker van de software, een zogenaamde “user representative”. Het betrekken van de klant levert enkele significante voordelen op. Hij of zij kan ervoor zorgen dat de vereisten correct geïnterpreteerd worden en kan de prioriteit ervan aangeven. Doordat het hele team aanwezig is, kunnen zij bijkomende vragen stellen ter verduidelijking. Door de continue aanwezigheid van de klant, is de kans dat het juiste product op tijd afgeleverd wordt groter. [13][20]

De vooruitgang van een Agile project wordt gemeten door werkende software als primair artefact te stellen. Er wordt niet gewerkt met uitgebreide documentatie, aangezien deze dikwijls moeilijk begrepen wordt en een slecht beeld geeft van de werkelijkheid. Een werkend stuk software daarentegen, geeft wel een correct beeld van de werkelijkheid, omdat deze stukken, gecombineerd, het uiteindelijke product vormen. Doordat men regelmatig (op het einde van elke iteratie) een werkend product aflevert, krijgt zowel de klant als het management een beter inzicht in de status van het project. Dit wil niet zeggen dat er absoluut geen documentatie gebruikt mag worden, integendeel. Er moet echter wel voor gezorgd worden dat de documentatie leesbaar is voor verscheidene doelgroepen en dat deze “light-weight” en up-to-date is. Een combinatie met het vorige principe - customer collaboration - levert snelle feedback op van de klant, waardoor fouten of misverstanden kunnen verholpen worden wanneer ze plaatsvinden. [13][20]

Zoals hierboven besproken, zijn software projecten dikwijls onderworpen aan veranderingen. Daarom is het niet nuttig een uitgebreide en gedetailleerde planning op te stellen voor het project. Naarmate het project namelijk vordert, krijgen de ontwikkelaars een beter inzicht in het probleem, begrijpt de klant beter wat hij of zij wil en verandert de beschikbare technologie. Deze factoren zorgen ervoor dat de planning grondig kan wijzigen. Het is beter het detailniveau van de planning te koppelen aan de tijd. De eerstvolgende week kan bijvoorbeeld in detail gepland worden, terwijl de verder verwijderde weken slechts ruw gepland worden. Het wijzigen van de resolutie van de planning over de tijd noemt men ook wel “Rolling Wave Planning”. Door het integreren van adaptieve planning in dit proces, kan men het plan aanpassen op basis van kennis en feedback opgedaan in eerdere fases van het project. [21]

2.4 Levenscyclus van Agile Software Ontwikkeling

De vier basisdisciplines van softwareontwikkeling (analyse, design, implementatie en testen) komen bij een Agile levenscyclus steeds samen voor tijdens elke iteratie. Deze werkwijze wordt in figuur 11 afgebeeld.



Figuur 11: Agile levenscyclus [22]

De aanwezige disciplines hebben niet steeds hetzelfde aandeel in elke iteratie. In bepaalde iteraties kan er bijvoorbeeld relatief veel analyse gebeuren en in minder mate implementatie. Dit zal vooral zo zijn in de initiële fase van het project, aangezien de requirements zich geleidelijk aan stabiliseren naar mate het project vordert met als gevolg dat de tijd nodig voor analyse afneemt. [13]

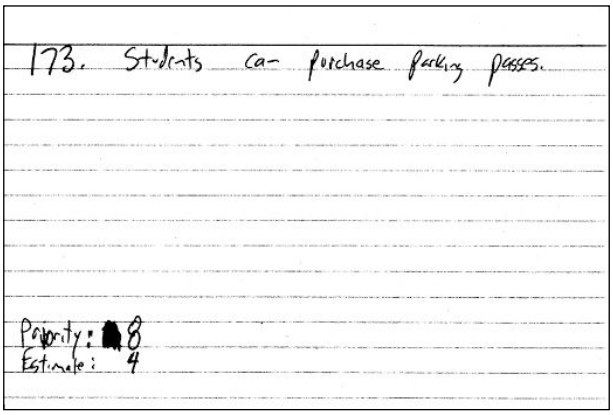
Ondanks het feit dat Agile Software Ontwikkeling afziet van het op voorhand grondig plannen, analyseren en ontwerpen van een systeem, is het aangeraden om, alvorens het proces van iteraties van start gaat, een korte initiële fase in te lassen waarin deze activiteiten minimaal behandeld worden. Tijdens deze fase, soms ook “cycle zero” genoemd, wordt de scope van het systeem gedefinieerd en een gedeelde visie gecreëerd. Verder worden de belangrijkste requirements opgesteld alsook het algemene doel van de applicatie. Zo ontstaat de mogelijkheid om onder andere een raming te maken van de kosten van het project, rekening te houden met de nodige performance, een basisarchitectuur te definiëren, een eerste hoog niveau user interface uit te werken en de stakeholders te identificeren. Om de efficiëntie van het proces te behouden, is de duur van

deze fase beperkt van een paar dagen tot hooguit enkele weken (afhankelijk van de projectgrootte, type applicatie, enzovoort). [23][24][25]

Bij de aanvang van een iteratie, worden telkens de requirements en het doel van die iteratie besproken tijdens een zogenaamde “requirements workshop”. Deze requirements zijn meestal een deelverzameling van de requirements opgesteld in cycle zero, aangepast volgens de huidige situatie van het project. Bij Agile Software Ontwikkeling worden User Stories gebruikt als methode voor het representeren van requirements [26][27]. Een User Story is een korte definitie van een hoogniveau requirement, opgesteld door de klant (of representatieve gebruiker) [27]. Enkele User Stories, voor bijvoorbeeld een applicatie die het toelaat trainingsuren te beheren van een voetbalclub, zouden kunnen zijn:

- Trainers kunnen hun trainingsuren opvragen
- Bestuursleden kunnen de contactgegevens van spelers en trainers aanpassen
- Spelers kunnen contactgegevens van hun groepsleden opvragen

Als artefact voor het noteren van User Stories kunnen index cards gebruikt worden (in deze context ook wel “Story Cards” genoemd). Een voorbeeld van een Story Card is te zien in figuur 12.



Figuur 12: Voorbeeld van een Story Card [27]

Het werken met Story Cards heeft als voordeel dat de requirements makkelijk gesorteerd kunnen worden, bijvoorbeeld volgens prioriteit. Verder kunnen ze op een muur of bord bevestigd worden zodat iedereen van het team ze kan zien, wat de communicatie en collaboratie verbetert. Uitgebreide requirements kunnen eventueel verder opgesplitst worden in kleinere onderdelen die passen op één kaartje.

Om een team in de goede richting te sturen, is er management nodig. Bij Agile Software Ontwikkeling wordt een zogenaamde “Process Facilitator” aangesteld die het team motiveert en conflicten oplost. Hij of zij is ook verantwoordelijk voor het aanleren van de Agile werkwijze.

Een Agile team bevat mensen met verschillende achtergrond en kennis, iedereen heeft zijn eigen expertise. Binnen een iteratie moeten de taken verdeeld worden onder deze teamleden. In een Agile team is iedereen gelijk en stelt iedereen zich multifunctioneel op. Iedere persoon neemt met andere woorden een deel van zowel het design, de implementatie als het testen op zich. Nadat de requirements uitgewerkt zijn in groep, worden deze omgezet in werkende software door elk van de groepsleden apart. Elk lid van het team neemt een bepaalde “feature” of reeks van features voor zich. De teamleden stellen zich zelf kandidaat bij het verdelen van de verantwoordelijkheden, ze kunnen dus zelf kiezen welke features ze zullen implementeren. Een feature wordt pas als voltooid gezien als deze volledig ontworpen, geïmplementeerd en getest is, met andere woorden als deze klaar is om afgeleverd te worden voor gebruik. Er wordt pas begonnen aan een volgende feature wanneer de huidige feature volledig klaar is. Op die manier bekomt men steeds een werkend geheel aan het einde van een iteratie. Onafgewerkte features worden overgedragen naar de volgende iteratie.

Bij Agile Software Ontwikkeling wordt er dikwijls per twee samengewerkt. Door deze samenwerking krijgen de teamleden de kans om bij te leren van elkaar en stijgt de kwaliteit van de geproduceerde code, omdat één van de twee teamleden continue de code nakijkt terwijl de andere deze schrijft. Deze techniek noemt men ook wel “Pair Programming”. [28]

Tijdens een iteratie is het nuttig dagelijks een korte vergadering te houden waar de teamleden de status van hun werk onderling bespreken. Enkel de ontwikkelaars zijn aanwezig, niet het management of de klant. Tijdens deze vergadering staan alle aanwezigen steeds recht, zo wordt de vergadering kort en bondig gehouden. De bedoeling is de situatie en de vooruitgang te schetsen, niet om problemen op te lossen. De communicatie wordt ondersteund door een centraal whiteboard, waarop de story cards aanwezig zijn. Dit helpt de teamleden een beeld te vormen van het werk dat nog moet gebeuren. Dit soort vergadering wordt ook wel een “stand-up meeting” of “Scrum” genoemd. Een typische stand-up meeting duurt 10 tot 15 minuten.

2.5 Voorbeelden van Agile methoden

In deze sectie worden enkele voorbeelden van Agile methodes besproken. Deze methodes gebruiken als basis de principes van Agile Software Ontwikkeling, maar voegen telkens hun eigen specifieke technieken toe.

2.5.1 Extreme Programming

Extreme Programming (XP) werd bedacht door Kent Beck in 1996. XP streeft naar een hoge klanttevredenheid en werkt rond vier waarden:

- Simpliciteit
- Communicatie
- Feedback
- Moed

Bij XP werkt een hecht team samen aan een project om op een zo kort mogelijke tijd werkende code af te leveren. Een XP team bestaat uit ontwikkelaars, het management en de klant. De klant speelt een uitgesproken rol tijdens het project en is dan ook betrokken gedurende het volledige project. [29]

Het plannen van een XP project gebeurt tijdens het zogenaamde “Planning Game”. Tijdens het Planning Game werken de developers samen met de klant de requirements uit voor het project. Elk requirement wordt opgesteld in de vorm van een User Story en genoteerd op een Story Card. Vervolgens schatten de ontwikkelaars de hoeveelheid tijd die nodig is om de verschillende user stories te implementeren. De klant beslist uiteindelijk welke user stories voorrang krijgen bij de implementatie. XP hanteert een iteratieve ontwikkelmethode met korte iteraties, die typisch één tot drie weken duren en maakt gebruik van iteratieve ontwikkeling met evolutionaire aflevering (zie sectie 2.2.6). De momenten waarop het ontwikkelteam een werkende versie aflevert, worden in overleg met de klant vastgelegd tijdens de “Release Planning”. Er wordt afgesproken welke User Stories moeten voltooid zijn bij elke release en ook de lengte van de iteraties wordt afgesproken. Alle iteraties bij XP hebben dezelfde lengte en zijn Timeboxed, hierdoor kan men makkelijk de vooruitgang meten. Het plannen van een iteratie gebeurt tijdens de “Iteration Planning”.

Een XP project vindt steeds plaats in een “project room”. Iedereen van het team is aanwezig in dezelfde kamer, ook de klant. Deze kamer ondersteunt de samenwerking binnen het team doordat er genoeg vrije ruimte is voor het maken van aantekeningen en het opstellen van story cards.

Bij het ontwerpen van een oplossing voor een user story, wordt er gestreefd naar de meest eenvoudige oplossing. Dit maakt de verschillende componenten van het systeem makkelijker om mee te werken. Hierdoor kan ieder teamlid makkelijker overweg met componenten die hij niet zelf ontwierp. Refactoring[3] wordt voortdurend toegepast tijdens het project, met als doel complexe componenten te vereenvoudigen.

Het eigenlijke implementeren van de user stories gebeurt met veel discipline. Testen is zéér belangrijk bij XP. Er zijn twee soorten van tests, namelijk de Unit Test en de Acceptance Test. Unit Tests worden opgesteld door de ontwikkelaars zelf voor er gestart wordt met implementeren. Vooraleer een bepaald component gecodeerd wordt, wordt er eerst een Unit test geschreven waaraan dat component moet voldoen. Een Acceptance Test wordt opgesteld door de klant, meestal per User Story. De implementatie van een user story is correct wanneer deze succesvol door de Acceptance Test komt. Het coderen gebeurt volgens strikte code-standaarden om de consistentie van de code te verhogen. Er wordt steeds Pair Programming toegepast. Hierdoor verbetert de kwaliteit van de code en worden er minder fouten gemaakt. De paren integreren om beurt regelmatig hun code met alle reeds geschreven code, zo wordt elk component tijdig getest op conflicten met de rest van het systeem en kan men tijdig code delen of hergebruiken. Het integreren gebeurt drie tot vier keer per dag op een gedeelde “code repository”. Regelmatig wordt er een “build” uitgevoerd op alle code. Wanneer deze build faalt, wordt het probleem opgelost voordat men verdergaat. Dit proces wordt ook wel “continues integration” genoemd. Tijdens het coderen is de klant steeds aanwezig om details te geven over de requirements en om Acceptance Tests op te stellen.

[3] Wikipedia - Code Refactoring, http://en.wikipedia.org/wiki/Refactoring

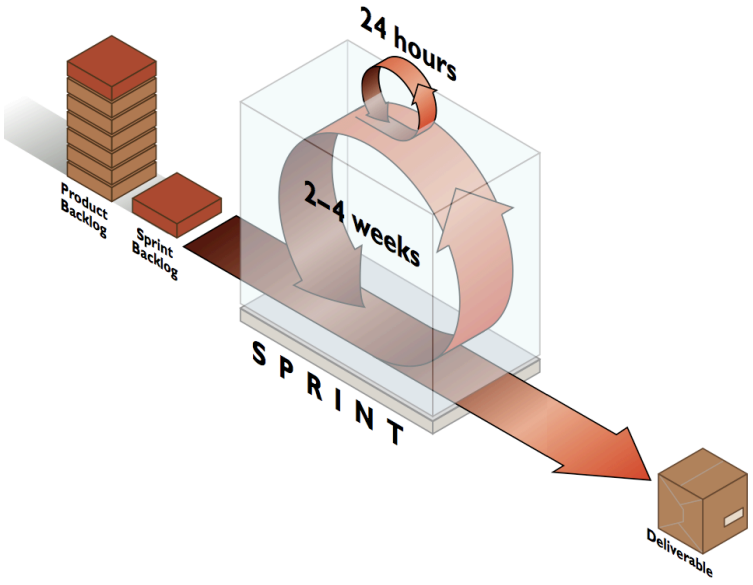
2.5.2 Scrum

Een tweede voorbeeld van een Agile ontwikkelmethode is Scrum [30]. In een Scrum team zijn hoofdzakelijk drie rollen gedefinieerd [30]:

- De Scrum Master
- De Product Owner
- Het Scrum Team

De Scrum Master is de coach van het project. Hij of zij zorgt ervoor dat conflicten opgelost worden en motiveert het team. De Product Owner stelt de requirements en bijbehorende prioriteiten op en representeert dus de klant. Het Scrum Team is een kleine groep van vijf tot negen mensen, is zelf-organiserend en kent geen leider. Alle leden dragen dezelfde verantwoordelijkheid voor het resultaat van de ontwikkelingen.

Een iteratie bij Scrum heeft de naam “Sprint”. Een Sprint duurt typisch twee tot vier weken. Tijdens een Sprint worden een deel van de requirements geïmplementeerd. De requirements, opgesteld door de Product Owner, worden opgeslagen in de zogenaamde “Product Backlog”. In de beginfase van een Sprint worden een aantal items met de hoogste prioriteit uit de Product Backlog gekozen ter implementatie. Het kiezen van de items gebeurt door de Product Owner in overleg met het Scrum Team. Deze items komen dan terecht in de “Sprint Backlog” en vormen het werkproduct voor de huidige Sprint. Deze werkwijze is te zien in Figuur 13.



Figuur 13: Het Scrum proces [86]

Een Sprint voegt telkens waarde toe aan het product door functies toe te voegen en verbeteringen aan te brengen. Het product van elke Sprint is volledig afgewerkt en de klant kan het in gebruik nemen. Op het einde van een Sprint vindt er telkens een demonstratie plaats voor onder andere de klant, het management en de gebruikers.

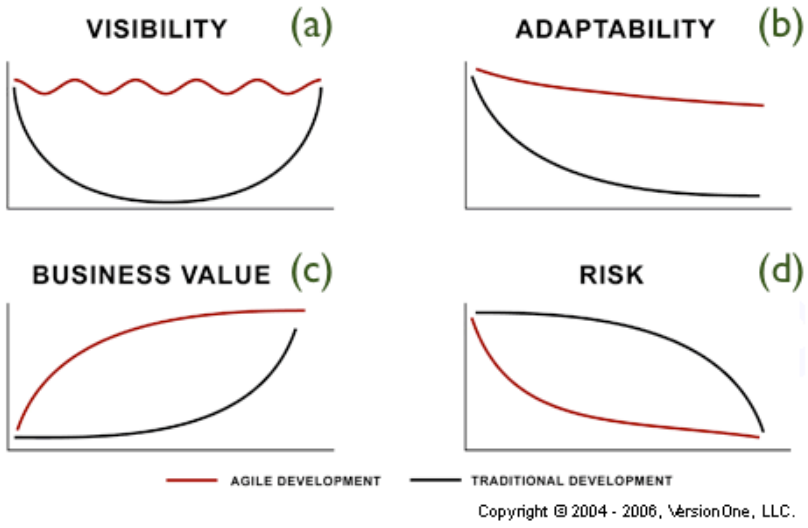
Bij de start van een nieuwe dag wordt er een korte stand-up meeting, ook wel “scrum” genoemd, gehouden met als aanwezigen de Scrum Master en het Scrum Team. Tijdens deze vergadering beantwoordt elk teamlid volgende vragen:

- Wat hebt u gedaan sinds de vorige meeting?
- Wat zal u vandaag gaan doen?
- Zijn er enige problemen die uw planning kunnen verstoren?

Doordat ieder teamlid deze vragen beantwoordt, krijgt men snel een beeld van de status van de Sprint en kunnen eventuele problemen aangepakt worden.

2.6 Voordelen van het gebruik van Agile ontwikkelmethodes

Zoals we zagen in de vorige secties elimineert Agile Software Ontwikkeling een grote reeks van de knelpunten van traditionele softwareontwikkeling. Figuur 14 geeft een vergelijking tussen traditionele methodes (zoals het Waterfall model) en Agile methodes op basis van vier factoren.



Figuur 14: Vergelijking tussen het Waterfall model en Agile Software Ontwikkeling [31]

Het gebruik van Agile methodes reduceert reeds vroeg in het proces het risico dat gepaard gaat bij het ontwikkelen van software vanwege van de manier waarop het proces opgebouwd is (figuur 14d). Een door feedback gedreven, iteratieve werkwijze zorgt ervoor dat het systeem beter voldoet aan de eisen van klant, dit mede omdat de klant zelf prioriteiten mag opleggen aan de verschillende requirements. Korte iteraties met een vaste lengte maken het mogelijk om efficiënt om te gaan met veranderingen van bijvoorbeeld technologie, requirements, beschikbare budget of de samenstelling van het team en verhogen de controle van het ontwikkelteam en de klant op het project (figuur 14b).

Het feit dat elke iteratie telkens een werkend subdeel van het gehele systeem aflevert, verhoogt de zogenaamde “Early Value” van het systeem. De klant kan met andere woorden sneller gebruik maken van het systeem, zonder dat dit volledig afgewerkt is (figuur 14c). Dit zorgt ook voor een verhoogde zichtbaarheid van het project (de mate waarin vooruitgang en dergelijke zichtbaar zijn voor de klant en de ontwikkelaars) (figuur 14a).

De fysische afstand tussen de ontwikkelaars is bij Agile Software Ontwikkeling klein: dagelijks zijn er meetings om de voortgang van het project en eventuele problemen te bespreken. Tijdens deze meetings wordt de voorkeur gegeven aan mondelinge conversatie in plaats van geschreven documenten. Hierdoor ontstaat er vlotte communicatie, wat zorgt voor snellere resultaten. [24]

Agile Software Ontwikkeling streeft naar maximale kwaliteit van het afgeleverde product. Dit is mogelijk door het gebruik van technieken zoals Pair Programming, refactoring en Unit Tests.

2.7 Nadelen van het gebruik van Agile ontwikkelmethodes

Wanneer er een Agile proces wordt gebruikt bij het ontwikkelen van een applicatie, focussen de ontwikkelaars zich vooral op de technische uitwerking van de onderliggende code van het systeem. Deze code wordt onderworpen aan strenge testen voordat deze incrementeel samengevoegd wordt. Terwijl de structuur van de code zeker niet onbelangrijk is (bijvoorbeeld omwille van performance), is het vitaal dat ontwikkelaars tevens rekening houden met het ontwerpen en ontwikkelen van de interface tussen het systeem en de gebruiker. Deze interface bepaalt namelijk voor het merendeel de gebruiksvriendelijkheid en bruikbaarheid van het systeem. [26] Verder wordt bij Agile Software Ontwikkeling de user interface van de te ontwikkelen applicatie, net zoals de achterliggende architectuur, iteratief en incrementeel ontwikkeld. Bij elke iteratie voegt men functionaliteit toe, met als mogelijk gevolg dat de user interface ingewikkeld en onoverzichtelijk wordt. Dit zorgt voor een daling van de productiviteit van de eindgebruiker en staat recht tegenover het feit dat software de gebruiker moet ondersteunen bij het uitvoeren van zijn of haar taken. Om dit probleem op te lossen, is het mogelijk om tijdens de initiële planningsfase (cycle zero, zie sectie 2.4) een eerste overzicht op te stellen van de algemene gebruikersinterface van de applicatie. Echter wordt deze mogelijkheid vaak niet benut omdat dit gezien wordt als “Big Up Front Analysis and Design”, een taboe bij Agile ontwikkelmethoden.

Voor het verkrijgen van informatie en feedback, wenden de ontwikkelaars zich tijdens een Agile proces tot de klant en tot representatieve gebruikers. Het is echter niet zeker dat deze

informatie ook de wensen en noden van de echte eindgebruikers illustreert. Ook het gebrek aan usability testing voor het evalueren van het systeem werkt deze stelling in de hand. [32]

Vanwege het feit dat Agile Software Ontwikkeling zwaar steunt op regelmatige (dagelijkse) communicatie binnen het projectteam, is het bepaalde gevallen moeilijk om gebruik te maken van deze methodologie. Dit kan bijvoorbeeld het geval zijn wanneer het team gedistribueerd werkt of als de teamleden een verschillende taal spreken of afkomstig zijn van een andere cultuur [24].

Bij Agile Software Ontwikkeling wordt er gestreefd naar een snel, efficiënt en flexibel ontwikkelproces, wat de mogelijkheid biedt om snel te reageren op veranderingen, echter de gebruiksvriendelijkheid van de software wordt hierdoor vaak over het hoofd gezien. [8][26] [32][23]

2.8 Conclusie

Traditionele methodes voor het ontwikkelen van software brengen een groot risico met zich mee omdat ze grotendeels steunen op een sequentieel procesmodel met uitgebreide documentatie. Vanwege het feit dat het systeem pas in een laat stadium gevalideerd wordt door de klant, kosten aanpassingen veel tijd en geld (zie sectie 1.2). Als oplossing hiervoor werd in dit hoofdstuk Agile Software Ontwikkeling besproken. De principes en structuur van dit procesmodel zorgen voor flexibiliteit en snellere resultaten en verminderen het risico op falen. Zoals in sectie 2.7 werd besproken, is één van de belangrijkste nadelen van Agile Software Ontwikkeling het gebrek aan aandacht voor de user interface van het systeem. Naarmate het project vordert is het mogelijk dat er inconsistentie optreedt omwille van de incrementele werkwijze en het feit dat regelmatig refactoring wordt toegepast. In hoofdstuk 3 zullen we daarom verdergaan met het bespreken van een methode voor het verbeteren van de interactie tussen het systeem en de gebruiker, namelijk User-Centered Design.

Hoofdstuk 3

User-Centered Design

3.1 Inleiding

Een belangrijk aspect bij het ontwerpen van software, is gebruikersvriendelijkheid. Het hoofddoel van een computerapplicatie, is de gebruiker de mogelijkheid geven zijn taken succesvol te voltooien, en dit op een intuïtieve en vlotte manier. Ontwerp of “design” van de gebruikerservaring is dus cruciaal voor een succesvolle applicatie. Een goed design zorgt onder andere voor goede productiviteit van de eindgebruikers, hogere klanttevredenheid, minder stress en lagere servicekosten [33].

User-Centered Design (UCD) is een proces dat het ontwerpen van producten ondersteunt door de gebruiker centraal te stellen doorheen het volledige ontwikkelproces. Een UCD benadering kan toegepast worden op het ontwerp van verscheidene objecten uit ons dagelijkse leven, zoals een stoel of een computerklavier. Echter in deze tekst wordt UCD, vanzelfsprekend, toegepast op het ontwerp van computerapplicaties. Tijdens het UCD proces spelen de (toekomstige) gebruikers van de software een belangrijke rol. Door hun input en feedback krijgen de ontwikkelaars namelijk een beter zicht op de eindgebruikers en hun taken. De term “User-Centered Design” werd voor het eerst gebruikt door Donald Norman in [34]. De voorbije jaren is er veel geschreven over UCD, maar in de literatuur is er geen unieke en eenduidige definitie te vinden voor een universeel UCD proces [35]. De reden hiervoor is dat elk softwareproject verschillend is, en dus een proces vergt dat aangepast is aan de specifieke eigenschappen ervan. Deze verschillen kunnen zich uiten in bijvoorbeeld de doelgroep van de software, de beschikbare resources voor het project (zoals tijd, budget of aantal teamleden), het type van software (bijvoorbeeld commercieel of op maat gemaakt), enzovoort. Wanneer een bedrijf beslist om UCD te gebruiken voor een project, stellen zij zelf een UCD proces op dat aangepast is aan hun noden. Zij kiezen dus zelf de technieken en methoden uit die het best passen bij de parameters van het project en leggen zelf de focus op bepaalde activiteiten. Ondanks het feit dat er geen universeel toepasbaar proces gedefinieerd is, bestaan er wel een reeks UCD raamwerken die de verschillende fases van een UCD proces aangeven op een hoger, meer abstract niveau. Het meest bekende voorbeeld van zo een raamwerk is ISO 13407 [33]. In deze thesis wordt er gebruik gemaakt van deze internationale standaard als referentie voor het bespreken van de verschillende fases van UCD, evenals de bijbehorende technieken en methoden. Andere voorbeelden van zulke raamwerken zijn de processen van Mayhew en Cooper & Reimann [36].

UCD is complementair aan bestaande design methoden en zorgt voor een gebruikersgericht perspectief [33], het beschrijft echter niet alle aspecten van een software ontwikkelingsproces. De focus van design bij UCD ligt namelijk op de gebruikersinterface van de software, een essentieel element van een computerapplicatie. De gebruikersinterface, of “user interface”, zorgt voor de communicatie en interactie tussen een computersysteem en zijn gebruiker, en bepaalt voor een groot deel de gebruiksvriendelijkheid van een applicatie. Het is dan ook van vitaal belang dat deze

gestructureerd, duidelijk en consistent opgebouwd is. [37] Jef Raskin zei hier in 2000 het volgende over:

> *"What users want is convenience and results. But all that they see is the interface. As far as the customer is concerned, the interface is the product." [38]*

Doordat de user interface een zodanige impact heeft op de uiteindelijke gebruikerservaring, spitst UCD zich enkel toe op het ontwikkelen van de user interface en niet op het ontwerpen van de achterliggende architectuur (bijvoorbeeld databases, algoritmen of objectstructuren). De ontwikkeling omvat naast het ontwerp van de user interface ook het ontwerp van de interactie tussen de gebruiker en de desbetreffende interface (ook wel "Interaction Design" genoemd). Verder merken we op dat UCD toegepast kan worden bij het ontwikkelen van een nieuwe applicatie, evenals bij het opwaarderen, verbeteren of reviseren van bestaande software.

In sectie 3.2 volgt een bespreking van de principes die UCD hanteert, gevolgd door een overzicht en bespreking van de cyclus gedefinieerd door de eerder aangehaalde ISO standaard in sectie 3.3. Sectie 3.4 bespreekt de technieken en methoden die voorhanden zijn tijdens het gebruik van UCD. In secties 3.5 en 3.6 wordt een kritische kijk gegeven op UCD, om te besluiten met enkele statistieken in sectie 3.7 over het gebruik van UCD in de praktijk.

3.2 Principes van User-Centered Design

Het doel van UCD is het produceren van producten die nuttig, bruikbaar en gebruiksvriendelijk zijn voor de eindgebruiker. De belangrijkste principes die vormgeven aan UCD, zijn volgens ISO 13407 [33]:

- Actieve betrokkenheid van de gebruiker tijdens het proces
- Correct alloceren van functionaliteit tussen het systeem enerzijds en de gebruiker anderzijds
- Continu reviseren van ontwerpen (ook wel itereren genoemd)
- Gebruik maken van technieken afkomstig uit verschillende disciplines

Deze vier basisprincipes zijn een belangrijk referentiepunt bij het ontwikkelen van een bruikbaar systeem. In onderstaande paragrafen worden deze principes besproken.

Tijdens het ontwerpen van een user interface, denken ontwikkelaars al te vaak dat de eindgebruikers dezelfde voorkennis, eigenschappen en werkwijze hebben als zijzelf. Dit is echter zelden het geval. Door een gebruikersgerichte aanpak krijgt het ontwikkelteam een beter inzicht in het profiel van de eindgebruikers, evenals in de taken die zij uitvoeren. Door het actief betrekken van gebruikers, wordt deze informatie op een effectieve manier vergaard. Ontwikkelaars kunnen namelijk de gebruikers zelf ondervragen (tijdens bijvoorbeeld een interview) of hun gedrag en werkomgeving observeren. Om dit mogelijk te maken, is het nodig dat ontwikkelaars en gebruikers nauwgezet samenwerken. Een

grondige analyse van de gebruikers en hun taken, zorgt ervoor dat problemen vroeger in ontwikkelcyclus opgelost kunnen worden, waardoor de kosten dalen [33]. Meer informatie over technieken die gebruikt kunnen worden tijdens een gebruikers- of taakonderzoek is te vinden in sectie 3.3.1. De samenwerking met de eindgebruikers is niet enkel beperkt tot de initiële fases van het project. Ook tijdens het ontwerp en evaluatie van de user interface blijft de gebruiker een belangrijke bron van feedback.

Het tweede principe van UCD geeft het belang aan van het verdelen van de verantwoordelijkheden tussen het systeem en zijn gebruiker. Tijdens het gebruik van een softwareapplicatie worden er verschillende acties en taken uitgevoerd. Deze acties en taken worden deels door het systeem uitgevoerd, en deels door de gebruiker zelf. Een optimale verdeling op basis van de karakteristieken van het systeem en de gebruiker, zorgt ervoor dat de gebruiker zijn taken sneller en makkelijker kan uitvoeren.

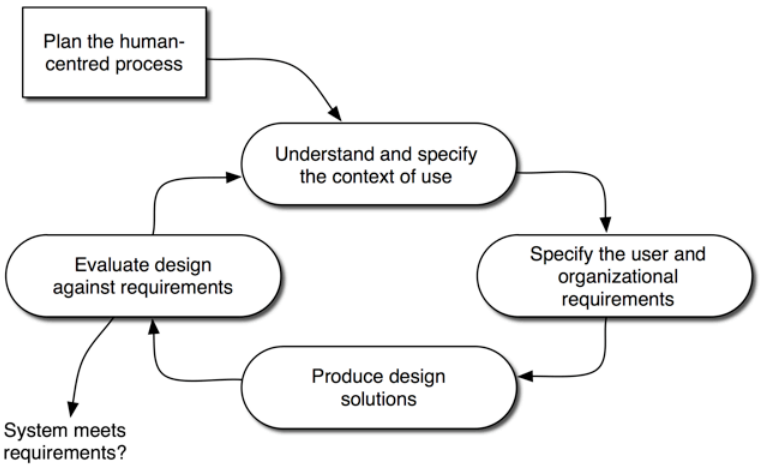
Eén van de redenen waarom vele softwaretoepassingen vaak niet voldoen aan de eisen van de klant, is het gebruik van een ontwikkelingsproces zonder iteraties (zie ook sectie 1.2) [2]. De iteraties, gebruikt binnen het UCD proces, zorgen ervoor dat men de gebruikers- en taakanalyse, het design en de evaluatie van de software geleidelijk aan kan uitvoeren, steeds verder werkend op een bestaand geheel en rekening houdend met feedback uit vorige iteraties. Door het betrekken van gebruikers tijdens elke iteratie, krijgen de ontwikkelaars steeds de mogelijkheid het design te evalueren aan de hand van bijvoorbeeld een gebruikerstest in een levensechte context. Wanneer men ondervindt dat de gebruiker minder productief is met een bepaald ontwerp, kan men verfijningen of verbeteringen aanbrengen in een volgende iteratie. Op die manier kunnen er tevens verschillende oplossingen getest en vergeleken worden.

Voor het effectief toepassen van de verschillende technieken van UCD (zie sectie 3.4), is er kennis en expertise nodig uit verschillende domeinen. Daarom is het noodzakelijk voor een UCD project dat er gebruikt gemaakt wordt van een multidisciplinair team. Zo een team kan onder andere bestaan uit de eindgebruiker, een domein-specialist, een programmeur, een interface designer en een HCI specialist [33]. Communicatie binnen dit team is belangrijk om tot een goed ontwerp van het systeem te komen.

Uit de hierboven besproken principes, vloeien enkele basistechnieken en -stappen voort. Deze technieken worden besproken in sectie 3.4. We gaan eerst verder met een onderzoek naar de opbouw van de UCD levenscyclus in sectie 3.3.

3.3 UCD levenscyclus

Zoals reeds werd aangehaald in sectie 3.1, is er geen unieke overeenkomst te vinden in de literatuur over wat een UCD proces inhoudt. Dit houd ons echter niet tegen om de algemene structuur en opbouw van een UCD proces te onderzoeken. In deze discussie bouwen we dan ook verder op het model voorgesteld in ISO 13407. Dit model wordt afgebeeld in figuur 15.



Figuur 15: UCD levenscyclus volgens ISO 13407 [33]

Het proces in figuur 15 gaat van start met een initiële stap, namelijk het identificeren van de behoefte naar een UCD aanpak. Men kan beslissen om UCD aan te wenden voor een bepaald project door te kijken naar het doel van het te ontwikkelen systeem [33].

Verder zien we dat de UCD levenscyclus bestaat uit vier hoofdactiviteiten, namelijk:

- Begrijpen en specificeren van de context van gebruik
- Specificeren van de requirements van de gebruiker en de klant
- Produceren van ontwerpen
- Evalueren van de ontwerpen ten opzichte van de vooropgestelde requirements

Deze activiteiten worden steeds herhaald totdat het systeem voldoet aan de vooropgestelde eisen. Deze herhaling gebeurd op een iteratieve manier, dit wil zeggen dat men beroep doet op ervaring en feedback uit vorige iteraties voor het aanpassen en verfijnen van het ontwerp in toekomstige iteraties. De gebruiker is hierbij een belangrijke bron van informatie, hij of zij is namelijk actief betrokken bij elk van de hierboven genoemde activiteiten. Een herhaling van de cyclus wordt ook wel een “iteratie” genoemd.

Vooraleer er van start gegaan wordt met deze cyclus, is het nodig een plan op te stellen voor verloop van het ontwikkelproces [33]. Dit plan identificeert ondermeer de duur van de gebruikte iteraties, de technieken en artefacten die gebruikt zullen worden, de samenstelling van het projectteam en een beschrijving van de mijlpalen van het project. Het doel van dit plan is ondersteuning bieden voor het ontwikkelproces door het incorporeren van voldoende tijd voor de verschillende UCD activiteiten. Alle taken die zullen plaatsvinden tijdens het proces worden onderzocht, om vervolgens een keuze te maken uit een reeks technieken die zullen gebruikt worden bij de uitvoering ervan [33].

In onderstaande secties worden de vier basisactiviteiten van een UCD proces besproken.

3.3.1 Begrijpen en specificeren van de context van gebruik

De eerste stap in de iteratieve cyclus, is een onderzoek naar de context waarin het systeem gebruikt zal worden. Een duidelijk begrip van deze context vormt de basis voor het ontwerp van het systeem, alsook de evaluatie ervan. De context van gebruik bestaat volgens ISO 13407 [33] uit drie onderdelen, namelijk:

- De *gebruikers* van het systeem
- De *omgeving* waarin de gebruikers met het systeem werken
- De *taken* die zij uitvoeren wanneer ze gebruik maken van het systeem

Wanneer men de context van gebruik gaat onderzoeken, wordt er een uitgebreid en gedetailleerd profiel opgesteld van de eindgebruikers en worden hun taken en werkomgeving onderzocht. Dit gebeurt tijdens het gebruikersonderzoek, de taakanalyse en het omgevingsonderzoek.

Het is van groot belang dat de gebruikers van de software op voorhand goed begrepen worden. Een goede notie van de kenmerken van de gebruikers helpt er immers voor te zorgen dat het eindproduct een goede bruikbaarheid kent. De informatie die verzamelt wordt, bestaat volgens McCracken et al. [39] uit twee delen.

Ten eerste is er de informatie die het karakter en de voorkeuren van de gebruiker beschrijft. Deze gegevens hebben betrekking op vier eigenschappen [40]:

- **Leerstijl van de gebruikers**
  Hoe gaan de gebruikers om met nieuwe software? Gaan ze direct aan de slag of zullen ze eerst de handleiding doornemen?
- **Ervaring met verschillende tools en interactietechnieken**
  Niet alle gebruikers hebben dezelfde ervaring met of kennis over bepaalde interfacetechnieken. Dit is belangrijk om weten bij het ontwerpen van een user interface. Het gebruik van niet gekende interactietechnieken kan ervoor zorgen dat de gebruiker minder productief zal zijn, of dat er verwarring optreedt.
- **Fysieke eigenschappen**
  De fysieke eigenschappen van de eindgebruikers kunnen alsook een invloed hebben op het design. Fysieke eigenschappen houden bijvoorbeeld leeftijd, geslacht, en mogelijke handicaps in. Doormiddel van de kennis over deze eigenschappen kan er bijvoorbeeld een beslissing gemaakt worden over het taalgebruik, het gebruik van lettertypes en kleuren of de keuze van in- en uitvoerapparaten.
- **Verschillen in cultuur**
  Culturele verschillen kunnen gegrond zijn op geografische locatie van de gebruikers, hun opleiding, hun beroep, en andere. Deze factoren hebben invloed op bijvoorbeeld de ondersteuning van verschillende talen in de user interface, het gebruikte taalniveau en het gebruik van terminologie uit het domein.

Ten tweede is er de informatie in verband met de kenmerken en het gebruik van de software. Betreft het een kritieke applicatie (bv. medische software) of is het doel eerder ontspanning? Wat is de context waarin de applicatie gebruikt wordt? Wordt de gebruiker vaak gehinderd tijdens zijn taken (bv. lawaaierige omgeving)? Antwoord op deze vragen verkrijgt men door het uitvoeren van een omgevingsonderzoek. Er wordt beschreven welke hard- en software er gebruikt wordt, evenals andere materialen zoals pen en papier of een tekentafel. Ook de sociale en fysieke omgeving wordt onderzocht. [33]

Tijdens de taakanalyse tenslotte, wordt er een onderzoek uitgevoerd naar de taken die de gebruiker uitvoert wanneer hij met het systeem werkt. Met behulp van de resultaten van dit onderzoek, kunnen de ontwikkelaars te weten komen waarvoor de gebruikers het systeem zullen gebruiken. Taakanalyse of “task analysis” biedt een raamwerk dat methoden en technieken aanreikt ter ondersteuning van dit onderzoek. [41]

De technieken en artefacten die gebruikt kunnen worden tijdens deze eerste fase van UCD, staan opgesomd in tabel 1. Een verdere beschrijving wordt gegeven in sectie 3.4.

Technieken	**Artefacten**
• Interviews [42][40][43][44][45][46][47][48] • Surveys [43][44][45][46][48] • Questionnaires [43][45][48] • Focus groups [42][43][44][47][48] • Etnografie [42][40][43][45][49] • Contextual Inquiry [40][46][47][50] • Taakanalyse [41] [40][46][47][48] • Bestaande documentatie raadplegen [40][43][51][48] • Observaties [33][45][48]	• Personas [40][45][47][52] • Scenario’s [33][40][53][48] • Taakmodellen [40][43][54]

Tabel 1: Technieken en artefacten gebruikt tijdens de eerste fase in de UCD cyclus

3.3.2 Specificeren van de gebruikers- en organisatorische requirements

Het ontwerpen en ontwikkelen van software wordt steeds voorafgegaan door een initiële fase waarin de vereisten voor het systeem bepaald worden. Deze vereisten worden ook wel “requirements” genoemd. Bij UCD is dit niet anders. Naast de traditionele functionele en technische requirements, worden tijdens een UCD proces ook de gebruikers- en organisatorische vereisten gespecificeerd [33]. Dit gebeurt tijdens de tweede fase in de UCD cyclus, in relatie met de omschrijving van de context van gebruik uit de eerste fase. Deze context kan namelijk een significante invloed hebben op de requirements van het systeem. Het betrokken personeel en eventueel andere partijen, de organisatie van het werk, de in- en uitvoermogelijkheden van de aanwezige systemen en de vereiste performance zijn onder andere elementen die het opstellen van de requirements kunnen beïnvloeden. [33][55]

De requirements omvatten ten eerste de vereisten voor en mogelijkheden van de software opdat de gebruiker deze succesvol kan gebruiken voor het uitvoeren van zijn taken. Deze beschrijving legt dus vast op welke manier de toekomstige software ondersteuning zal bieden voor de gebruiker. Ten tweede zijn er de organisatorische requirements, dit zijn eisen vanwege de organisatie die gebruik zal maken van de software. Onder andere legislatieve en statutaire vereisten zijn een onderdeel hiervan.

De beschrijving van de requirements dient als invoer voor het ontwerpen en evalueren van de user interface in de volgende fases van de cyclus. Bij het gebruik van de requirements zijn twee partijen betrokken, namelijk de gebruikers en de ontwikkelaars. De gebruikers zorgen voor verificatie en feedback, terwijl de ontwikkelaars de requirements gebruiken ter ontwikkeling van de interface. Het is dus nodig dat de requirements begrijpbaar zijn voor beide partijen. [55]

De lijst met requirements is geen statisch document, in die zin dat telkens wanneer er een nieuwe iteratie plaatsvindt van de UCD cyclus deze requirements verfijnt, aangepast of uitgebreid kunnen worden. Het is dikwijls zo dat de requirements pas correct geïnterpreteerd worden na een reeks van iteraties.

Mogelijke technieken en artefacten voor deze fase zijn wederom te vinden in tabel 2.

Technieken	**Artefacten**
• Interviews [42][40][43][44][45][46][47][48] • Surveys [43][44][45][46][48] • Questionnaires [43][45][48] • Focus groups [42][43][44][47][48]	• Requirements document [33] • Use cases [55] • Personas [40][45][47][52][48] • Scenario's [33][40][53][48] • Paper prototypes [37][33][50][48] • Index cards [22][48] • Sticky notes [56][48]

Tabel 2: Technieken en artefacten gebruikt tijdens de tweede fase in de UCD cyclus

3.3.3 Produceren van ontwerpen

Tijdens de derde fase van de cyclus, ook wel "design" genoemd, wordt de uiteindelijke user interface van de applicatie ontworpen. Dit gebeurt ondermeer aan de hand van data verzameld tijdens de eerste twee fases. Ook reeds opgedane expertise uit vorige projecten, richtlijnen uit stijlgidsen en standaarden, en kennis uit het domein van de psychologie en ergonomie kunnen dienen als invoer voor de designfase. [33]

Om ervoor te zorgen dat de ontwikkelaars de user interface op een correcte manier ontwerpen, is het noodzakelijk tijdig feedback te krijgen van de gebruikers. Dit is mogelijk door reeds vroeg in het design mock-ups op te stellen van de interface. Met behulp van deze mock-ups kan de user interface vervolgens gesimuleerd worden voor de gebruikers. Deze aanpak reduceert het risico op het afleveren van een gebruikersinterface die niet voldoet aan de eisen van de gebruiker. Door te werken met "goedkope" voorstellingen van

de user interface gaat er geen tijd en geld verloren aan het bouwen van uitgebreide interfaces. Het steeds reviseren zorgt verder voor een maximale gebruiksvriendelijkheid van de uiteindelijke gebruikersinterface. Deze techniek wordt ook wel “Prototyping” genoemd. Verder kunnen de mock-ups gebruikt worden voor het voorstellen van verschillende oplossingen voor een probleem of als artefact ter ondersteuning van de communicatie binnen het ontwikkelteam. Om de vooruitgang tijdens de verschillende iteraties van de designfase te meten, is het noodzakelijk dat veranderingen vastgelegd en bijgehouden worden in een document. [33]

In tabel 3 worden de gebruikte technieken en artefacten tijdens het design opgesomd.

Technieken	**Artefacten**
• Prototyping [33][40][45][51][48] • User interface design [33][48][57] • Wizard of Oz [42][48] • Storyboarding [48]	• Mock-ups [33][56][42][48] • Prototypes [33][40][45][51][48] • Index cards [22][48] • Sticky notes [56][48] • Style guides [48][57]

Tabel 3: Technieken en artefacten gebruikt tijdens de derde fase in de UCD cyclus

3.3.4 Evalueren van ontwerpen ten opzichte van de vooropgestelde requirements

Evaluatie is een essentieel onderdeel van het UCD proces. Tijdens deze fase worden de ontwerpen getest en geverifieerd ten opzichte van de vooropgestelde eisen. Evaluatie moet uitgevoerd worden doorheen de volledige cyclus, en mag niet beperkt worden tot enkel de laatste fase. Volgens ISO 13407 [33] kan evaluatie gebruikt worden voor drie doeleinden, namelijk;

- Het verzamelen van feedback ter verbetering van het ontwerp
- Controleren of de doelstellingen bereikt zijn
- Het monitoren van het gebruik van het systeem

Evaluatie vindt dus zowel plaats tijdens het evalueren van de ontwerpen als tijdens het Prototypen en kan gebeuren door eindgebruikers of door aangewezen experts. Deze laatste optie is vooral nuttig wanneer het contact met de eindgebruikers moeilijk is (bijvoorbeeld tijdens een vakantieperiode of vanwege afstand). Echter evaluatie door de eindgebruikers mag zeker niet ontbreken tijdens de UCD cyclus.

Tabel 4 beschrijft de technieken en artefacten gebruikt tijdens de evaluatie.

Technieken	Artefacten
• Prototyping [33][40][45][51][48] • Thinking Aloud [56][45][58][48] • Meeting performance [59] • Retrospective testing [59] • Cognitive Walkthrough [59][60] • Heurisitische evaluatie [59][61] • Perspective based evaluatie [59] [62]	• Mock-ups [33][56][42][48] • Prototypes [33][40][45][51][48] • Personas [40][45][47][52] • Scenario's [33][40][53][48] • Use cases [55]

Tabel 4: Technieken en artefacten gebruikt tijdens de derde fase in de UCD cyclus

Wanneer het systeem voldoet aan de vooropgestelde vereisten, kan men verder gaan met de volledige implementatie ervan om zo te komen tot een volledig functioneel en afgewerkt product dat kan worden vrijgegeven aan de klant. Ook tijdens de implementatie wordt er regelmatig een usability test uitgevoerd, om te controleren of de uiteindelijke applicatie voldoet aan de vereisten.

Tijdens en na het uitrollen (of verspreiden) van de software naar de klant, staat het ontwikkelteam klaar om feedback te ontvangen. Deze feedback kan verkregen worden op verschillende manieren, zoals een forum met reacties, een ingebouwde functie in de software waarmee gebruikers hun problemen en feedback kunnen verzenden naar het ontwikkelteam, het uitvoeren van interviews of questionnaires, enzovoort.

3.4 UCD technieken, methoden en artefacten

In deze sectie worden de technieken, methoden en artefacten beschreven die gebruikt worden tijdens UCD.

3.4.1 Interviews, surveys en questionnaires

Tijdens het verzamelen van informatie over de context van gebruik (sectie 3.3.1) en bij het opstellen van de requirements (sectie 3.3.2) zijn er verschillende "ondervragende" technieken voorhanden, namelijk het afnemen van interviews en het uitvoeren van surveys en questionnaires.

Tijdens een interview worden er vragen gesteld aan de gebruikers, de klant, eventuele domein experts en andere stakeholders ter onderzoek naar de context waarin de software gebruikt zal worden of voor het opstellen van de requirements [48]. De directe communicatie bij een interview zorgt voor flexibiliteit bij het stellen van extra vragen of het uitdiepen van informatie. Een interview kan gestructureerd, semi-gestructureerd of ongestructureerd zijn [59]. Een gestructureerd interview bestaat uit een reeks vooraf gedefinieerde vragen waar niet van afgeweken wordt. Bij een ongestructureerd interview worden de vragen veelal

bepaald tijdens het interview zelf, gebaseerd op het profiel en de antwoorden van de geïnterviewde. Semi-gestructureerde interviews zijn een combinatie van deze twee en zijn de meest gebruikte methode bij UCD [48]. De vragen liggen in grote lijnen op voorhand vast, maar tijdens het interview kan er bijgestuurd worden en kan de interviewer de focus leggen op bepaalde aspecten. De uitkomst is steeds een verzameling kwalitatieve data die men achteraf analyseert. Interviews nemen typisch veel tijd in beslag, aangezien er meestal slechts één persoon tegelijk wordt geïnterviewd. Groepsinterviews, oftewel focus groups, worden besproken in sectie 3.4.3.

Surveys en questionnaires hebben dezelfde structuur als een gestructureerd interview [59], en kunnen al dan niet schriftelijk of mondeling afgelegd worden. De schriftelijke werkwijze heeft als voordeel dat er meerdere personen tegelijk ondervraagd kunnen worden en dus verkrijgt men sneller resultaat. Omdat er geen rechtstreekse communicatie hoeft plaats te vinden, kunnen de surveys en questionnaires verspreid worden via email of post over een grote populatie. Daarentegen is het feit dat er veelal gewerkt wordt met vaste, gesloten vragen nadelig voor de diepgang en flexibiliteit van het onderzoek. Surveys en questionnaires zorgen voor kwantitatieve data die automatisch geanalyseerd kan worden, wat zorgt voor een besparing van tijd. [48]

We merken op dat, om tot een meer nauwkeurige studie te komen, een combinatie kan gebruikt worden van de drie bovenstaande technieken. In de sociologie wordt deze werkwijze ook wel “triangulatie” genoemd. [44]

3.4.2 Observaties

Tijdens de eerste en de laatste fase in de UCD cyclus kunnen de gebruikers geobserveerd worden, meer bepaald tijdens het gebruikersonderzoek en de evaluatie. Een observator bestudeert de activiteiten en het gedrag van een gebruiker wanneer hij of zij met het systeem actief is. Dit systeem kan het huidige systeem zijn (observaties in fase 1), oftewel het toekomstig systeem (observaties in fase 4).

Er bestaan twee soorten van observaties, namelijk directe en indirecte observaties. Bij directe observaties is de observator steeds fysiek aanwezig in de ruimte waarin de gebruiker zich bevindt. Tijdens indirecte observaties wordt er videomateriaal gebruikt om de gebruiker te observeren. Dit heeft als voordeel dat men achteraf een meer nauwkeurige analyse kan uitvoeren. Echter kunnen er geen verdiepende vragen gesteld worden wanneer men de videobeelden bekijkt. [48]

Door het observeren van gebruikers is het mogelijk details en bijzonderheden waar te nemen die mogelijk onopgemerkt zouden blijven wanneer men enkel ondervragende technieken zou toepassen, zoals interviews of surveys (zie sectie 3.4.1).

De data verzameld tijdens de observaties in de eerste fase van de UCD cyclus, kan gebruikt worden bij het opstellen van scenario’s (zie sectie 3.4.8) [42]. Resultaten van observaties tijdens de evaluatie dienen als invoer voor een volgende iteratie van de cyclus.

3.4.3 Focus groups

Focus groups zijn groepsdiscussies of -interviews met acht tot twaalf deelnemers. Het onderwerp ter discussie wordt op voorhand vastgelegd. Tijdens de discussie worden

meningen gedeeld en ideeën uitgewisseld, terwijl een moderator bijstuurt en vragen stelt. [48][63]

Doordat er in groepsverband gewerkt wordt, worden de deelnemers gestimuleerd om creatief te denken en samen te werken. Het collectieve beeld over het onderwerp zorgt voor een beter begrip en inzicht. Verder nemen focus groups minder tijd in beslag dan het afnemen van individuele interviews. [48][63]

Volgens Stockton et al. [63] zijn er ook nadelen verbonden aan het gebruik van focus groups, namelijk:

- Het is mogelijk dat de mening van een bepaalde deelnemer beïnvloed wordt door andere deelnemers
- Het analyseren van de bekomen kwalitatieve data is tijdrovend
- Meer deelnemers zorgen ervoor dat het aantal vragen dat kan gesteld worden afneemt
- Kwaliteit van de bekomen informatie hangt af van de vaardigheden van de moderator
- Deelname van personen met auditieve, cognitieve of communicatiestoornissen bemoeilijkt het proces

Bij het gebruik van focus groups tijdens een UCD proces, kan er gediscussieerd worden over de context van gebruik, alsook kan er gebrainstormd worden over de requirements van het systeem. Volgens Gulliksen et al. [42] zijn focus groups vooral van toepassing bij het ontwikkelen van systemen bedoeld voor een algemeen publiek, bijvoorbeeld besturingssystemen zoals Microsoft WIndows of een mediaspeler zoals iTunes.

3.4.4 Etnografie

Etnografie is een onderzoeksmethode, afgeleid van socio[4] - en antropologie[5], die de menselijke interactie bestudeert in een sociale of culturele omgeving en in de context van sociale activiteiten. Dourish [49] definieert etnografie als volgt:

> *"An approach to social inquiry characterized by long-term immersive engagement with particular cultures in the effort to understand and explicate how they are experienced by their members. It is an interpretive, analytic practice."*

Voor het uitvoeren van een etnografie, worden dikwijls experts aangesproken. In de context van UCD, wordt etnografie aangewend om de eindgebruikers te observeren wanneer zij vertoeven in hun alledaagse werkomgeving. Deze manier van werken biedt de onderzoekers de mogelijkheid het systeem waar te nemen door de ogen van de gebruiker. Dit resulteert in een betere kennis over de manier waarop een gebruiker zijn taken uitvoert, de eventuele collaboratie die plaatsvindt tussen verschillende gebruikers en de verschillende applicaties die gebruikt worden. [64][40][49]

---

[4] Wikipedia - Sociologie, http://nl.wikipedia.org/wiki/Sociologie

[5] Wikipedia - Antropologie, http://nl.wikipedia.org/wiki/Antropologie

In contrast met observaties zoals besproken in sectie 3.4.2, gaat etnografie een stap verder door de gebruikers te observeren in hun dagelijkse omgeving. Deze omgeving maakt deel uit van de context van gebruik van het systeem en heeft een belangrijke invloed op de manier waarop gebruikers hun taken en activiteiten uitvoeren.

Er zijn verschillende soorten van etnografische studies, de vijf meest bekende methoden worden hieronder opgesomd [40].

- **Concurrent**
  Tijdens concurrente etnografie worden de gebruikers bestudeerd gelijktijdig met het ontwerp van de applicatie. Door de iteratieve aard van het designproces, kan het ontwerp telkens verfijnd of gereviseerd worden op basis van nieuwe informatie uit het etnografisch onderzoek.
- **Evaluatief**
  Deze manier van werken kan aangewend worden bij het ontwerp van een (gedeeltelijk) nieuwe user interface. Hierdoor kunnen onderzoekers verschillende ontwerpen vergelijken door het gedrag en werkwijze van de gebruikers waar te nemen. Tijdens evaluatieve etnografie wordt de focus meestal gelegd op een klein onderdeel van de totale interface, wat zorgt voor een besparing van tijd.
- **Quick and Dirty**
  Deze methode is handig wanneer men slechts een globaal beeld wil schetsen van de context waarin gebruikers hun taken uitvoeren, dit om ernstige problemen met de bruikbaarheid van de applicatie op te sporen. Door de beperkte diepgang kan deze methode toegepast worden in een relatief korte tijdspanne of in situaties waar er onvoldoende budget beschikbaar is voor een uitgebreide studie.
- **Rapid**
  Gelijkaardig aan de Quick and Dirty methode, maar dan uitgevoerd op een meer formele manier en met een meer specifieke focus. Rapid etnografie wordt gebruikt wanneer er weinig tijd ter beschikking is. Dikwijls worden er meerdere observators ingezet.
- **Gebruik van reeds voltooide studies**
  Door het gebruik van resultaten uit eerder uitgevoerde studies, kan men tijd en geld besparen. Het is echter niet vanzelfsprekend een studie te vinden die het gewenste domein accuraat benadert.

Een etnografische studie moet niet enkel beperkt blijven tot het observeren van gebruikers. Er kunnen ook vragen gesteld worden en de onderzoeker kan zelfs kiezen om te participeren in de taken van de gebruiker om zo een beter inzicht te krijgen in de context van een applicatie. Ook kan er video- en audiomateriaal verzameld worden (mits de nodige toestemming van de gebruiker) dat achteraf geanalyseerd kan worden. Het resultaat van een etnografische studie, is een verzameling data die, na interpretatie en analyse, gebruikt kan worden bij het ontwerpen van de applicatie [49]. Deze data is hoofdzakelijk kwalitatief en bevat omschrijvingen van de acties van de gebruiker, een beschrijving van de omgeving, enzovoort. Meting van de tijd of het aantal fouten dat de gebruiker maakt daarentegen, levert kwantitatieve data op.

Door het toepassen van etnografie bij het UCD gebruikersonderzoek, wordt het mogelijk kenmerken of eigenschappen van de gebruikers en hun omgeving op te sporen die over het hoofd zouden worden gezien indien men enkel interviews of standaard observaties zou aanwenden. Een belangrijk nadeel echter, is dat etnografie typisch veel tijd in beslag neemt,

wat dikwijls niet aanwezig bij een softwareproject omdat de klant zit te wachten op resultaten. Wanneer tijd een kwestie is, maakt men best gebruik van Rapid of Quick and Dirty etnografie (zie hierboven). Verder is het niet altijd mogelijk gebruikers te observeren in hun habitat, vanwege bijvoorbeeld privacy, geluidshinder of het feit dat men de gebruiker stoort tijdens het observeren. In dit geval kan er gekozen worden voor een gecontroleerde laboratoriumomgeving voor het uitvoeren van de etnografie. [65]

We merken op dat de resultaten van een etnografisch onderzoek slechts geldig zijn in een bepaalde temporele context. De technologie, de mensen en de omgeving kennen allen een voortdurende vooruitgang. Hierdoor is het niet vanzelfsprekend om te steunen op resultaten van eerdere etnografische studies. Echter, de theoretische bijdrage van deze studies is wel van waarde voor toekomstig onderzoek. [49]

3.4.5 Contextual Inquiry

Contextual Inquiry is een specifiekere vorm van etnografie. Dezelfde principes worden toegepast, maar de focus wordt eerder gelegd op het stellen van vragen. De designers gaan dus “in the field”, observeren de gebruikers en voeren conversaties met hen. Technieken zoals interviews en questionnaire (zie sectie 3.4.1) worden hierbij gebruikt. Op die manier wordt een beter zicht verkregen op de requirements van de gebruikers, omdat zij zelf uitleg kunnen geven bij de taken die ze uitvoeren. Het is efficiënter voor beide partijen wanneer dit in de levensechte situatie kan gebeuren, meer bepaald in de omgeving waarin er dagelijks wordt gewerkt.

Tijdens een Contextual Inquiry wordt er data verzameld volgend het meester/leerling model. De gebruiker neemt hier de rol in van de meester, de designer(s) de rol van leerling(en). Het is dus de bedoeling dat de gebruiker uitleg geeft aan de designers over zijn of haar werk.

De volgende vier principes vormen de basis voor Contextual Inquiry [66]:

- **Context**
  Het is essentieel dat de gebruikers geobserveerd en ondervraagd worden in de context van hun dagelijkse activiteiten.
- **Partnerschap**
  De designers werken nauw samen met de gebruikers met als doel het werk van gebruikers zo grondig mogelijk te begrijpen. Periodes van observatie worden afgewisseld met conversaties.
- **Interpretatie**
  Contextual Inquiry heeft slechts zin wanneer de acties en het gedrag van de gebruiker correct geïnterpreteerd worden. Regelmatig wordt er, in overleg met de gebruiker, gecontroleerd of de bevindingen van de observator correct werden waargenomen.
- **Focus**
  Door het leggen van focus op bepaalde aspecten van het werk van de gebruiker, kan de observator extra details waarnemen.

3.4.6 Taakanalyse

Het uitwerken van de user interface van een computerapplicatie gebeurt op een zeer laag niveau, de taken die de gebruiker zal uitvoeren met behulp van deze applicatie worden echter gespecificeerd op een relatief hoog niveau. Daartoe is een manier noodzakelijk die kan zorgen voor een mapping van de user interface naar de taken van de gebruiker. Taakanalyse is hiervoor geschikt. [41]

Taakanalyse vindt plaats tijdens de eerste fase van de UCD cyclus en biedt een raamwerk waarmee de acties en cognitieve processen die betrekking hebben op het uitvoeren van taken binnen het huidige systeem op een gestructureerde manier geanalyseerd en uitgediept kunnen worden [48]. Het doel van deze studie is een beter beeld te krijgen van de menselijke activiteit en de reeks taken en functies die het systeem zal moeten ondersteunen, om vervolgens tijdens de designfase deze informatie te kunnen gebruiken om de user interface op te bouwen zodat deze taken effectief ondersteunt worden [41][48]. Verder kan taakanalyse gebruikt worden voor het opdelen van functionaliteit tussen het systeem en de gebruiker, met andere woorden welke taken neemt het systeem voor zijn rekening, en welke de gebruiker.

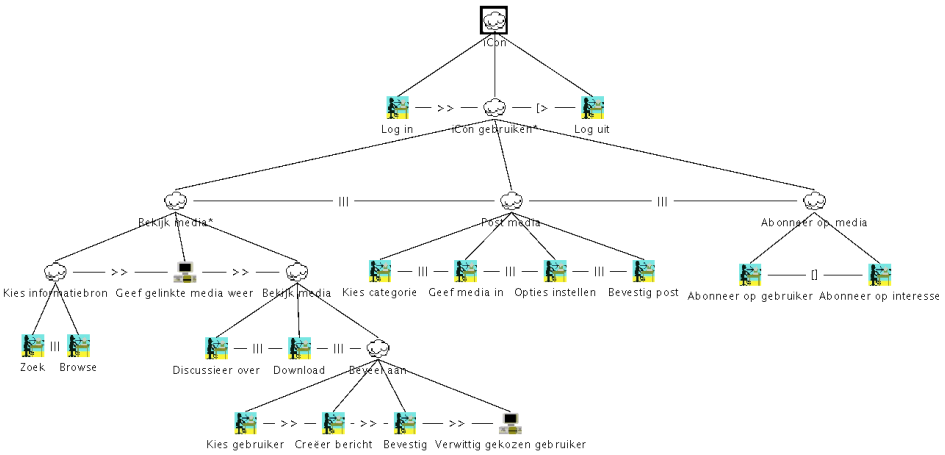
We merken het verschil op tussen taakanalyse enerzijds, en specificatie van de requirements anderzijds. Tijdens de taakanalyse wordt er een onderzoek uitgevoerd naar de manier waarop gebruikers hun taken uitvoerden vóór dat het nieuwe systeem geïntroduceerd werd in hun werkomgeving. Belangrijke aandachtspunten hierbij zijn de hoofddoelen en duur van hun activiteiten en de eigenschappen van hun taken die mogelijk een invloed hebben op de bruikbaarheid van het systeem. Requirements specificatie daarentegen, draait voornamelijk rond het opstellen van een reeks vereisten vanwege de gebruiker en de klant, opdat het systeem voor hen bruikbaar is. Hieronder vallen bijvoorbeeld de functies die het systeem moet ondersteunen, de mogelijke schaalbaarheid en eventuele legislatieve vereisten. [33]

Er zijn verscheidene varianten van taakanalyse, waaronder Hierarchical Task Analysis (HTA) [41][48][67] en cognitieve methoden [41][67]. De meest gebruikte variant is HTA [68].

Bij Hierarchical Task Analysis worden de taken van de gebruiker ontbonden in subtaken, acties en operaties en dit op een top-down manier [41][48][67]. Het hoogste niveau van de hiërarchie bestaat uit het doel van de gebruiker. In lagere niveaus wordt dit doel ontleed in de taken, acties en operaties nodig om dit doel te bereiken. Hierbij kan de diepgang van de ontbinding gekozen worden aan de hand van de complexiteit van de taken [67] en rekening houdend met de beschikbare tijd en het budget dat voor handen is [41].

Voor het voorstellen van de taken van de gebruiker, kan er beroep gedaan worden op verschillende notaties. De resultaten van HTA worden meestal grafisch voorgesteld door middel van zogenaamde taakmodellen. Deze modellen geven een overzicht van de taken en door de eenvoudige notatie kunnen ze snel begrepen worden door alle leden van het team, ook de gebruikers. Het gebruik van taakmodellen, zoals de ConcurTaskTrees (of CTT) notatie [69], zorgt voor een visuele representatie van de acties die een gebruiker onderneemt bij het uitvoeren van een taak. Vanwege het feit dat er gewerkt wordt met een visuele representatie van de acties van de gebruiker, biedt CTT een goede ondersteuning van het UCD proces. Wegens de multidisciplinaire aard van UCD, is het namelijk noodzakelijk dat de gebruikte notatie begrepen kan worden door mensen met een verschillende achtergrond en met verschillende kennis. Verder is CTT geschikt voor het modelleren van een groot aantal taken vanwege de compacte notatie. Ten slotte zorgt de consistente en eenduidige notatie voor het vermijden van misverstanden bij het

interpreteren van het model. [69] Bij het opstellen van een CTT kan er gebruik gemaakt worden van de informatie aanwezig in de gecreëerde scenario's. De CTT beschrijft dan de opeenvolging van acties nodig om een taak te voltooien. Figuur 16 toont een voorbeeld van een CTT, ontworpen tijdens het project van het vak Gebruikersgerichte Systeemontwikkeling (GGSO) [40]. Meer informatie over ConcurTaskTrees, zoals details over de notatie en dergelijke, is te vinden in [69].

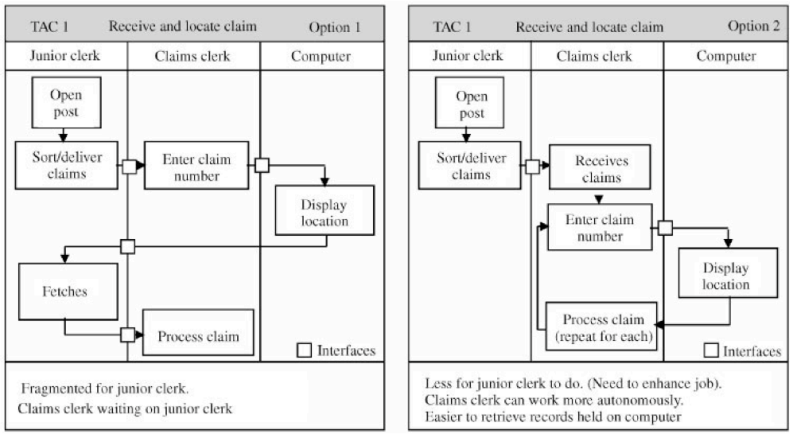


Figuur 16: Voorbeeld van een ConcurTaskTree

Het creëren van bijvoorbeeld een CTT kan gebeuren door middel van tools. Voor CTT bestaat er de tool Teresa[6]. Een andere tool ter ondersteuning van het proces van taakanalyse, is TaskArchitect [68]. Deze tool maakt gebruikt van Hierarchical Task Analysis voor het opstellen van een overzicht van de taken van de gebruiker.

Tijdens de taakanalyse is het tevens mogelijk een allocatie van de functionaliteit uit te voeren (zie sectie 3.4.6). Het weergeven van deze allocatie kan gebeuren door een zogenaamd "Allocation of Function" diagram. In zulk diagram zijn de verschillende entiteiten binnen het systeem afgebeeld, samen met hun verantwoordelijkheden. Figuur 17 geeft een voorbeeld van zo een diagram. [48]

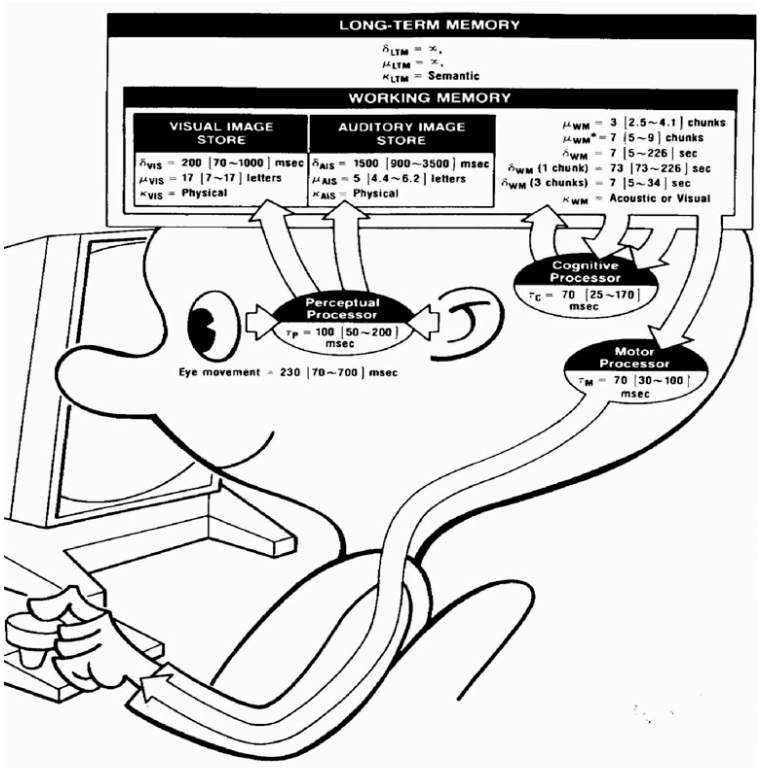
[6] Teresa - Tranformation Environment for Interactive Systems Representations, http://giove.isti.cnr.it/teresa.html



Figuur 17: een Allocation of Function diagram [48]

In [41] wordt er getwijfeld aan de bruikbaarheid van HTA binnen een HCI context. Het feit dat HTA zich focust op het systeem (aard is system-centric) en de cognitieve processen van de gebruiker beschouwt als een “zwarte doos”, maakt dat deze techniek het niet toelaat rekening te houden met de complexe sociale en fysieke context waarin de taken worden uitgevoerd. Cognitieve technieken voor taakanalyse kunnen hiervoor aangewend worden.

Het domein van cognitieve modelering bestudeert de menselijke cognitie, relevant bij het uitoefenen van taken en activiteiten. De “Model Human Processor” (MHP), het basismodel voor de cognitieve technieken, stelt de interactie van een persoon met zijn omgeving voor als drie op elkaar inwerkende processors, namelijk een perceptuele, een motorische en een cognitieve processor (zie figuur 18) [70]. Verder bestaat het systeem uit verschillende geheugens die in verbinding staan met elkaar en met de processors. MHP kan toegepast worden voor het modelleren van verschillende acties, van het oprapen van balpen tot bijvoorbeeld het gebruik van de zoekfunctie van een computer.



Figuur 18: De Model Human Processor [70]

Het model in zijn pure vorm is echter niet bruikbaar voor het uitvoeren van een taakanalyse. Daarom stelde de bedenkers van MHP een nieuw model voor op basis van MHP, namelijk GOMS (Goals, Operators, Methods and Selection rules) [41][59]. GOMS beschrijft taken op een zeer laag niveau, namelijk volgens doelen, operators, methoden en selectieregels. In tegenstelling tot HTA, dat taken op een hoog niveau beschrijft, werkt GOMS op het “keystroke” niveau. Volgens Hochstein [71] kan GOMS toegepast worden op drie verschillende manieren, namelijk:

1. **Voorspellend (predictive)**
   GOMS kan gebruikt worden voor het voorspellen van de tijd die een gebruiker nodig zal hebben om een bepaalde taak uit te voeren. Dit kan nuttig zijn om een vergelijking te maken van verschillende user interface ontwerpen. Deze voorspellingen vermijden het gebruik van veel tijd en grote budgetten, omdat het niet steeds nodig is gebruikers te betrekken bij het onderzoek.

2. **Omschrijvend (descriptive)**
   Voor een beschrijving van de taken die een systeem moet ondersteunen, kan tevens gebruik gemaakt worden van GOMS. Op die manier verkrijgt men een beter inzicht op de manier waarop een gebruiker een taak uitvoert.

3. **Voorschrijvend (prescriptive)**
   Een derde toepassing van GOMS is het aanleren van taken aan de gebruiker. Concreet kan dit toegepast worden bij het ontwikkelen van hulpfuncties of trainingsprogramma’s.

GOMS kan toegepast worden tijdens de eerste fase van de UCD cyclus bij het modelleren van de manier waarop gebruikers hun taken uitvoeren binnen het huidige systeem (omschrijvend). Verder kan GOMS ook aangewend worden bij het ontwerpen en evalueren van de user interface (voorspellend).

Nadelen van GOMS zijn ondermeer het feit dat er enkel rekening gehouden wordt met de ervaren gebruiker bij het voorspellen van performance, terwijl het systeem in de meeste gevallen een variërend doelpubliek kent, en dat het modelleren van parallelle taken niet mogelijk is. [71]

Cognitive Task Analysis (CTA) is een andere cognitieve techniek voor taakanalyse. CTA taakmodellering is abstracter en kent een hoger niveau dan GOMS [41]. CTA oogt op het uitwerken van een volledige representatie van het domein van het systeem, daarom is voor het uitvoeren van CTA specifieke kennis en expertise vereist, zoals contextual inquiry, interviews en etnografie. Door het multidisciplinair karakter van UCD zijn deze factoren ter beschikking. CTA neemt veel tijd in beslag en is een complexe bezigheid, daarom wordt het in de praktijk weinig gebruikt. [41]

Doordat methoden en technieken voor taakanalyse steeds complexer en uitgebreider werden door de jaren heen, is het niet eenvoudig om deze in de praktijk toe te passen, vanwege een gebrek aan tijd, budget, expertise of kennis. Daarom is het in de toekomst nodig de bruikbaarheid en integratie van deze technieken te verbeteren, in de vorm van toolkits en automatisering. [41]

3.4.7 Bestaande documentatie gebruiken

Tijdens de taakanalyse kan er, indien er reeds bestaande software is, gebruik gemaakt worden van de documentatie horende bij deze software [72]. Door het analyseren van handleidingen en dergelijke kan er meer informatie over de taken van de gebruiker bekomen worden, bijvoorbeeld welke functies aanwezig zijn en wat het doel is van de software.

3.4.8 Personas en scenario's

Het product van de gebruikers- en taakanalyse in fase 1 van de cyclus, is een reeks data die de eigenschappen, kenmerken en taken van de gebruiker reflecteert. Deze data moet opgeslagen worden opdat deze efficiënt kan ingezet worden tijdens toekomstige fases van de ontwikkelcyclus. Het opslaan kan gebeuren door middel van zogenaamde artefacten. De meest gebruikte artefacten tijdens de eerste twee fases van de UCD cyclus zijn personas en scenario's.

Wanneer er een softwareapplicatie ontwikkeld wordt, moet er rekening gehouden worden met de verschillen tussen de eindgebruikers ervan. Het is praktisch onmogelijk om een profiel op te maken van iedere gebruiker apart. Personas bieden hiervoor een oplossing. Een persona is een fictief persoon die een bepaalde gebruikersgroep voorstelt of representeert [73]. De informatie bekomen tijdens het gebruikersonderzoek wordt opgenomen in verschillende personas. Deze personas kunnen vervolgens gebruikt worden bij het ontwerpen en evalueren van de user interface. Figuur 19 toont een voorbeeld van een persona.

Figuur 19: een voorbeeld van een persona[7]

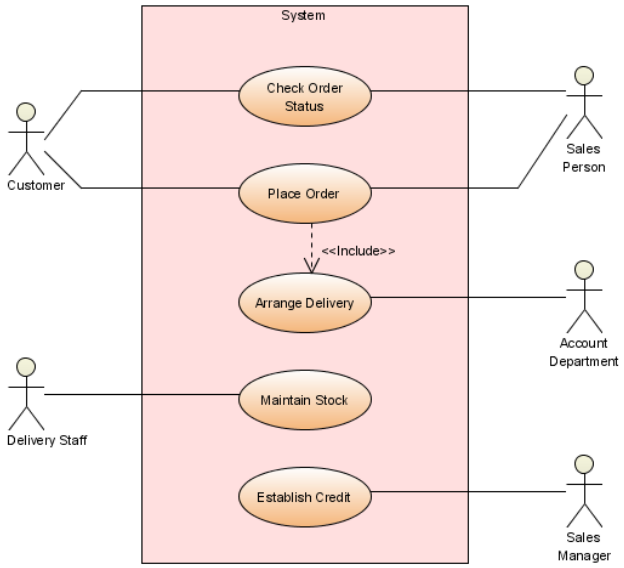
Door het gebruik van personas krijgen de ontwikkelaars bij het ontwerpen de mogelijkheid efficiënt rekening te houden met een groot aantal gebruikersgroepen. Een softwareproject telt typisch een tiental personas, telkens bestaande uit het doel van de persona, enkele persoonlijke details, een foto, een lijst met vaardigheden en een karakterbeschrijving. Personas kunnen gebruikt worden tijdens de design- en evaluatiefase.

De gecreëerde personas kunnen vervolgens ingezet worden bij het opstellen van scenario's. Een scenario is een gedetailleerde beschrijving, in verhalende stijl, van de handelingen die een bepaalde gebruiker (voorgesteld door een persona) uitvoert tijdens het uitoefenen van een taak. Volgens Bodker [74] zijn er drie situaties waarvoor scenario's kunnen ingezet worden, namelijk het voorstellen en identificeren van oplossingen, het illustreren van alternatieve oplossingen en het opsporen van eventuele problemen. Scenario's scheppen dus de mogelijkheid voor het UCD team om samen met de gebruikers en de designers te communiceren over mogelijke situaties van gebruik van de software, rekening houdend met de eindgebruikers.

[7] http://www.chopsticker.com

3.4.9 Use cases

Voor het specificeren van de gebruikersrequirements in fase 2 van de UCD cyclus, kan gebruik gemaakt worden van zogenaamd “use cases”. Een use case beschrijft mogelijke interacties tussen een systeem en één of meerdere actoren, veroorzaakt door een impuls vanwege één van deze actoren [55]. Use cases kunnen zowel tekstueel als grafisch voorgesteld worden. De grafische voorstelling heeft als voordeel dat deze makkelijker te begrijpen is en een beter overzicht biedt van de vereisten. Een voorbeeld van een use case diagram is te zien in figuur 20.



Figuur 20: Voorbeeld van een use case diagram

Use cases zijn echter niet het meest geschikte middel bij UCD. Ze zijn namelijk opgesteld vanuit een technisch standpunt en houden geen rekening met de behoeften van de gebruikers. Verificatie van de requirements vanwege de gebruikers is noodzakelijk opdat deze correct gespecificeerd kunnen worden. Dit is niet eenvoudig vanwege de technische aard van use cases. [55]

De kern van het probleem ligt in het feit dat use cases begrijpbaar moeten zijn voor zowel de gebruikers als de ontwikkelaars. De behoeften van de gebruikers moeten dus vertaald worden naar gebruikersrequirements. In [55] wordt hierom een proces voorgesteld dat werkt volgens een reeks notaties, waartussen getransformeerd wordt. Er wordt gestart met een sequentiediagram dat de verschillende activiteiten weergeeft die een gebruiker doorloopt bij het uitvoeren van een taak. Vervolgens wordt een zogenaamde “user needs table” opgesteld, die refereert naar het eerder genoemde sequentiediagram. Een user needs table detailleert elke stap uit het sequentiediagram door het beschrijven van alle mogelijkheden en problemen horende bij die stap. Dit document wordt uiteindelijk omgezet naar een reeks use cases die op een coherente manier ondersteuning bieden aan de ontwikkelaars, werkende vanuit het standpunt van de gebruikers. Figuur 21 toont een voorbeeld van een user need table.

Task sequence:	Problems and possibilities:
Step 1: When trapped in an elevator, passenger makes an emergency alarm.	• Passengers want to get out of the elevator as soon as possible • All kinds of passengers must be able to make an alarm call (blind, foreigners etc.) • Sometimes passengers may make false alarms unintentionally. • Passengers may be in panic. • Passengers need instant confirmation that they have created a connection to the service centre operator and that they are going to get help.
Step 2: Unoccupied service centre operator receives the emergency alarm call and asks for information.	• Different versions and types of remote monitoring systems. • Passenger is the only information source. • Service centre operator does not notice the emergency alarm call.
Step 3: Service centre operator completes transmission of information to the system and sends it to the area serviceman.	• Laborious phase for the service centre operator. • Simultaneous calls must be differentiated. • Serviceman cannot see all information. • Inadequate information from a site system. • Possibility: Instructions as to how to operate the system. • Possibility: Possibility to open phone line from Call Centre to the elevator.
Step 4: Service centre operator calls the serviceman and reads the description of the failure.	• Extra work for the service centre operator.

Figuur 21: user needs table [55]

Door het gebruik van de verschillende modellen, wordt het mogelijk de requirements te verifiëren bij de gebruikers, terwijl deze ondersteunt worden door use cases voor gebruik tijdens de ontwikkeling.

3.4.10 Index cards en sticky notes

Index cards en sticky notes bieden de mogelijkheid om op een bondige manier informatie weer te geven onder de vorm van kleine notities. Ze kunnen onder andere gebruikt worden om de resultaten van het context-onderzoek uit fase 1 samen te vatten of om de requirements uit fase 2 voor te stellen. Tijdens het ontwerp van de user interface kunnen er index cards en sticky notes gebruikt worden om functies of mock-ups van de interface te illustreren. Bij de laatste fase in de cyclus tenslotte, is het mogelijk ontbrekende of incorrecte features te noteren om een overzicht te krijgen van de doelstellingen van de volgende iteratie van de cyclus.

Wanneer de notities bevestigd worden op een wand of bord, kunnen deze gegroepeerd en gesorteerd worden in samenwerking met de gebruikers. Deze techniek faciliteert de collaboratie binnen het team en kan zorgen voor een mentaal model van het systeem. Zo kunnen er zogenaamde storyboards (zie sectie 3.4.14) en affinity diagrams opgesteld worden. In een affinity diagram [48] worden bepaalde gerelateerde features (bijvoorbeeld samenhorende functies van het systeem) gegroepeerd door het schikken van de notities. Het gebruik van kleuren kan zorgen voor categorisatie en prioritering. In figuur 22 is het gebruik van sticky notes te zien bij het opstellen van een affinity diagram.

Figuur 22: Affinity diagram opgebouwd uit sticky notes

3.4.11 Prototyping

Het proces van Prototyping vindt plaats tijdens de derde fase in de UCD cyclus. Volgens Redmond-Pyle [75] is Prototyping

> *"het bouwen van prototypes en informeel onderzoeken of deze bruikbaar zijn voor de eindgebruiker".*

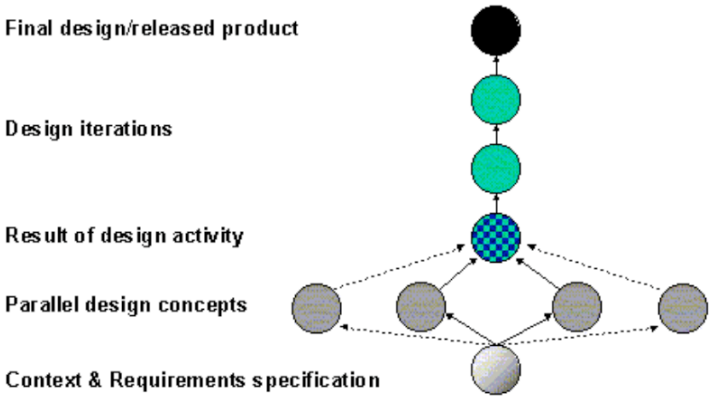
Prototyping omvat dus niet enkel het bouwen van prototypes, maar ook de evaluatie ervan. Het is een iteratief en incrementeel proces, in die zin dat tijdens elke iteratie verbeteringen aangebracht worden of eventueel nieuwe functies worden geïntroduceerd in het systeem. De prototypes onder ontwikkeling evolueren van een simpele mock-up in een vroeg stadium in de ontwikkeling, tot een volledig werkende (simulatie van een) applicatie. "Werkende" in die zin dat er volledige interactie mogelijk is, met als doel het verzamelen van feedback. Figuur 23 geef de evolutie van prototypes gedurende de UCD cyclus weer. [48]

Figuur 23: evolutie van prototypes gedurende de UCD cyclus

Met behulp van mock-ups en Paper Prototypes kunnen de ontwikkelaars initiële (experimentele) ontwerpen tonen aan de gebruiker en eventueel alternatieve ontwerpen

illustreren. De gebruiker kan vervolgens deze ontwerpen beoordelen en eventueel uittesten indien er gewerkt wordt met een interactief prototype. Doordat deze initiële prototypes “goedkoop” zijn, kunnen ze makkelijk en snel aangepast wordt op basis van feedback van gebruikers, designers of human factors experts. Deze aanvankelijk snelle ontwikkeling wordt ook wel Rapid Application Development (RAD) genoemd. [75][48]

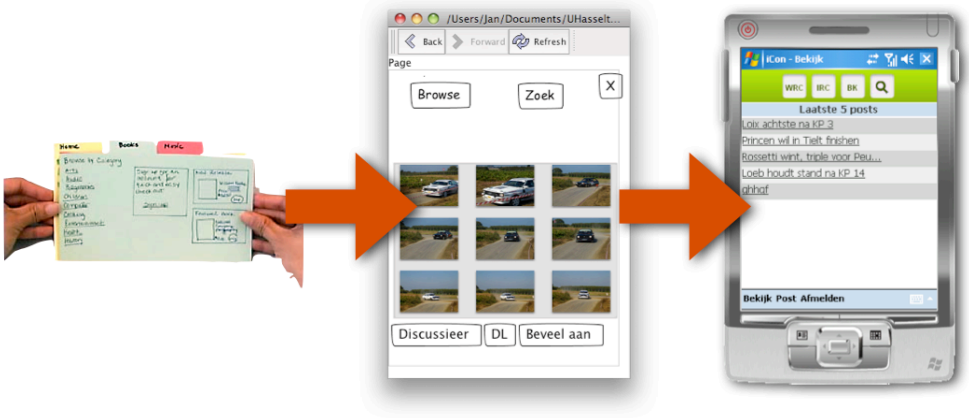
Voor het uittesten van verschillende design-oplossingen, kan men “Parallel Design” gebruiken. Bij Parallel Design gaan verschillende UI ontwerpers, meestal in groepjes van twee, gelijktijdig werken aan verscheidene ontwerpen van de user interface. Deze “voorstellen” worden vervolgens geëvalueerd, om uiteindelijk voor één oplossing te kiezen. Deze evaluatie gebeurt op basis van de scenario’s uit fase 1 van de cyclus. [42][43][48] Figuur 24 toont de context van Parallel Design in het UCD proces.



Figuur 24: Parallel Design in het UCD proces [43]

De evaluatie van de low- en high-fidelity prototypes gebeurt tevens in samenwerking met de gebruikers. Door de gebruikers reeds in een vroeg stadium van de ontwikkeling te betrekken, worden eventuele problemen snel opgelost. [75][48]

Figuur 25 geeft een voorbeeld van de evolutie van een prototype voor een user interface op een PDA.



Figuur 25: Evolutie prototypes voor user interface voor PDA (Paper prototype -> Low-Fidelity Prototype -> High-Fidelity Prototype)

Om bruikbaar te zijn voor evaluatie en als invoer voor verder design, moet een prototype volgens Redmond-Pyle [75] voldoen aan drie criteria:

- Coverage van een gedefinieerd deel van de interface
- De mogelijkheid tot interactie
- Het toelaten van evaluatie

Er zijn verschillende vormen en niveaus van prototypes. Deze kunnen opgedeeld worden volgens een reeks kenmerken, namelijk [75]:

- Scope
- Diepgang
- Evolutiemogelijkheden
- Niveau van realisme

De *scope* duidt aan of een prototype al dan niet een ontwerp voor de volledige interface inhoudt. De designers kunnen er namelijk voor opteren om tijdens een zekere iteratie zich enkel toe te spitsen op een bepaalde feature of reeks van features. De *diepgang* bepaalt de mogelijkheden van een prototype. In een horizontaal prototype worden vele features ontworpen, maar met weinig diepgang of functionaliteit. Verticale prototypes daarentegen, leggen de focus op een bepaald deel van de interface en leveren hiervoor een gedetailleerde implementatie. De *evolutiemogelijkheden* duiden aan of een prototype al dan niet kan doorgroeien tot de uiteindelijke afgewerkte applicatie, of dat het eerder een zogenaamd “throw-away” prototype betreft dat bijvoorbeeld enkel gebruikt wordt voor het controleren op bruikbaarheid van experimentele functies of interactietechnieken. Doorheen de fase van Prototyping wordt er gewerkt met prototypes met een evoluerend *niveau van realisme*. Initieel ontwerpen de designers een interface die de globale functionaliteit

reflecteert, dit meestal in de vorm van een prototype op papier. Naarmate het project vordert, krijgen de prototypes steeds meer kenmerken van de uiteindelijke user interface.

3.4.12 User Interface design

Het eigenlijke ontwerp van een user interface gebeurt door UI designers, in samenwerking met onder andere psychologen en grafici [57]. Er wordt gestart vanaf de initiële mock-ups en Paper Prototypes, om vervolgens iteratief verder te werken op basis van resultaten van de evaluatie van de interface. Bij het ontwerpen wordt er gewerkt volgens bestaande standaarden en stijlgidsen (style guides). Hierdoor verhoogt de kans op een consistente interface en bespaart men tijd bij de ontwikkeling van de interface [48].

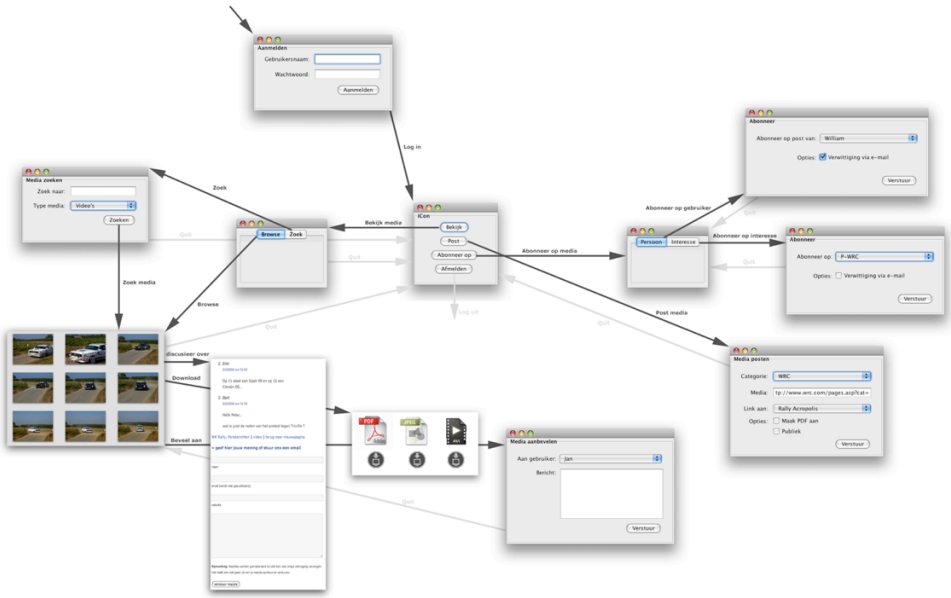
3.4.13 Wizard of Oz

Bij het proces van Prototyping (zie sectie 3.4.11) bestaat de kans dat een bepaald ontwerp, al dan niet in een ver gevorderd stadium, afgekeurd wordt. De oorzaak kan onder andere slechte performance zijn, of het feit dat de bruikbaarheid niet voldoende is. Wanneer er dan veel tijd gespendeerd werd aan dit ontwerp, is deze tijd verloren en stijgen de kosten. De Wizard of Oz (WOZ) biedt een oplossing voor dit probleem. WOZ zorgt ervoor dat bepaalde functionaliteit van een systeem gesimuleerd kan worden, waardoor men deze niet hoeft te implementeren. Deze werkwijze is handig voor complexe functies die typisch moeilijk en tijdrovend zijn om te ontwikkelen. [48][76]

Concreet werkt WOZ als volgt. De opstelling bestaat telkens uit een testpersoon (de gebruiker), een computer en een zogenaamde Wizard (iemand van de ontwikkelaars). Wanneer de gebruiker nu bijvoorbeeld een zoekterm ingeeft in de user interface en de zoekactie start, krijgt hij of zij niet rechtstreeks de zoekresultaten te zien. De zoekterm zal eerst naar de Wizard gestuurd worden. Deze persoon kan dan bijvoorbeeld zelfgekozen zoekresultaten teruggeven, oftewel de zoekterm omvormen en uitvoeren. Uiteindelijk wordt het resultaat getoond op het scherm van de gebruiker. Tijdens dit proces heeft de gebruiker geen weet van het feit dat het systeem slechts een simulatie is, achter de schermen bestuurt door een menselijk persoon. Deze werkwijze zorgt ervoor dat er verschillende scenario's getest kunnen worden zonder deze volledig te implementeren tot op bijvoorbeeld database niveau.

3.4.14 Storyboarding

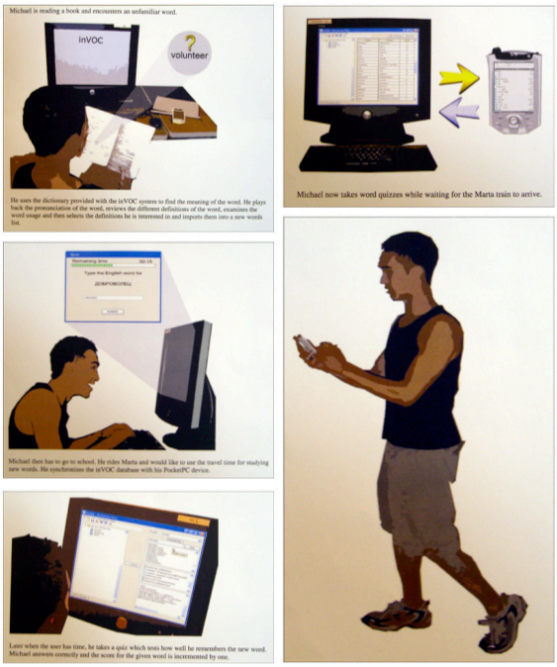
Tijdens het beginstadium van de designfase kan Storyboarding gebruikt worden om een totaaloverzicht te geven van de applicatie. Een storyboard illustreert de relatie tussen in- en uitvoer van het systeem op een visuele manier. Figuur 26 toont een voorbeeld van een storyboard.



Figuur 26: Voorbeeld van een storyboard

Als uitgangspunt voor het creëren van storyboards worden scenario's gebruikt. De inhoud van een scenario wordt met andere woorden visueel geïllustreerd. Een storyboard geeft een overzicht van de functionaliteit van en navigatie binnen het systeem. Door de visuele voorstelling kunnen gebruikers makkelijk deelnemen bij het creëren en aanpassen van storyboards. [48][76]

In [76] merkt men op dat computers meer en meer mobiel gebruikt worden, zoals bijvoorbeeld telefoons of PDA's. Hierdoor heeft de context waarin deze computers, en dus ook de aanwezige applicaties, gebruikt worden een grotere invloed op de manier waarop de gebruikers hun taken uitvoeren. Daarom is het nuttig het concept van Storyboarding uit te breiden door niet alleen user interface elementen af te beelden, maar ook illustraties van de situatie waarin het systeem gebruikt wordt. Figuur 27 toont een storyboard verrijkt met context-informatie.



Figuur 27: Storyboard met context-informatie

3.4.15 Evaluatietechnieken

Een greep uit de belangrijkste evaluatiemethoden wordt gegeven in tabel 5 [59].

Techniek	Uitvoerder	Beschrijving
Thinking Aloud	gebruiker	Terwijl de gebruiker interageert met het prototype, denkt hij of zij luid op. Hierdoor krijgen de observators een beter beeld van wat de gebruiker denkt en ondervindt tijdens het werken met de applicatie.
Meeting performance	gebruiker	Door het meten van de performance van de gebruiker, wordt er kwantitatieve data verkregen over de werkwijze van de gebruiker. Bij grafische software kan bijvoorbeeld de foutmarge gemeten worden bij het selecteren of manipuleren van kleine objecten. Bij de analyse van deze data, kunnen de designers beslissen om te kiezen voor bijvoorbeeld een nieuwe interactiemethode.

Techniek	Uitvoerder	Beschrijving
Retrospective testing	gebruiker	Retrospective testing bestaat uit twee delen, namelijk het testen van het prototype door de gebruiker, gevolgd door een review van de testsessie samen met de gebruiker. Tijdens het testen worden er videobeelden gemaakt van de gebruiker, deze worden overlopen tijdens de review en de onderzoeker kan bijkomende vragen stellen ter verduidelijking.
Cognitive Walkthrough	expert	Tijdens een Cognitive Walkthrough, gaan de evaluatoren samen met de ontwerpers de interface doorlopen met in het achterhoofd het doel van de gebruiker. De verschillende stappen die de gebruiker moet doorlopen om zijn doel te bereiken, samen met de bijbehorende acties, worden overlopen. Tijdens dit proces wordt er geverifieerd of de doorlopen stappen intuïtief en duidelijk zijn.
Heuristische evaluatie	expert	Ter uitvoering van een heuristische evaluatie, worden een reeks experts aangesteld die de interface gaan controleren en toetsen aan een reeks van “heuristieken”. Dit is een verzameling herkende usability principes. De gevonden problemen worden opgesomd en krijgen telkens een classificatie mee. Deze indeling duidt de ernst van het gevonden probleem aan.
Perspective Based	expert	Tljdens Perspective Based evaluatie werkt elke evaluator vanuit een verschillend standpunt en volgen zij een strikte evaluatieprocedure. Doordat de evaluators beschikken over verschillende verantwoordelijkheden en elk perspectief de focus legt op verschillende gebruikersstandpunten of -profielen, kunnen er meer problemen opgespoord worden. [77]

Tabel 5: overzicht van enkele evaluatiemethodes bij UCD

We merken op dat de kolom “uitvoerder” in tabel 5 de oorsprong van de vergaarde data aanduidt. Bijvoorbeeld bij het meten van performance, voert de gebruiker de eigenlijke test uit, maar de evaluators (eventueel samen met de ontwerpers) gaan de gegenereerde data analyseren en conclusies trekken.

3.5 Voordelen van het gebruik van UCD

Het toepassen van user-centered design bij het ontwikkelen van ondermeer softwareproducten heeft een aantal significante voordelen, zowel op economisch gebied als voor het succes van het product bij de klant en de eindgebruikers. In onderstaande secties worden deze voordelen besproken.

3.5.1 Economische voordelen

In [78] worden enkele voordelen van UCD aangehaald vanuit een economisch standpunt. Deze voordelen worden hieronder besproken.

- Reeds vroeg in de ontwikkelcyclus wordt de gebruiker actief betrokken bij het ontwikkelen van het systeem. Daarom is het mogelijk de tijd en kosten van het project te reduceren, omdat grote aanpassingen laat in het proces worden beperkt. Door regelmatige input en feedback van de gebruiker, worden problemen op tijd geïdentificeerd en is de kost om deze problemen op te lossen laag. Volgens [79] treden 63% van alle softwareprojecten buiten hun vooropgestelde kostenraming omwille van problemen met de bruikbaarheid van het systeem. Verder is volgens Nielsen de return on investment voor usability engineering 50 tot 100 keer groter dan de kost [79].
- UCD biedt de mogelijkheid producten te bouwen met een hoge mate van gebruiksvriendelijkheid en groot gebruiksgemak. Ontwikkelde producten kunnen daarom onder deze noemer op de markt gebracht worden, wat de attentie van klanten wekt. Het succesvolle gebruik van UCD zorgt ervoor dat het product zich differentieert van andere producten in de markt, met stijgende verkoopcijfers als gevolg. Gebruiksgemak is één van de belangrijkste redenen die de keuze voor een bepaald softwaresysteem beïnvloedt [79].
- Dankzij een efficiënte samenwerking met de eindgebruiker tijdens de ontwikkeling van het product, zal de gebruiker snel aan de slag kunnen met het eindproduct ter uitvoering van zijn taken. Zijn of haar productiviteit zal stijgen omdat het product intuïtief opgebouwd is. Mede hierdoor kan de omvang van handleidingen teruggebracht worden en eventuele opleidingen en trainingssessies worden gereduceerd. In het jaar 1991 bespaarde IBM $6,800,000 doordat werknemers per taak 9,6 minuten bespaarde. Deze reductie in uitvoeringstijd was te wijten aan usability research [79].
- Doordat de applicatie intuïtief werkt en uitvoerig getest is door zowel gebruikers als experts, zullen de gebruikers minder problemen ondervinden in hun werkomgeving. Hierdoor is het mogelijk kosten te besparen op de service en ondersteuning van het product. Tijdens een studie bij IBM gedurende 90 dagen, ondervond men dat 87% van alle contacten met de help desk gerelateerd waren aan problemen met usability [79].

3.5.2 Voordelen voor klant en eindgebruiker

Abras et al. [56] vermeldt de voordelen van een UCD benadering ten opzichte van de eindgebruikers en de klant:

- Door voldoende aandacht te besteden aan de noden en eisen van de gebruikers, zorgt de toepassing van UCD ervoor dat zij meer tevreden zijn met het eindproduct en het aangenamer vinden om mee te werken, wat stress en frustratie vermijdt.

- Het collaboratieve karakter van het UCD ontwikkelproces, resulteert in creatieve oplossingen wat de innovatie ten goede komt. Zo kunnen bijvoorbeeld alternatieven ontwikkeld en getest worden die zorgen voor een hogere productiviteit bij de eindgebruikers.
- Doorheen het UCD proces worden gebruikers geobserveerd in hun natuurlijke omgeving. De context van hun werk wordt zo in rekening gebracht, met als gevolg dat het product in ontwikkeling beter kan geïntegreerd worden in de bestaande werkomgeving.

3.6 Nadelen van het gebruik van UCD

Ondanks de vele voordelen, zijn er ook enkele nadelen verbonden aan de UCD werkwijze. Bij het gebruik van UCD in een organisatie kunnen verschillende problemen de kop op steken. Deze kunnen opgedeeld worden in vier domeinen [56], en worden besproken in onderstaande punten.

- **Participatie van de gebruikers**
  Eerst en vooral moeten de gebruikers geïdentificeerd worden. Wanneer de gebruikers niet bereikbaar zijn, wordt er best gewerkt met representatieve gebruikers. Indien er gewerkt moet worden zonder gebruikers, kan er gebruik gemaakt worden van scenario's en psychologische expertise over mensen. Deze manier van werken kan ook gebruikt worden wanneer de gebruikers onbekend zijn. Bij het kiezen van representatieve gebruikers is van belang dat het verschil tussen hen maximaal is en dat er verschillende categorieën van gebruikers gebruikt worden. Wanneer een lange tijd dezelfde gebruikers betrokken zijn, kunnen ze teveel bekend worden met de software en zijn ze niet representatief meer. Gebruikers kunnen geselecteerd worden op willekeurige basis. Dit is nuttig voor sommige projecten, echter moet er steeds gestreefd worden naar een grote variatie. De gebruikers zijn best gedurende het hele proces aanwezig om een goede bruikbaarheid van het eindproduct te garanderen. Door het toepassen van fieldworking kunnen designers en developers een beter idee krijgen van de omgeving waar het product gebruikt gaat worden. Het gebruik van videobeelden kan nuttig zijn voor het analyseren en visualiseren van scenario's en geven een duidelijk beeld hoe de gebruikers met het product omgaan. De UCD facilitator is de tussenpersoon tussen de users en de designers/developers. Hij of zij lost problemen op of dient enkel als observator. De facilitator moet verschillende rollen kunnen aannemen: als hij praat met een gebruiker moet hij dit doen vanuit het standpunt van designer of ontwikkelaar en visa versa. Gebruikers veranderen dikwijls van gedacht en weten niet goed wat ze willen. Een empirische aanpak is daarom aangeraden bij het betrekken van de gebruikers in het designproces. Parallel design en iteratieve ontwikkeling zijn onontbeerlijk bij UCD. Gebruikers werken met mock-ups met een stijgend detailniveau naarmate het project vordert. Zij evalueren deze mock-ups en helpen mee met het herdesign ervan.
- **Terminologie**
  Gebruikers begrijpen niet altijd de gebruikte termen bij systeemontwikkeling en design. Het is daarom aangeraden gebruik te maken van low-fidelity mock-ups voor de interactie met de gebruikers.
- **Kosten van UCD**
  Door de uitgebreide analyse van en de samenwerking met gebruikers, neemt het UCD proces veel tijd in beslag. Verder is UCD toevoegen aan een bestaand proces dikwijls zeer duur. Er wordt best vanaf het begin gewerkt met UCD.

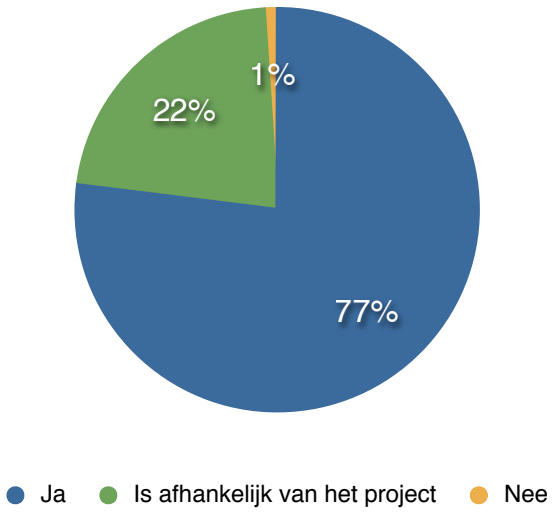
- **Organisatie**
  Er moet rekening gehouden worden met eventuele conflicten die kunnen voorkomen tijdens het proces. Gebruikers staan zeer laag in de hiërarchie van de organisatie, met als gevolg dat hun inspraak soms te klein is. Managers willen graag hun zegje doorduwen.
- **Communicatie**
  Het is niet zo zeer belangrijk dat de gebruikers de designers begrijpen, andersom wel. De designer moet duidelijk maken aan de gebruiker welke technologieën er beschikbaar zijn en waarvoor ze kunnen gebruikt worden, ze moeten de gebruiker zijn noden, ideeën en verwachtingen helpen uitdrukken.

3.7 Gebruik van UCD in de praktijk

Deze sectie bespreekt het gebruik van User-Centered Design in de praktijk. In [80] werd een onderzoek gevoerd naar de mate waarin er rekening gehouden wordt met “usability” bij het ontwikkelen van software. De populatie bij dit onderzoek werd gedefinieerd als alle softwarebedrijven uit Noorwegen.

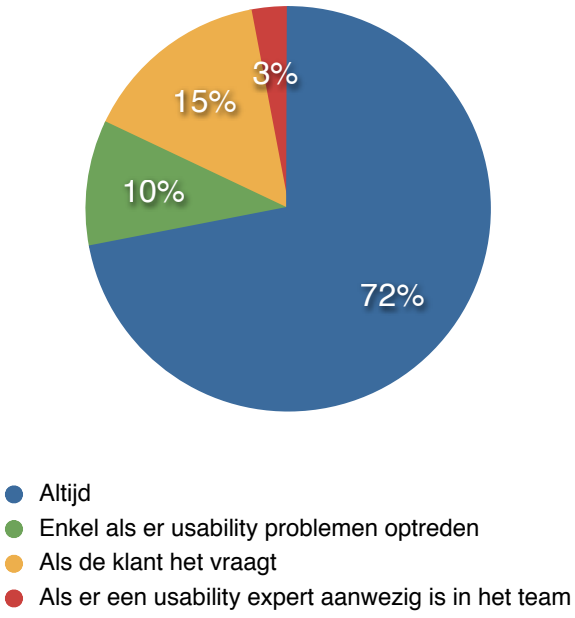
Onder andere volgende vragen werden gesteld aan de deelnemers van de survey:

1. *Is usability belangrijk tijdens uw projecten?*



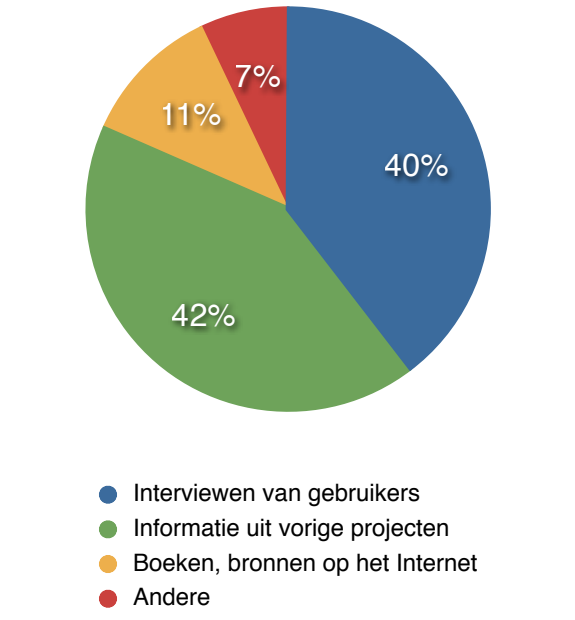
We zien dat het belang van usability erkend wordt door het merendeel van de bedrijven. 77% van de bedrijven vindt de bruikbaarheid van het systeem belangrijk. Slechts 1% van de ondervraagde bedrijven hecht geen belang aan bruikbaarheid.

2. *Houdt u rekening met usability tijdens het opstellen van de requirements voor het systeem?*



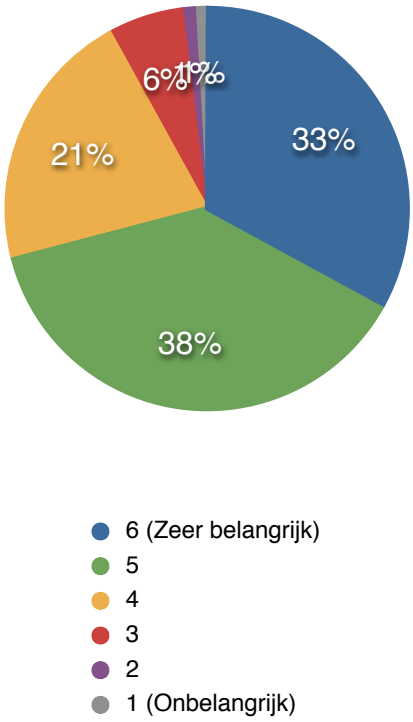
We zien dat het belang van usability erkend wordt door het merendeel van de bedrijven. 72% houdt steeds rekening met de bruikbaarheid van het systeem en dit reeds tijdens de initiële fase in de UCD cyclus, namelijk bij het specificeren van de requirements.

3. *Hoe verzamelt u usability requirements?*



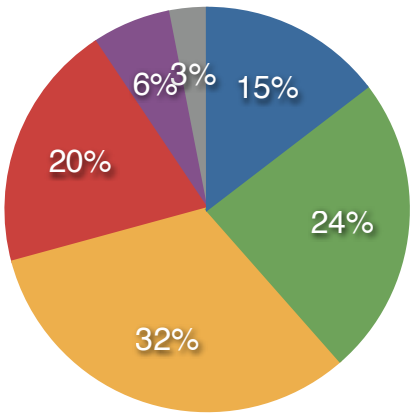
Het specificeren van usability requirements gebeurt voornamelijk door het interviewen van gebruikers en door het raadplegen van resultaten uit vorige projecten.

4. *Hoe belangrijk zijn usability requirements voor het succes van uw projecten?*



De resultaten van de survey wijzen erop dat er zeer veel waarde gehecht wordt aan usability requirements voor het verhogen van de slaagkans van een project. 71% van de bedrijven gaf een rating van 5 of 6 op deze vraag.

5. *Hoe belangrijk is usability testing voor het succes van uw projecten?*



Wanneer we de antwoorden op deze vraag vergelijken met de antwoorden op vraag 4, merken we op dat er minder belang gehecht wordt aan usability testing dan aan usability requirements.

Ook in [46] werd er een onderzoek uitgevoerd met betrekking tot UCD in de praktijk, de focus ligt hier echter op de organisatie van een UCD project en de gebruikte methoden en technieken. In totaal vulden 103 personen de questionnaire in. Onderstaande tabel vat de resultaten van dit onderzoek samen.

Methode of techniek	Impact	Gebruik
Iterative design	2,15	65
Usability evaluation	2,39	43
Task analysis	2,61	34
Informal expert review	3,28	31
Field studies	2,00	28
Focus groups	2,79	16
Formal usability evaluation	2,86	15
Prototype without user testing	3,07	15
User interviews	3,00	11
Surveys	3,17	9
User requirements analysis	2,00	7
Participatory design	3,40	7
Card sorting	3,33	5

Tabel 6: Gebruik van UCD methoden en technieken in de praktijk [46]

De impact van een bepaalde methode of techniek op onder andere de bruikbaarheid van het systeem en kostenbesparing tijdens de ontwikkeling, wordt uitgedrukt op een schaal gaande van 1 (meeste impact) tot 5 (minste impact). De kolom “gebruik” duidt aan hoeveel van de ondervraagde personen de desbetreffende methode of techniek gebruikt bij het ontwikkelen van een systeem. De tabel is aflopend gesorteerd op gebruik. In theorie zou het logisch zijn dat de methoden met de hoogste impact het meest gebruikt zouden worden. Echter in tabel 6 zien we dat dit niet altijd zo is. Informal expert review wordt bijvoorbeeld vaak toegepast in de praktijk, maar heeft niet zozeer een grote impact, terwijl field studies en requirements analysis weinig gebruikt worden ondanks hun hoge impact. De verklaring hiervoor is dat in de praktijk tijd en kosten een belangrijke rol spelen bij het produceren van software. Deadlines moet men zoveel mogelijk respecteren en het budget mag niet overschreden worden. Daarom zal men dikwijls de kosten en voordelen van de beschikbare methodes en technieken afwegen om tot de beste oplossing te komen.

Het onderzoek toonde tevens aan dat het nut en de bruikbaarheid van de ontwikkelde producten binnen een bedrijf steeg wanneer er UCD gebruikt werd. Respectievelijk 79% en 82% van de ondervraagden bevestigde dit.

3.8 Conclusie

In dit hoofdstuk werd User-Centered Design besproken. Door de inzet van UCD tijdens het ontwikkelen van software, doelt men op gebruiksvriendelijke en bruikbare applicaties. Het uitvoeren van een uitgebreide analyse van de context en de requirements, het werken met prototypes bij het ontwerpen van de applicatie en een grondige evaluatie door de gebruikers in een levensecht situatie, zorgt ervoor dat er een goede gebruikerservaring kan gecreëerd worden. De UCD cyclus is iteratief opgebouwd, waardoor de applicatie steeds verfijnt en verbeterd wordt.

In sectie 3.5 en 3.6 werden de voor- en nadelen van het gebruik van UCD aangetoond. Belangrijkste voordelen zijn de daling van de ontwikkelkosten door evaluatie vroeg in de UCD cyclus en een betere bruikbaarheid van het systeem. De complexe communicatie binnen het multidisciplinaire team, het feit dat het UCD proces veel tijd in beslag neemt en de gecompliceerde samenwerking met de gebruikers zijn de voornaamste nadelen.

Hoofdstuk 4

Een geïntegreerd ontwikkelingsproces

4.1 Inleiding en doelstellingen

Tijdens het onderzoek uit voorgaande hoofdstukken hebben we stilgestaan bij enkele belangrijke aspecten van een softwareontwikkelingsproject. We kunnen hierbij aansluitend dan ook een reeks voorwaarden opgeven waaraan een softwareproduct moet voldoen om te kunnen spreken van een geslaagd project. We kunnen stellen dat het product van een geslaagd softwareproject een stabiel, bruikbaar, functioneel en nuttig systeem is dat ondersteuning biedt aan zijn gebruikers bij het uitvoeren van hun taken, en dit op een gebruiksvriendelijke manier. Dit product moet tevens binnen het vastgelegde budget en binnen de afgesproken tijdspanne voltooid worden. Ook de ontwikkelaars van de software zelf spelen een belangrijke rol: om een goede voortgang en een hoge efficiëntie van het proces mogelijk te maken moeten zij op een comfortabele manier kunnen werken en de collaboratie en communicatie met hun collega's moet vlot kunnen verlopen.

In dit hoofdstuk zullen we, rekening houdend met de vermelde doelstellingen, pogen een proces te definiëren dat de slaagkansen van een softwareproject probeert te verhogen, zonder de bruikbaarheid en gebruikersvriendelijkheid van het eindproduct uit het oog te verliezen. Hiervoor zullen we ons baseren op de methodologieën besproken in hoofdstuk 2 en 3, namelijk Agile Software Ontwikkeling en User-Centered Design respectievelijk. Deze methodologieën zorgen namelijk elk apart voor de invulling van een deel van de doelstellingen. Om echter aan al de genoemde doelstellingen te voldoen, is het nuttig om een nieuw proces te definiëren dat deze twee methodologieën integreert.

In de komende secties zal blijken dat de stakeholders die betrokken zijn bij het proces (ontwikkelaars, klant, vormgevers, ...), een uiteenlopende achtergrond en kennis hebben. Onder andere om de samenwerking tussen deze personen te ondersteunen, zal een tool voorgesteld worden ter ondersteuning van het nieuwe proces. Deze tool zal een belangrijk ingrediënt blijken te zijn bij de integratie.

We starten in sectie 4.2 met de vraag waarom we kiezen voor een integratie op basis van Agile Software Ontwikkeling en UCD. In sectie 4.3 nemen we enkele oplossingen uit de literatuur onder de loep. Daarna volgt een uitdieping van de fundamenten van de voorgestelde integratie in sectie 4.4. Vertrekkende van deze basis zullen we zelf een proces uitwerken. De werkwijze hiervoor is terug te vinden in sectie 4.5. In sectie 4.6 bespreken we de voordelen van de uitgevoerde integratie.

4.2 Waarom een integratie?

Alvorens verder te gaan, is het nuttig de vraag te stellen of de twee aangehaalde methodologieën niet reeds voldoen aan de zonet gestelde eisen voor succesvolle en kwalitatieve software. Zoals we reeds ondervonden in hoofdstuk 2, zorgt Agile Software Ontwikkeling voor een technisch goed uitgewerkt systeem (o.a. door het uitvoeren van strenge testen) dat beantwoordt aan de requirements vanwege de klant. De kans op een stabiel en nuttig systeem is dus groot. Korte iteraties en een goede samenwerking met de klant reduceren het risico op falen omdat de kans groter wordt dat de ontwikkeling van het systeem binnen het budget en binnen de afgesproken deadlines voltooid wordt. Zoals er echter in sectie 2.7 werd aangehaald, is een hoge mate van bruikbaarheid en gebruiksvriendelijkheid van het afgeleverde systeem niet altijd vanzelfsprekend. User-Centered Design definieert een proces en een reeks methoden en technieken die de nood naar deze twee factoren probeert in te vullen. UCD vereist echter veel tijd voor het uitvoeren van onder andere context-onderzoeken en gebruikerstesten. Het gebruik van UCD mist daarom veel van de voordelen van Agile Software Ontwikkeling (zoals de snelle resultaten en het efficiënt hanteren van veranderingen). Daarom zullen we de focus leggen op een integratie van deze twee methoden.

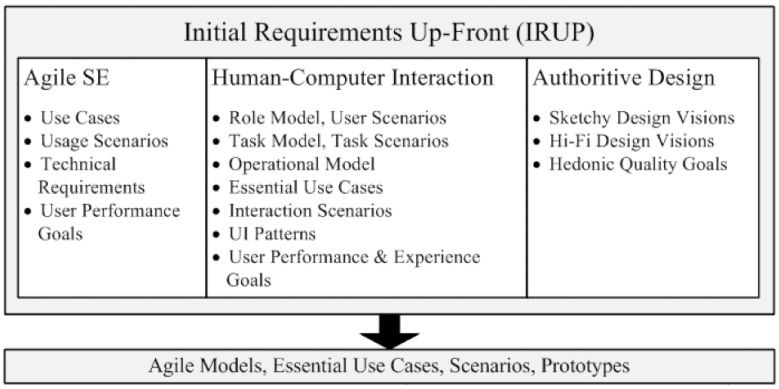
4.3 Oplossingen uit de literatuur

In de literatuur worden reeds enkele oplossingen voorgesteld. In onderstaande secties bespreken we kort enkele van de belangrijkste. In sectie 4.5 bouwen we vervolgens een eigen proces uit. Enkele ingrediënten uit deze sectie zullen hierbij aangewend worden.

4.3.1 CRUISER: een brug tussen SE en HCI

Het CRUISER (Cross-Discipline User Interface and Software Engineering Lifecycle) proces is een ontwikkelmethode beschreven door Memmel et al. [90], gebaseerd op XP en Agile Modeling (AM) [91] en poogt de kloof tussen SE en HCI (meerbepaald Agile Software Ontwikkeling en User Interface Design) te overbruggen. Dit proces steunt op het gebruik van modellen en patterns als primaire artefacten. Tijdens de loop van een project worden er artefacten gebruikt die eigen zijn aan het domein dat aansluit bij de behandelde activiteit (bij de definitie van de technische requirements worden er bijvoorbeeld use cases gebruikt, terwijl voor de beschrijving van een bepaald scenario storyboards aangewend worden). Voor de communicatie tussen de teamleden en met de stakeholders worden deze artefacten vertaald naar scenario's en prototypes. Deze werkwijze behoudt de voordelen van domein-eigen artefacten, terwijl de vertaling zorgt voor een efficiënte communicatie met de stakeholders.

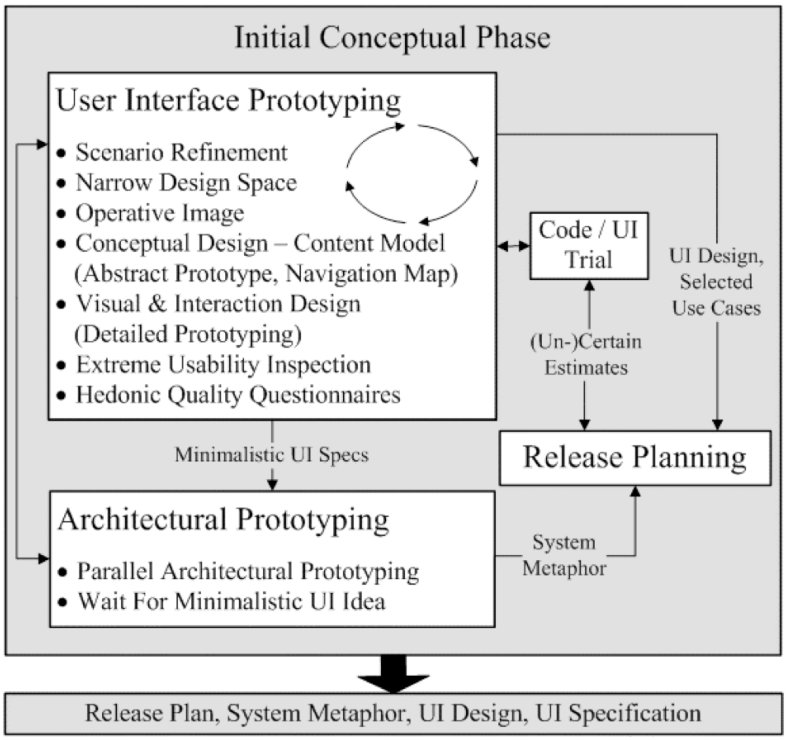
Figuur 28 toont het verloop van de initiële fase waarin de requirements verzameld worden in samenwerking met de stakeholders. Hierbij worden light-weight artefacten uit SE en HCI gebruikt.



Figuur 28: initiële fase bij een CRUISER proces (verzamelen van de requirements) [90]

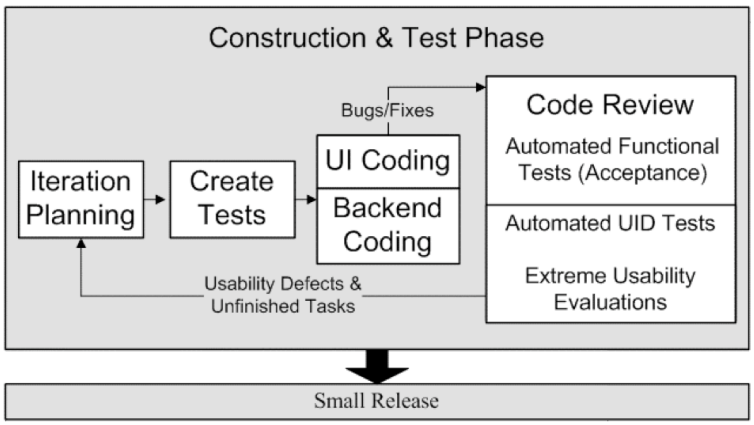
Het resultaat van deze fase zijn een reeks modellen, use cases, scenario's en prototypes die verder verfijnd worden in volgende fases.

De tweede fase in de CRUISER levenscyclus is de "Initial Conceptual Phase" (ICP, zie figuur 29). Om de efficiëntie te verhogen worden vanaf deze fase de user interface en de systeemarchitectuur zoveel mogelijk in parallel uitgewerkt aan de hand van prototyping. Vooraleer dit mogelijk is worden de nodige interfaces tussen de UI en de systeemarchitectuur geïdentificeerd aan de hand van de artefacten geproduceerd tijdens de initiële fase. Deze interfaces bieden een overzicht aan de ontwikkelaars van de afhankelijkheden tussen de systeemarchitectuur en de UI. Door deze interfaces correct te definiëren wordt het eenvoudiger om parallel te werken. Verder worden tijdens deze fases de prototypes uit de initiële fase verder uitgewerkt om zo te komen tot een oplossing die gesteund wordt door alle stakeholders. Hierbij wordt er rekening gehouden met het nodige Agile karakter van de prototypes: aanpassingen moeten snel kunnen plaatsvinden, interactiviteit moet toegelaten zijn en de prototypes moeten begrijpbaar (en eventueel manipuleerbaar) zijn voor alle stakeholders. Het produceren van prototypes gebeurt dan ook best met de juiste tools.



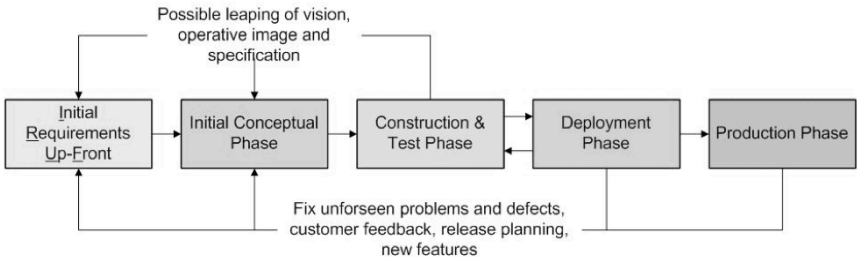
Figuur 29: CRUISER: Initial Conceptual Phase

Het coderen van de software gaat van start tijdens de derde fase in het proces, namelijk de Construction & Test Phase (CTP, zie figuur 30). Hierbij wordt de iteratieve en incrementele werkwijze van XP als basis overgenomen. Ook Pair Programming wordt toegepast in dit kader. De pairs zijn samengesteld uit twee ontwikkelaars of uit één ontwikkelaar samen met één UI Designer.



Figuur 30: CRUISER: Construction & Test Phase

De laatste twee fases in de CRUISER levenscyclus zijn de “Deployment Phase” en de “Production Phase”. Tijdens deze fases wordt feedback opgevangen op het afgeleverde product. Deze feedback kan gaan van de aanvraag tot nieuwe functionaliteit of het aanpassen van huidige functionaliteit. Het volledige CRUISER proces staat afgebeeld in figuur 31.



Figuur 31: Overzicht van de CRUISER levenscyclus

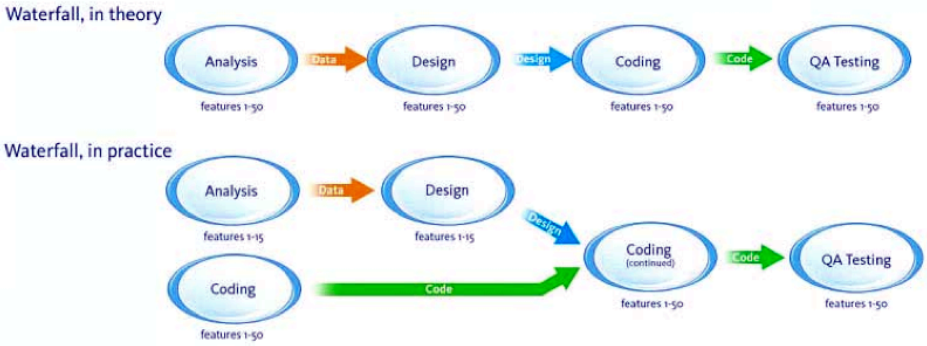
4.3.2 Parallelle tracks (Autodesk)

In [22] beschrijft men de werkwijze die aangewend werd binnen het bedrijf Autodesk[8] (vroeger Alias) voor de ontwikkeling van een reeks softwareproducten (o.a. SketchBook[9]). Voordien werd hier steeds het Waterfall proces aangewend voor ontwikkelingen, maar vanwege de grotere flexibiliteit en zekerheid van Agile processen (zie hoofdstuk 2), werd er voor een reeks projecten beslist te werken volgens een Agile methodologie. Om de bruikbaarheid en gebruiksvriendelijkheid van het eindproduct echter te garanderen en om de eindgebruiker niet uit het oog te verliezen, werd er een proces bedacht dat elementen van User-Centered Design combineert met een Agile werkwijze.

Het gebruik van Waterfall binnen Autodesk zorgde voor enkele problemen. Zoals we kunnen zien in figuur 32, bestaat het Waterfall proces uit een reeks sterk afgebakende fases (zie ook hoofdstuk 1). UCD past goed in dit proces omdat de usability activiteiten (etnografie, interviews, focus groepen, prototyping, ...) in een ideaal UCD proces steeds plaats vinden voor de codering van de software. In de praktijk echter begin men reeds met coderen vanaf de start van het project, wat ervoor zorgde dat voor een groot aantal features een slecht of zelfs geen design geproduceerd werd (zie figuur 32). Dikwijls was er ook een afwijking tussen de implementatie en het oorspronkelijk design. Het resultaat was een product met niet bruikbare en zelfs ontbrekende functionaliteit. Uiteindelijk koos men voor een Agile werkwijze vanwege de mogelijkheid om de ontwikkeling van het product op te delen in kleinere delen en deze iteratief te ontwikkelen. Deze opsplitsing gebeurt (zoals besproken in hoofdstuk 2) volgens een “just-in-time” werkwijze. Hierdoor verkleint de granulariteit van de UCD activiteiten en de ontwikkelingen omdat men elke feature apart kan behandelen.

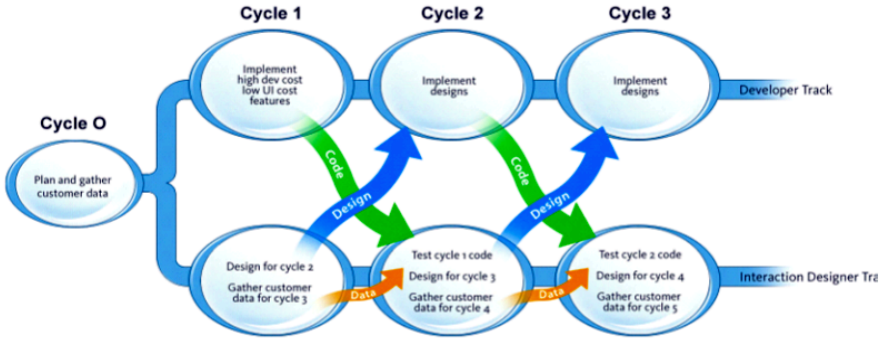
[8] http://www.autodesk.com

[9] http://usa.autodesk.com/adsk/servlet/index?id=6848332&siteID=123112



Figuur 32: Contrast tussen Waterfall in theorie, en het verloop van een Waterfall proces binnen Autodesk [22]

Een belangrijk onderdeel van UCD is de uitvoering van iteraties over een design van de software. Deze iteraties vinden best plaats voordat het design gecodeerd wordt omdat men makkelijker, sneller en goedkoper aanpassingen kan doen aan prototypes dan aan gecodeerde software. Ook kan het ontwerp vooraf met de eindgebruikers geëvalueerd worden ter controle van de bruikbaarheid en gebruiksvriendelijkheid. Zoals bleek uit ons onderzoek in hoofdstuk 2, gaat men binnen een Agile proces zo snel mogelijk (meestal vanaf dag één, afhankelijk van het project) van start met het coderen van de software. Daarom werd er beslist om het design en de implementatie te laten plaatsvinden in twee aparte, parallelle sporen (zie figuur 33).



Figuur 33: Opsplitsing van de activiteiten binnen opeenvolgende iteraties in twee parallelle tracks [22]

Deze werkwijze impliceert dat de designers een stap voorblijven op de ontwikkelaars: in iteratie x wordt een ontwerp gemaakt voor de geplande feature. Dit design wordt iteratief ontwikkeld (prototyping) en dus ook getest binnen dezelfde iteratie. Het resultaat van deze activiteiten wordt doorgegeven aan de ontwikkelaars ter implementatie in de volgende iteratie (x+1). Belangrijk bij deze werkwijze is de opsplitsing van het totale design van de software in kleinere stukjes die apart behandeld kunnen worden binnen de korte tijdspanne van een Agile iteratie (typisch 4 tot 6 weken).

UCD is traditioneel een zwaar gedocumenteerd proces: o.a. requirements worden genoteerd, evenals het gedrag van de UI en resultaten en bevindingen van usability tests (zie hoofdstuk 3). Het schrijven enerzijds en het raadplegen anderzijds van deze documentatie neemt veel tijd in beslag. Daarom werd er gekozen om hoofdzakelijk te werken met de reeds vertrouwde index cards uit Agile Software Ontwikkeling (zie ook hoofdstuk 2). Dit artefact zorgt voor een makkelijke communicatie binnen het project team. Wijzigingen aan een design worden nog steeds in tekst beschreven. Deze tekst wordt echter enkel gebruikt binnen de workflow van de designers en niet ter communicatie naar de rest van het team.

Het proces dat we zullen definiëren zal tevens gebruik maken van parallelle tracks, zoals beschreven in [22]. Een gedetailleerde beschrijving van de werkwijze van deze methode wordt dan ook gegeven in sectie 4.5.

4.4 Fundamenten

In zijn discussie [81] met Kent Beck, haalde Alan Cooper aan dat er twee kanten zijn aan software, namelijk de communicatie met de computerhardware en de communicatie met de gebruiker (zie figuur 34). Deze twee entiteiten zijn verschillend, in die zin dat hardware een voorspelbaar deterministisch gedrag kent terwijl een menselijke gebruiker onderhevig is aan bijvoorbeeld emotie en stress.

Figuur 34: Communicatie van de software met de hardware enerzijds en de gebruiker anderzijds

Het ontwerpen en ontwikkelen van deze twee kanten van een softwaresysteem moet dan ook op verschillende wijze gebeuren. De tak van Software Engineering (SE) houdt zich voornamelijk bezig met de architectuur van het onderliggende systeem, terwijl Human-Computer Interaction (HCI) de communicatie tussen het systeem en zijn gebruikers uitwerkt [26][32].

SE en HCI leggen elk de focus op een ander aspect van een softwareproduct (zoals afgebeeld in figuur 34). Afzonderlijk dragen ze elk hun deel bij aan het ontwikkelingsproces, maar om uiteindelijk een gebruikersvriendelijk en functioneel geheel mogelijk te maken, is er inbreng nodig van beide domeinen [90]. Bijgevolg is een vlotte samenwerking en communicatie tussen deze twee domeinen uiterst noodzakelijk. In dit hoofdstuk is het dan ook de bedoeling een proces uit te werken dat elementen uit beide domeinen op een

efficiënte manier combineert. Efficiënt in die zin dat er een raamwerk geboden wordt waarin personen uit beide domeinen kunnen functioneren en samenwerken, zonder belangrijke methoden en technieken uit één van beide domeinen uit het oog te verliezen.

Bij een softwareontwikkelingsproces zijn er bepaalde stappen die steeds terugkeren, zoals o.a. het verzamelen van requirements, het ontwerp en de ontwikkeling. Voor het beheren van de resultaten van elk van deze stappen worden zogenaamde “artefacten” gebruikt. Deze documenten worden voornamelijk als volgt aangewend [90]:

- Invoer bieden aan volgende fases in de ontwikkeling
- Weergeven van de voortgang van het project
- Communicatie tussen de verschillende stakeholders ondersteunen

De manier waarop deze artefacten gebruikt worden en de aard ervan, is een belangrijke factor voor het al dan niet slagen van het project. Daarom is het belangrijk dat telkens de juiste artefacten gekozen worden, afhankelijk van het type project.

Gezien de eisen van het merendeel van de hedendaagse softwareprojecten, zorgen formele artefacten en zware documentatie voor inflexibiliteit en moeilijke communicatie tussen de verschillende stakeholders [90]. Bij de samenwerking tussen SE en HCI wordt dit gegeven enkel bevestigd: het aantal disciplines binnen het projectteam verhoogd met als gevolg dat het belang van de communicatie en de aangewende documenten vergroot. Daarom is het efficiënter te werken met Agile methoden en artefacten, zowel uit SE als uit HCI.

Vanwege de contrasterende domeinen van SE enerzijds en HCI anderzijds, worden bij deze twee disciplines veelal verschillende artefacten gebruikt. Bij SE is het noodzakelijk om bijvoorbeeld de gevraagde functionaliteit te vertalen in concrete UML diagrammen en Use Cases, terwijl er bij HCI gebruikt gemaakt wordt van minder formele documenten zoals scenario’s, Storyboards en styleguides. Dit verschil in graad van formaliteit is fundamenteel aan deze domeinen.

Zoals bleek uit ons onderzoek in hoofdstuk 2, faciliteert een Agile werkwijze de communicatie tussen de stakeholders. Agile SE (zoals besproken in hoofdstuk 2) is daarom een goed uitgangspunt voor het vertegenwoordigen van de methoden en technieken uit de Software Engineering bij het beoogde proces.

User-Centered Design daarentegen (zie hoofdstuk 3), kent per definitie geen doorgedreven Agile werkwijze: bij UCD kan er gebruik gemaakt worden van tijdsintensieve methoden en formele artefacten zoals o.a. taakanalyse (zie sectie 3.4.6), use cases (zie sectie 3.4.9) en etnografisch onderzoek (zie sectie 3.4.4). Uit het onderzoek in hoofdstuk 3 bleek echter ook dat eveneens minder-formele artefacten gebruikt kunnen worden bij UCD (zoals scenario’s en personas - zie sectie 3.4.8), evenals light-weight methoden zoals bijvoorbeeld Storyboarding (sectie 3.4.14). De juiste keuze van Agile technieken en artefacten uit UCD kan dus resulteren in een Agile werkwijze die kan aangewend worden bij de integratie van SE en HCI.

4.4.1 Enkele aandachtpunten

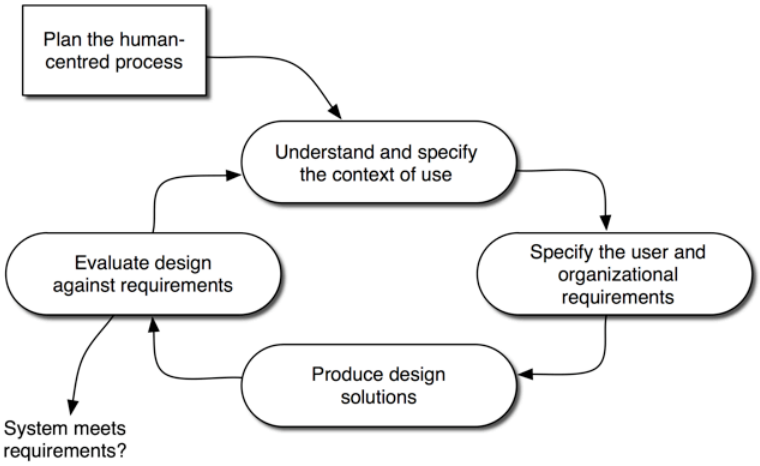
In sectie 2.4 belichtten we de levenscyclus van een typisch Agile proces. We merkten op dat tijdens de inleidende fase (cycle zero) een horizontale analyse en eventueel een horizontaal design wordt uitgevoerd. Deze fase wordt beëindigd vanaf het moment dat er voldoende hoogniveau requirements zijn verzameld en de scope van het project in grote lijnen duidelijk is. Deze werkwijze volgt uit de veronderstelling dat verdere details verticaal gespecificeerd kunnen worden naarmate het project vordert. Wanneer de software in ontwikkeling sterk gefocust is op zijn gebruikersinterface (UI), zoals vandaag de dag steeds meer het geval is, ontstaan er enkele problemen wanneer er gewerkt wordt met Agile Software Ontwikkeling. Door de iteratieve en incrementele ontwikkelingen (zie sectie 2.2) is het mogelijk dat de gebruikersinterface inconsistent wordt of dat de samenhang ervan negatief wordt beïnvloed. Ook is het niet eenvoudig om de software te testen met eindgebruikers wanneer de gebruikersinterface en -interactie onderhevig is aan constante veranderingen. Om die reden is Agile Software Ontwikkeling beter geschikt als methode voor het ontwikkelen van software die meer afhankelijk is van de achterliggende architectuur en niet zozeer van gebruikersinterface. De achterliggende principes kunnen echter wel dienen als basis voor een integratie. [90] UCD probeert een oplossing te bieden voor dit probleem.

4.4.2 Vergelijking van de levenscycli van Agile Software Ontwikkeling en UCD

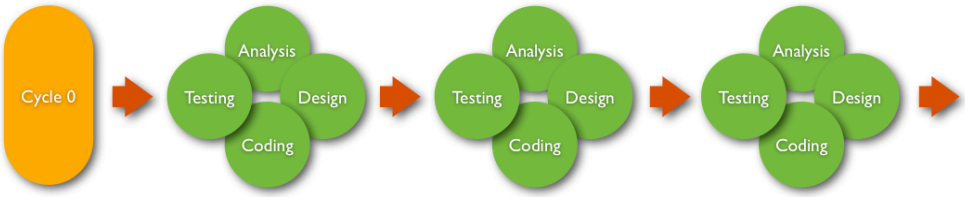
Het basisprincipe achter Agile Software Ontwikkeling is, zoals we zagen in hoofdstuk 2, een iteratieve en incrementele werkwijze. Deze manier van werken komt tevens terug bij UCD. Zoals we zagen in hoofdstuk 2 verstrekt de iteratieve aard van Agile methoden een goede basis voor het verzamelen van feedback tijdens het ontwikkelproces, terwijl UCD op een iteratieve wijze het ontwerp van de applicatie uitwerkt. Beide methoden delen dus een door feedback gedreven proces. Verder gebeurt de communicatie tussen de verschillende partijen met behulp van visuele schema's: bij Agile Software Ontwikkeling wordt de communicatie ondersteund door gecodeerde prototypes van het systeem, terwijl bij UCD er vooral gewerkt wordt met mock-ups van de user interface. [26][32]

Om de verschillen en overeenkomsten verder te illustreren, zullen we nu de levenscycli van beide methoden met elkaar vergelijken. In figuur 35 en figuur 36 staan deze cycli afgebeeld. Op het eerste zicht zijn deze vrij compatibel: de verschillende fases kunnen op hoog niveau op elkaar afgebeeld worden als volgt:

- De eerste 2 fases in de UCD cyclus komen overeen met de analysefase uit de Agile cyclus
- De derde fase in de UCD cyclus komt overeen met de design- en implementatiefase uit de Agile cyclus
- De laatste fase in de UCD cyclus komt overeen met testfase uit de Agile cyclus



Figuur 35: UCD levenscyclus volgens ISO 13407 [33]



Figuur 36: Agile levenscyclus [22]

De lengte van de verschillende cycli en de duur van de activiteiten binnen een bepaalde cyclus, kennen binnen een UCD proces echter een sterke fluctuatie. Verder gebruikt UCD andere technieken en methoden dan een Agile proces. Het uitvoeren van onder andere etnografie, interviews, Prototyping en usability testing neemt typisch veel tijd in beslag, omwille van de complexe samenwerking met de eindgebruikers en de mate waarin fieldwork wordt verricht. Doordat deze activiteiten veel tijd vergen, wordt UCD door aanhangers van Agile methoden vaak gezien als “Big Up Front Analysis and Design” [26][50]. Deze vooroordelen zijn echter niet altijd gegrond. De reden waarom doorgedreven initiële analyse en design verworpen wordt binnen de Agile gemeenschap, is de aanname dat - naarmate het project vordert - er veranderingen optreden op gebied van onder andere requirements en technologie (zie sectie 2.3). De focus tijdens het UCD context-onderzoek ligt echter op het begrijpen en interpreteren van de algemene werkomgeving waarin het nieuwe systeem zal fungeren en de eigenschappen van de gebruikers en de manier waarop zij hun taken uitvoeren. Deze context is namelijk fundamenteel aan het gebruik van het systeem en kent weinig veranderingen doorheen de tijd [50]. Naarmate het UCD project vordert, wordt er meer en meer toegespitst op het iteratief opstellen, uitwerken en evalueren van prototypes. Zoals reeds eerder werd vermeld, omvat UCD ook een reeks Agile technieken en artefacten

die meer flexibiliteit bieden aan het ontwikkelingsproces. Mede hierdoor wordt het mogelijk om tot een integratie met Agile Software Ontwikkeling te komen.

Een belangrijk aandachtspunt bij de integratie, is de interactie met de klant. Bij beide methoden - zowel Agile Software Ontwikkeling als UCD - is er continu contact tussen het projectteam en de klant. De klant behoort echter niet altijd tot de eindgebruikers van de software. Bij Agile Software Ontwikkeling wordt er gestreefd naar het invullen van de noden en eisen van diegene die de software aankoopt (de werkelijke klant). Deze heeft echter niet steeds hetzelfde profiel als de eindgebruiker, wat ertoe kan leiden dat de software voor de eindgebruiker moeilijk of niet bruikbaar is. Daarom wordt bij UCD de klant gedefinieerd als zijnde de eindgebruiker [26][81] (zie ook sectie 3.2). Een gevolg hiervan is dat de te ontwikkelen software beter zal passen binnen de workflow van de eindgebruikers, wat resulteert in een hogere tevredenheid en productiviteit.

Wanneer we tenslotte de voortgang van een Agile en UCD proces vergelijken, merken we op dat een Agile proces sterker gecoördineerd is dan een UCD proces. Agile Software Ontwikkeling legt strikte deadlines op voor elke iteratie (Timeboxing, zie sectie 2.2.5). Bovendien moet het product van een iteratie telkens een volledig afgewerkt en getest geheel vormen dat kan samengevoegd worden met de rest van het systeem. UCD daarentegen legt - zeker in het begin van het project - de focus op het ontwerpen van de user interface in zijn totaliteit. Vervolgens worden bepaalde functies verder apart uitgewerkt en bijgevoegd bij het geheel. Deze werkwijze wordt gebruikt om de consistentie van het systeem te maximaliseren.

Om een integratie van UCD en Agile Software Ontwikkeling mogelijk te maken, moeten we dus onderzoeken hoe we de UCD levenscyclus kunnen afstemmen op een Agile levenscyclus. Ook is het nodig UCD methoden en technieken te selecteren die nuttig zijn binnen een snel en wendbaar proces zoals Agile Software Ontwikkeling. Deze selectie hangt sterk af van het project in kwestie en is afhankelijk van de mate waarin het ontwikkelteam reeds bekend is met het applicatiedomein, de beschikbare data uit gelijkaardige projecten in het verleden, de doelgroep van het systeem, enzovoort. We merken op dat het van primair belang is dat we de principes van beide methoden zo veel mogelijk in stand houden.

4.5 Uitwerking eigen proces

In deze sectie trachten we zelf een ontwikkelingsproces te definiëren met als uitgangspunt de literatuur die we in voorgaande secties en hoofdstukken behandelden. Voor dit proces stellen we drie primaire doelstellingen voor:

- **Het proces moet een Agile karakter hebben**
  Een Agile proces is uiterst geschikt wanneer er binnen het ontwikkelteam een grote diversiteit bestaat van verschillende disciplines. Door het verschil in terminologie en kennis is goede samenwerking en communicatie noodzakelijk. Een Agile proces stimuleert dit (zie ook hoofdstuk 2 en sectie 4.2). Andere voordelen van een Agile werkwijze zijn het verlaagde risico op een mislukt project en de snelle aflevering van werkende software.
- **De geproduceerde software moet voldoen aan de eisen van de klant**
  Deze doelstelling spreekt voor zich: wanneer de ontwikkelde software niet voldoet aan de

eisen van de klant, kunnen we niet spreken van een geslaagd project. Een Agile werkwijze ondersteund overigens deze doelstelling.

- **De geproduceerde software moet gebruiksvriendelijk zijn voor de eindgebruikers**
  In sectie 4.2.1 brachten we een belangrijk nadeel van een Agile werkwijze onder de aandacht, namelijk het feit dat de gebruiksvriendelijkheid van de ontwikkelde software in gevaar kan komen. Daarom krijgt dit aspect bijzondere aandacht binnen ons proces door het aanwenden van methoden en technieken uit UCD (zie hoofdstuk 3).

Deze doelstellingen betekenen in essentie een integratie van SE en HCI. In sectie 4.4 bespraken we reeds de bruikbaarheid van Agile Software Ontwikkeling enerzijds en Agile UCD anderzijds als basis voor deze integratie. We gaan dan nu ook verder met een uiteenzetting over hoe deze integratie kan plaatsvinden om zo te komen tot een proces dat voldoet aan bovenstaande doelstellingen.

4.5.1 Samenstelling van het projectteam

Wanneer we een vergelijking maken tussen de vaardigheden en kennis aanwezig binnen Agile teams enerzijds en UCD teams anderzijds, merken we een contrast op. Agile teams werken hoofdzakelijk vanuit het standpunt van Software Engineering, terwijl UCD teams een multidisciplinaire aanpak hanteren eigen aan het domein van HCI (zie sectie 2.3, sectie 3.2 en figuur 34).

Om te komen tot een succesvol softwareproject, is - zoals besproken in hoofdstuk 2 en 3 en sectie 4.4 - de aanwezigheid van een aantal stakeholders van cruciaal belang. Onderstaande tabel geeft een overzicht van de belangrijkste stakeholders bij een typisch softwareproject, samen met bijbehorende disciplines en vaardigheden.

Usability experts & designers	Gebruikers- en taakanalyse, ontwerp UI, usability engineering, ergonomie, ...
Developers	Vertalen van een ontwerp naar programmacode
Klant	Stellen van requirements en doelstellingen

Tabel 7: overzicht van belangrijkste stakeholders bij een softwareproject, samen met bijbehorende disciplines

Deze verscheidenheid aan disciplines kan in de praktijk de samenwerking binnen het projectteam belemmeren. Het is bijvoorbeeld dikwijls zo dat designers moeilijk kunnen inschatten wat de technische mogelijkheden zijn van het systeem. Om hiervoor een antwoord te kunnen formuleren, moeten zij een zekere basiskennis hebben over de technische zijde van het systeem. Vooraleer ze echter van start kunnen gaan, is het nodig de requirements goed te begrijpen. Ze zouden dus ook kennis moeten hebben van o.a het applicatiedomein en het doel van de applicatie. De ontwikkelaars moeten op hun beurt de ontwerpen van de designers begrijpen om deze zo correct mogelijk om te zetten in code. Ook de klant, die in vele gevallen niet over enige technische achtergrond beschikt, speelt een prominente rol: hij of zij is namelijk vragende partij en beschrijft o.a. de functionele

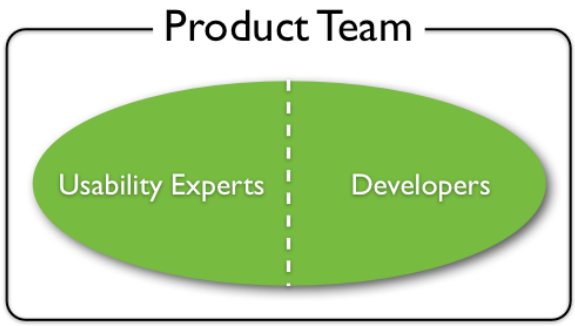
vereisten van de software. Hierbij is het nodig dat de klant samenzit met het ontwikkelteam om o.a. de technische mogelijkheden te begrijpen.

Om deze grote verscheidenheid aan disciplines te overbruggen, zou het nodig zijn om een tussenpersoon aan te stellen die bemiddeld tussen de verschillende partijen. Deze persoon zou een enorm breed kennisveld moeten beheersen: hij of zij zou namelijk thuis moeten zijn in al de bovengenoemde disciplines. Dit is niet vanzelfsprekend. Een andere oplossing is echter het gebruik van efficiënte communicatie [92]. De klant kan zijn requirements en de doelstellingen voor het project duidelijk maken aan de designers en de ontwikkelaars, terwijl zij op hun beurt eventuele problemen of moeilijkheden kunnen aanhalen en de klant helpen met het uitdrukken van zijn of haar requirements. De designers kunnen de ontwikkelaars op hun beurt bijstaan bij het omzetten van de ontwerpen in code, terwijl de ontwikkelaars de designers wijzen op de technische haalbaar van de voorgestelde oplossingen. Deze uitwisseling van ideeën en kennis vergt de nodige collaboratie en communicatie. Deze aspecten zullen dan dan ook een belangrijk ingrediënt van ons proces.

In sectie 4.4 haalden we reeds het belang van artefacten aan tijdens deze samenwerking. Ter ondersteuning van enerzijds het gebruik van deze artefacten, en anderzijds de totale workflow van het proces, zal in de komende secties het belang van een digitale tool geïntroduceerd worden. Zo een tool kan zorgen voor digitaal beheer van de data vergaard tijdens het ontwikkelingsproces. Digitaal beheer brengt vele voordelen met zich mee waaronder het eenvoudig wijzigen, opslaan en uitwisselen van data, het bijhouden van wijzigingen (zgn. versioning[10]), de toevoeging van interactiviteit en koppeling tussen verschillende artefacten en de mogelijkheid om meerdere views van dezelfde data te genereren. Vooral binnen een multidisciplinair team kan een een digitale tool enkele grote voordelen bieden. Indien de tool zo ontworpen wordt dat de artefacten zelf en de koppeling tussen deze artefacten wordt voorgesteld op een visuele manier, wordt het eenvoudiger voor teamleden met verschillende achtergronden om een beeld te krijgen van de noden van de software zonder gebruik te maken van complexe notaties en dergelijke. Wanneer het projectteam verspreid is over meerdere locaties, biedt een digitaal medium bovendien het voordeel dat data efficiënter uitgewisseld kan worden. Doorheen de volgende secties zullen we telkens waar nuttig de functie beschrijven die een eventuele tool kan hebben binnen ons proces. Niettegenstaande de vele voordelen van een digitale werkwijze, mogen we deze echter niet beschouwen als een heilige graal: een digitale werkwijze biedt niet steeds de beste oplossing voor elke situatie. Het type van artefact dat aangewend wordt, hangt sterk af van de huidige fase in het project, de vaardigheden van de teamleden, en dergelijke. Zoals we zullen zien bieden “traditionele” documenten (meestal op papier) in sommige gevallen een betere oplossing dan hun digitale tegenhangers.

We gaan nu verder met de concrete beschrijving van de samenstelling van het projectteam bij ons proces. Tijdens de integratie gaan we op zoek naar een optimale verdeling van de taken onder de teamleden. De efficiëntie is daarbij het hoogst wanneer ieder teamlid instaat voor de taken en activiteiten die hij of zij het best beheerst. Het nieuwe team dat zo ontstaat zullen we het “Product Team” noemen en bestaat uit twee delen, namelijk de “Usability Experts” en de “Developers” (zie figuur 37). Deze benamingen worden doorheen de rest van deze tekst gebruikt.

[10] http://en.wikipedia.org/wiki/Software_versioning



Figuur 37: Samenstelling van het Product Team

De groep van de Usability Experts bestaan uit personen met een brede achtergrond: dit kunnen mensen met kennis van onder andere ergonomie, grafisch ontwerp, gebruikers- en taakonderzoek en UI design.

4.5.2 Opstarten van het proces

Eén van de hoofdkenmerken van het proces dat we zullen definiëren, is het potentieel om een bruikbaar en gebruiksvriendelijk systeem te ontwikkelen. Een voorafgaand onderzoek naar de context waarin het systeem zal fungeren, een identificatie van de belangrijkste requirements en het opstellen van een high-level user interface zijn daarbij essentiële ingrediënten (zoals werd aangehaald in hoofdstuk 3, sectie 3.3.1). Deze activiteiten moeten gebeuren alvorens men kan starten met het uitwerken van de applicatie en zullen dan ook plaatsvinden tijdens een aparte fase in ons proces, vergelijkbaar met de zogenaamde “cycle zero” van een Agile levenscyclus. Aangezien we gekozen hebben voor een Agile werkwijze, is deze initiële fase best van zo kort mogelijke duur en is het noodzakelijk om op een hoog niveau te werk te gaan.

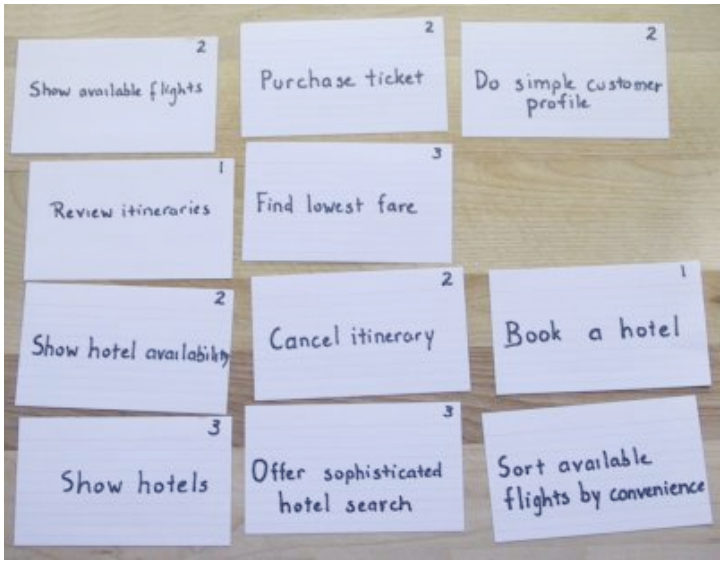
Tijdens de activiteiten binnen deze cycle zero is telkens het volledige Product Team betrokken. De Usability Experts zijn namelijk in bezit van de nodige expertise voor bijvoorbeeld het uitvoeren van field studies of het uitwerken van een initiële interface mock-up, terwijl de Developers de technische haalbaarheid kunnen controleren, kennis over de gebruikers en hun taken kunnen opdoen en de mogelijkheid krijgen om de basisarchitectuur van het onderliggende systeem uit te werken.

Indien mogelijk kan men, om tijd te besparen, gebruik maken van data uit vorige projecten. Wanneer dit echter niet het geval is, is het uitvoeren van een gebruikersonderzoek en een taakanalyse noodzakelijk [82]. Quick and Dirty en Rapid etnografie (zie sectie 3.4.4) zijn in dit kader geschikte methodes voor het bestuderen van de gebruikers, omdat deze snelle resultaten garanderen.

Tijdens het gebruikers- en taakonderzoek noteren de onderzoekers (Usability Experts) informatie over respectievelijk het profiel van de gebruikers in observatie en de manier waarop zij hun taken uitvoeren (zie hoofdstuk 3 voor meer details). Hierbij denken we o.a. aan persoonsleeftijd, persoonlijk takenpakket en computervaardigheden enerzijds en anderzijds het gebruik van tools, communicatie met andere personen, het gebruik van in- en uitvoerapparaten, kortom de totale workflow. Personas en scenario’s zijn ideale notaties

voor het documenteren van de hierbij vergaarde informatie (zie sectie 3.4.8). Personas stellen de belangrijkste rollen binnen het systeem voor, terwijl het scenario de werkwijze van de gebruikers beschrijft evenals de interactie van deze gebruikers met hun omgeving en tussen de gebruikers onderling [22][82].

Tijdens of na het onderzoek, is het voor de Usability Experts in vele gevallen gewenst om hun vaststellingen en resultaten te delen met de eindgebruikers en de klant zodat zij hun feedback kunnen geven met als doel de bekomen data te verbeteren en te verfijnen. Het gebruik van eenvoudige artefacten is op dat moment aangeraden vanwege de lage leercurve en het feit dat ze eenvoudig te hanteren zijn. Het proces is het efficiëntst wanneer de data visueel voorgesteld wordt in schemavorm, bijvoorbeeld door middel van index cards of Post It notes. Deze kaartjes en plakkertjes kunnen snel uitgewisseld, gesorteerd, gegroepeerd en aangepast worden en hebben het voordeel dat informatie kort en bondig gehouden wordt. Dit zorgt voor flexibiliteit en een snellere communicatie binnen het team [93]. Figuur 38 illustreert het gebruik van index cards.



Figuur 38: het gebruik van index cards

Een digitaal medium zou geen ideale oplossing bieden voor dit proces vanwege het feit dat papieren, “goedkope” artefacten het toelaten sneller en efficiënter samen te werken doordat deze eenvoudig kunnen opgesteld, aangepast en geordend worden. Verder is bij dit proces geen specifieke softwarekennis vereist vanwege de eindgebruikers of klant. Daarbovenop is het dikwijls zo dat de data in kwestie “on the field” vergaard wordt. In zulke gevallen heeft men niet steeds toegang tot een digitale tool voor het invoeren van deze data.

Het is echter wel nuttig om de data uiteindelijk op te kunnen slaan in een digitale tool voor verder beheer en circulatie ervan binnen het ontwikkelteam. Voor meer informatie hierover, zie hoofdstuk 5.

Tijdens cycle zero is het verder ook mogelijk om op basis van de vergaarde personas en de scenario's enkele high-level User Stories op te stellen (zie sectie 2.4) [82]. Om de basisvereisten van het systeem duidelijk te maken, is tijdens het opstellen van de User Stories prioritering van uiterst belang. Aangezien de basisfunctionaliteit dan reeds vroeg geïmplementeerd kan worden, verkrijgt men snel feedback en kan risico reeds vroeg in het project gereduceerd worden. Hierbij geldt hetzelfde proces als zonet werd beschreven voor personas en scenario's: tijdens brainstorming met de klant en de eindgebruikers wordt er gewerkt met eenvoudige en "goedkope" media, die uiteindelijk ingevoerd kunnen worden in een digitale tool.

De bekomen informatie kan eventueel vertaald worden naar één of meerdere Storyboards (zie sectie 3.4.14) of low-fidelity prototypes [26]. Deze artefacten zorgen voor een visuele voorstelling die snel geïnterpreteerd kan worden. Voor het opstellen van een eerste user interface mock-up (wireframe), kunnen de Usability Experts gebruik maken van Parallel Design (zie sectie 3.4.11). Op basis van de data uit het gebruikers-, taak-, en contextonderzoek wordt dan, in samenwerking met de gebruikers en Developers, één van de ontwikkelde ontwerpen geselecteerd.

We merken op dat bovenstaand proces slecht van korte duur mag zijn. In een Agile proces, waarbij men typisch zéér vroeg begint met het coderen van de software, is het niet gewenst een lange tijd stil te staan tijdens de initialisatiefase. Daarom is het nodig om bijvoorbeeld de focus te leggen om enkel de belangrijkste personas en hun taken op een hoog niveau te analyseren. Verdere details worden namelijk geleidelijk aan duidelijk doorheen de komende iteraties van het proces.

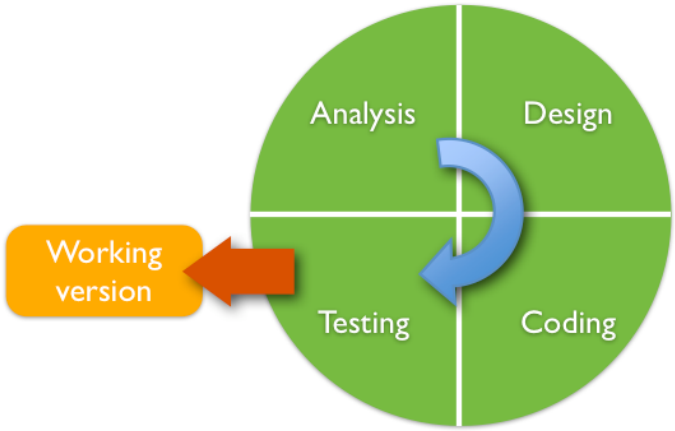
4.5.3 Iteraties met parallelle tracks

Wanneer de nodige data verzameld is in cycle zero, kan er gestart worden met de feitelijke ontwikkeling van het systeem. Zowel Agile Software Ontwikkeling als UCD kennen een iteratieve werkwijze. Het proces dat we in dit hoofdstuk definiëren, zal dan ook iteratief zijn. We moeten echter wel rekening houden met de wijze waarop we de iteraties uit beide methoden gaan integreren. Zoals we schreven in sectie 4.2, zijn er namelijk enkele belangrijke verschillen:

- De analyse bij UCD spitst zich meer toe op de eindgebruikers van het systeem, door het uitvoeren van field studies en dergelijke.
- Het design kent bij UCD een sterker iteratief karakter.
- Over het algemeen wordt bij UCD het coderen van het systeem uitgesteld totdat men zekerheid heeft dat de prototypes voldoen aan de eisen van de gebruikers en de klant. (Bij high-fidelity prototyping kan men ervoor kiezen om de achterliggende code van de prototypes ook gedeeltelijk uit te werken om de functionaliteit van het prototype te maximaliseren om zo nauwkeuriger te kunnen testen).
- In tegenstelling tot Agile Software Ontwikkeling, worden er bij UCD usability tests uitgevoerd. Bij Agile Software Ontwikkeling gebruikt men Acceptance tests voor het valideren van de geschreven code.
- Bij Agile Software Ontwikkeling werken de ontwikkelaars tijdens elke iteratie slechts enkele features van het totale systeem uit. Deze features worden geleidelijk aan samengevoegd. Een Agile proces kent dus een uitgesproken incrementeel karakter.

In hoofdstuk 3 werd reeds aangehaald dat de activiteiten in deze vijf punten een grote invloed hebben op de bruikbaarheid en gebruiksvriendelijkheid van het systeem. We zullen hier dan ook zoveel mogelijk rekening mee houden bij het definiëren van ons nieuw proces.

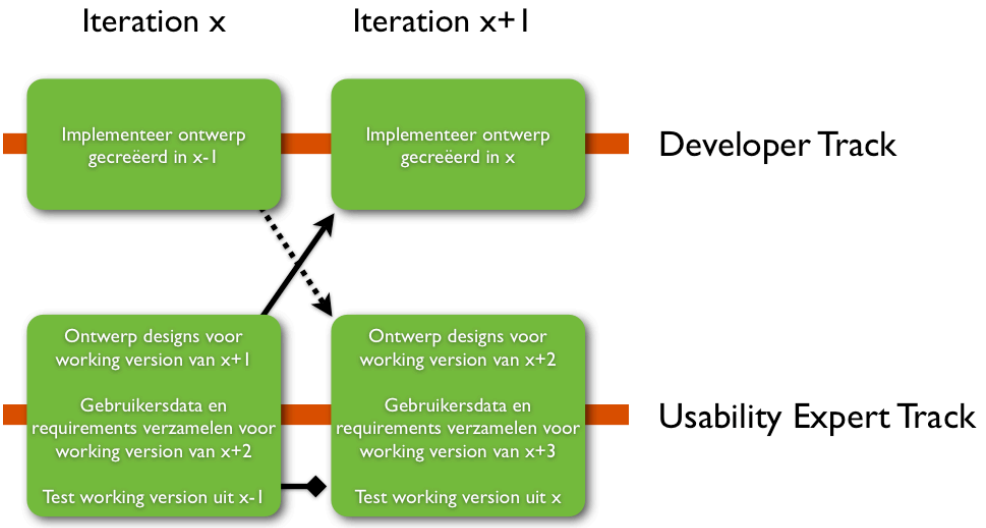
Figuur 39 toont een iteratie uit een Agile proces. Zoals werd besproken in sectie 2.4, bestaat zo een iteratie telkens uit analyse, design, implementatie en testen. De pijl in de figuur geeft aan dat deze stappen sequentieel doorlopen worden in de aangegeven (logische) volgorde. Het product van een iteratie vormt telkens een werkend, getest geheel dat kan gedemonstreerd worden aan de klant (een working version). Belangrijk opmerking hierbij is dat een iteratie bij Agile Software Ontwikkeling Timeboxed is. De deadline voor elke iteratie ligt met andere woorden steeds vast.



Figuur 39: Een iteratie binnen een Agile proces

De activiteiten in bovenstaande opsomming nemen verder relatief veel tijd in beslag. Daarbovenop is het nodig dat analyse en design steeds moeten plaatsvinden alvorens men kan starten met het coderen. Aangezien de activiteiten tijdens de analyse, het design en het testen voornamelijk taken zijn van de Usability Experts en het coderen voor het grootste deel het werk is van de Developers, zullen we ervoor kiezen om analyse, design en testing parallel uit te voeren met het coderen. Deze werkwijze zorgt ervoor dat er tijd bespaard wordt [22][50][83][84]. Dit parallellisme verandert echter niets aan het feit dat de Developers moeten wachten met coderen totdat de Usability Experts klaar zijn met de analyse en het design. Daarom is het efficiënter dat de Usability Experts steeds één of meerdere iteraties vooruit werken op de Developers. Deze werkwijze wordt geïllustreerd in figuur 40. In [22] wordt beschreven hoe een gelijkaardig proces succesvol werd aangewend bij het ontwikkelen van enkele applicaties (onder andere SketchBook) binnen het bedrijf Autodesk[11] (vroeger Alias) (zie sectie 4.3.2). Het CRUISER proces (zie sectie 4.3.1) werkt ook volgens een gelijkaardige parallelle werkwijze.

[11] http://www.autodesk.com



Figuur 40: Gescheiden design- en ontwikkelsporen

In figuur 40 zien we dat het Product Team zich opsplitst in twee sporen (oftewel “tracks”), namelijk de Developers gaan hun werk doen in de “Developer Track” en de Usability Experts in de “Usability Expert Track”. Concreet geldt de volgende werkwijze (volgorde heeft geen betekenis):

- Tijdens iteratie x werken de Usability Experts aan het ontwerp van een subdeel van de totale applicatie. Wanneer dit ontwerp voltooid is wordt het doorgegeven aan de Developers ter implementatie in iteratie x+1. (⟶ in figuur 40)
- Gelijktijdig met het ontwerp, worden in iteratie x de requirements en gebruikersdata verzameld die zullen dienen als invoer voor het ontwerp van de working version van iteratie x+2. Deze data zal dienen als startpunt voor het design in iteratie x+1. (⟶◆ in figuur 40)
- Verder voeren de Usability Experts tijdens iteratie x een usability test uit op de working version van iteratie x-1. (⇢ in figuur 40)

Om deze workflow mogelijk te maken, is het nodig dat de activiteiten binnen de Usability Expert track toegepast worden op een subdeel van de user interface in plaats van op het volledig design. We moeten het met andere woorden mogelijk maken analyse, design en evaluatie geleidelijk aan uit te voeren om zo incrementeel te werken naar een volledig afgewerkt product. Dit probleem kunnen we oplossen door “Chunking” toe te passen [22] [82]. Chunking toegepast op een user interface betekent niets anders dan het opdelen van deze interface in verscheidene “Design Chunks” die apart behandeld (meer bepaald geanalyseerd, ontworpen en getest) kunnen worden door de Usability Experts. Deze werkwijze is echter niet vanzelfsprekend, aangezien een user interface zeer complex kan zijn en dikwijls moeilijk op te delen valt in onderdelen die passen binnen de relatief korte

iteraties van een Agile proces. Een oplossing voor dit probleem wordt besproken in [22], en bestaat voornamelijk uit twee delen:

- Eerst en vooral is het nodig een reeks designdoelstellingen op te stellen die samen een overzicht geven voor het design van het volledige systeem. Elke Design Chunk laat het vervolgens toe een deelverzameling van deze doelstellingen te bereiken. Tijdens elke iteratie analyseren de Usability Experts de huidige situatie en maken op basis van de beschikbare data en resultaten uit vorige iteraties een keuze uit de designdoelstellingen die in de huidige iteratie aan bod zullen komen.
- Verder moet er rekening gehouden worden met het feit dat tijdens een Agile proces de working versions van elke iteratie telkens verder bouwen op elkaar. Dit moet dus ook zo zijn voor de Design Chunks. Daarom is het van belang dat de Design Chunks in eerste instantie de basis van de user interface definiëren. Dit zijn elementen die fundamenteel zijn voor de user interface van het systeem en die praktisch onveranderd blijven naarmate er elementen toegevoegd worden in latere iteraties.

We kunnen concluderen dat er tijdens cycle zero een horizontaal prototype opgesteld wordt voor de user interface van het volledige systeem, terwijl tijdens het iteratieve proces verticale prototypes gebruikt worden om bepaalde features van deze interface dieper uit te werken (zie sectie 3.4.11). In volgende secties wordt de werkwijze binnen één enkele iteratie besproken.

4.5.3.1 Analyse

Zoals eerder werd aangehaald, selecteren de Usability Experts tijdens iteratie x de User Stories voor de working version van iteratie x+2, en onderzoeken zij de context van gebruik van het systeem dat van belang is voor het ontwerp van deze working version in iteratie x +1. Eindgebruikers worden hierbij betrokken door het uitvoeren van etnografisch onderzoek en interviews. Door de korte duur en vaste deadline van elke iteratie, moet dit onderzoek snel en efficiënt gebeuren. Zoals in cycle zero (zie sectie 4.3.2) is Quick and Dirty of Rapid etnografie hier van toepassing. In plaats van tijdrovende interviews, kunnen er ook questionnaires en surveys gebruikt worden die men per email of post kan versturen. De taakmodellen, personas en User Stories die gecreëerd werden in cycle zero worden doorheen het proces van iteraties verfijnd en uitgebreid. Na eventuele brainstorming met de klant en de eindgebruikers, is het nodig dat de bekomen informatie gecommuniceerd wordt met de rest van het team (developers, UI designers, grafici, etc.). Zij gebruiken deze informatie namelijk voor de concrete ontwikkeling van het systeem. Vanwege de multidisciplinaire aard van het Product Team, wordt de voorkeur gegeven aan een uniforme indeling van de informatie. Een digitale tool kan hier een aanzienlijke meerwaarde bieden.

4.5.3.2 Design

Concurrent met de analyse besproken in vorige alinea, ontwerpen de Usability Experts tijdens iteratie x de user interface voor de working version van iteratie x+1. Als invoer hiervoor worden de gebruikersdata en User Stories gebruikt die in iteratie x-1 uitgewerkt werden. Het ontwikkelen van de designs gebeurt op een iteratieve manier: de Usability Experts ontwerpen de user interface met behulp van prototypes en voeren er binnen dezelfde iteratie usability tests (zie sectie 4.3.3.4) op uit in samenwerking met de eindgebruikers. Constateert men een probleem, dan gaat men terug naar de ontwerp-tafel.

Zoals we reeds zagen in sectie 3.4.11, noemt men dit proces van iteratief design Prototyping. De gebruikte prototypes hebben hier een dynamisch niveau van realisme: voor elke iteratie kan er, naarmate de complexiteit van de huidige Design Chunk en de beschikbare tijd, gekozen worden voor een low-, medium- of high-fidelity prototype. In de meeste gevallen zijn echter low-fidelity prototypes (bijvoorbeeld op papier) aangeraden omdat deze snel gecreëerd en aangepast kunnen worden [83]. Bij een hoger niveau van realisme, kan Wizard of Oz aangewend worden om de bruikbaarheid van de ontwerpen te controleren. Hierdoor is het niet nodig dat de Usability Experts de prototypes coderen om deze te testen, aangezien Wizard of Oz het toelaat een simulatie van het prototype uit te voeren, zonder dit te implementeren. (zie ook sectie 3.4.13)

Tijdens het design van de user interface wordt er gebruik gemaakt van de data die bekomen werd tijdens het voorgaande gebruikers-, taak-, en contextonderzoek. De vorm waarin deze informatie gepresenteerd wordt is van groot belang voor de efficiënte en correcte uitvoering van het designproces. Een digitale tool zou hier ondersteuning kunnen bieden door het voorzien in interactieve mogelijkheden voor het afbeelden van een gedetailleerd overzicht van het systeem, zijn gebruikers en hun taken. Op die manier kunnen de designers een beeld krijgen van de manier waarop gebruikers met het systeem omgaan en hoe het systeem past binnen hun dagelijkse workflow. Een volgende stap voor de designers zou kunnen zijn dat zij uiteindelijk de ontworpen user interface kunnen “vasthangen” aan bepaalde scenes uit het storyboard. Zo wordt het mogelijk voor de ontwikkelaars om snel een bepaalde context te associëren met de bijbehorende gebruikersinterface. Dit levert een rijke set aan informatie op voor gebruik tijdens het coderen. De detaillering van deze informatie in combinatie van een correcte interpretatie ervan door de ontwikkelaars verhoogd de kans op een geheel dat overeenstemt met de vooropgestelde eisen.

4.5.3.3 Implementatie

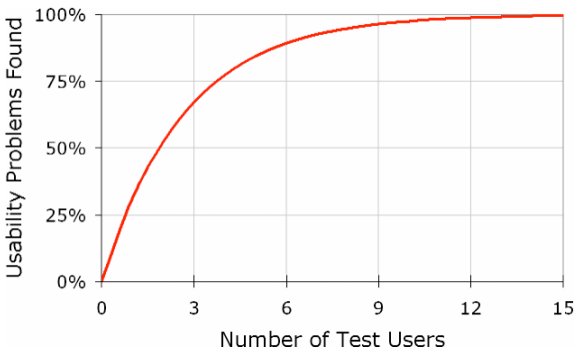
Tijdens iteratie x coderen de Developers de ontwerpen die gecreëerd werden door de Usability Experts in iteratie x-1. Tijdens deze fase in het proces is het van cruciaal belang dat de Developers de informatie uit de voorgaande analyse-, design-, en testfases juist interpreteren en correct omzetten in een gecodeerd stuk software. Hiervoor is het nodig dat zij de gebruikte artefacten begrijpen. De Usability Experts ondersteunen hierbij de Developers door details te verduidelijken en vragen te beantwoorden. Een digitale tool kan ook hier meerwaarde bieden: door het voorzien van een interactieve koppeling tussen de gebruikte artefacten en de mogelijkheid om op- of aanmerkingen toe te voegen, kan dit medium een leidraad vormen voor gebruik tijdens de ontwikkeling. Het product van deze ontwikkelingen is een working version die een afgewerkt subdeel van het totale systeem vormt.

4.5.3.4 Evaluatie

De Usability Experts voeren in iteratie x een usability test uit op de working version die de Developers voltooiden in iteratie x-1. Verder vindt er tijdens het Prototypen continue evaluatie plaats.

Usability tests nemen typisch veel tijd in beslag, aangezien er samengewerkt moet worden met de eindgebruikers om de beste resultaten te verkrijgen (zie sectie 3.3.4). Eindgebruikers zijn niet altijd ter beschikking en het inplannen van testsessies kan het proces vertragen. In [85] bespreekt Nielsen het feit dat een usability test met maximaal vijf gebruikers het meest efficiënt is. De grafiek in figuur 41 geeft de gemiddelde verhouding weer tussen het aantal

betrokken testpersonen en de gevonden fouten bij een typische usability test. Deze theorie is enkel geldig wanneer de gebruikers van de software vergelijkbare karakteristieken hebben (zoals bij websites dikwijls het geval is).



Figuur 41: Verhouding tussen aantal betrokken testpersonen en gevonden fouten bij een typische usability test [85]

In de grafiek zien we dat wanneer een usability test uitgevoerd wordt met één persoon, er reeds meer dan 25% van alle usability problemen gevonden worden. Wanneer een tweede gebruiker toegevoegd wordt, merkt deze persoon typisch een reeks problemen op die reeds gevonden werden tijdens de test met de eerste gebruiker. Bovenop deze problemen ontdekt hij of zij meestal ook een reeks andere problemen. Dit verschijnsel blijft gelden wanneer men testpersonen toevoegt. Bij 5 personen wordt er meer dan 80% van alle usability problemen gevonden. Extra gebruikers toevoegen biedt dan nog weinig meerwaarde. We zien echter dat alle problemen pas gevonden worden wanneer er 15 gebruikers betrokken zijn bij de testen, maar doordat het proces van Prototyping iteratief is, vinden er meerdere testsessies plaats. Na drie iteraties worden dan praktisch alle problemen ontdekt. Zoals we eerder echter aanhaalden, is deze werkwijze enkel van toepassing met gelijkaardige gebruikers. In de meeste gevallen kent het doelpubliek meer diversiteit en werden er verscheidene personas opgesteld. Nielsen beveelt dan ook aan om steeds drie tot vier gebruikers per gebruikersgroep - of persona - te betrekken bij elke testsessie.

Een tweede methode om efficiënt om te gaan met eindgebruikers wordt besproken in [22], namelijk het werken met een variatie aan testgebruikers. Tijdens de eerste fases van het proces kan er gewerkt worden met representatieve gebruikers. Dit kunnen interne personen zijn die het testen van de prototypes op zich nemen. Hun taak bestaat erin de prototypes te testen op fouten en de functionaliteit in zijn geheel te testen. Niets gaat echter boven de werkelijke eindgebruikers. Daarom komt in de latere iteraties het profiel van de testers geleidelijk aan korter bij het profiel van de werkelijke eindgebruikers. Deze werkwijze houdt rekening met de korte iteraties van een Agile proces, terwijl er toch gestreefd wordt naar gebruiksvriendelijkheid. Bij een gebrek aan tijd kan de evaluatie ook gebeuren door experts, maar een finale test door eindgebruikers mag niet ontbreken (zie ook sectie 3.3.4).

4.5.3.5 Bijzonderheden

Wanneer we het verloop van het totale proces bekijken, merken we op dat tijdens de eerste iteratie de Usability Experts bezig zijn met een eerste user interface design en er nog geen user interface componenten voltooid zijn. De Developers zouden dus in principe moeten wachten tot iteratie 2 voordat zij kunnen beginnen met de implementatie. Een betere oplossing is echter dat zij zich tijdens iteratie 1 bezig houden met het implementeren van achterliggende softwarearchitectuur, waarvoor weinig of geen user interface design nodig is. Een goed voorbeeld hiervan is het ontwikkelen van een database schema.

Verder zijn er verschillende partijen betrokken tijdens de uitvoering van het proces, met elk hun eigen functie. De verscheidenheid aan taken, verantwoordelijkheden en kennis kan zorgen voor communicatieproblemen, aangezien ieder groep zijn eigen “taal” spreekt. Om de kans tot slagen te vergroten is het daarom belangrijk dat er goed kan worden samengewerkt. Om dit mogelijk te maken zijn er consistente communicatiemiddelen nodig, manieren om informatie uit te wisselen die door iedereen begrepen kunnen worden. Het is dan ook essentieel dat de artefacten die gebruikt worden ter ondersteuning van de communicatie flexibel zijn. De korte iteraties binnen een Agile proces laten het immers niet toe uitgebreide en formele documenten te hanteren. Een ideaal medium hiervoor is een visuele voorstelling van de data (eventueel digitaal). Er wordt best gebruik gemaakt van visuele voorstellingen van de data, bijvoorbeeld user-interface mock-ups, storyboards of index cards met korte stukken tekst. Binnen een Agile proces gebeurt de meeste communicatie mondeling in groep. Tijdens deze gesprekken is het handig dat de documentatie steeds zichtbaar is voor iedereen en rechtstreeks kan aangepast worden. Ook de eindgebruikers moeten actief kunnen deelnemen, dus ook zij moeten de gebruikte documenten begrijpen.

Een belangrijke voorwaarde opdat het proces succesvol uitgevoerd kan worden, is dat de Usability Experts maximaal gebruik maken van het contact met de eindgebruikers. We hebben gezien dat tijdens elke iteratie de eindgebruikers betrokken worden tijdens zowel de analyse, het design als de evaluatie. Het is dus van belang de tijd wanneer de gebruikers beschikbaar zijn optimaal te benutten, aangezien deze tijd in de praktijk vaak zeldzaam is.

4.6 Voordelen van integratie

De integratie van UCD in Agile Software Ontwikkeling resulteert in een reeks voordelen. Doordat de eindgebruiker een centrale plaats krijgt in het beschreven ontwikkelproces, kent de software een groter gebruiksgemak en dit terwijl de voordelen van Agile Software Ontwikkeling behouden blijven. Doordat de focus verlegd wordt van de klant naar de eindgebruiker, krijgt het projectteam een beter inzicht op de échte behoeften van de uiteindelijke gebruikers van de software.

De Usability Experts zorgen ervoor dat de designs grondig getest en verfijnt worden voordat de ontwikkelaars starten met de implementatie ervan. Het resultaat hiervan is dat het risico op een ongewenst resultaat kleiner is.

De opvolging van een IT project gebeurt door de verschillende stakeholders. Deze personen hebben niet allemaal dezelfde (technische) achtergrond. Een goed communicatiemiddel is dus noodzakelijk om de voortgang van het project duidelijk te kunnen illustreren. Projectopvolging bij Agile Software Ontwikkeling gebeurt aan de hand van afgewerkte stukken functionaliteit, in de vorm van geteste programmacode. Om een goede inschatting

te maken van de voortgang, vereist deze werkwijze dat alle stakeholders de programmacode moeten begrijpen en dat ze de functie ervan weten te plaatsen binnen de context van de volledige applicatie. Dikwijls moeten de ontwikkelaars zelf uitleg geven bij dit proces. Echter, door de integratie van UCD, is er de mogelijkheid om afgewerkte user interface designs te tonen aan de stakeholders. Deze visuele voorstelling van de applicatie geeft een beter beeld van de huidige stand van zaken en slaagt er beter in een context te scheppen voor de reeds ontwikkelde functionaliteit. Dit is mogelijk doordat er reeds in de vroegste stappen in het proces een globale user interface wordt ontworpen waaraan telkens functies toegevoegd worden. Ook de eindgebruikers en de klant kunnen zo een zicht krijgen op de voortgang en feedback geven. [26]

Door het incrementeel ontwikkelen van functionaliteit kan, zoals vermeld in de inleiding, de user interface van de uiteindelijke applicatie onsamenhangend zijn. De integratie van UCD zorgt er echter voor dat reeds vanaf de start van het project de usability van de applicatie centraal staat.

4.7 Conclusie

Dit hoofdstuk beschreef de integratie van UCD in Agile Software Ontwikkeling. Deze integratie is fundamenteel een samenvoeging van de domeinen van SE en HCI, daarom stuitten we ook op enkele verschillen tussen deze twee methodologieën. We hebben gezien dat een integratie pas succesvol kan zijn wanneer we de duur en het type van de UCD activiteiten kunnen aanpassen, respectievelijk selecteren zodat deze passen binnen de werkwijze van Agile Software Ontwikkeling. Verder is het primordiaal dat de hoofdprincipes van beide methoden in stand blijven.

Omdat we kozen om het Product Team op te delen in twee groepen en deze parallel te laten werken, is communicatie en samenwerking tussen deze twee groepen zeer belangrijk. De efficiëntie van deze communicatie wordt versterkt door het gebruik van visuele artefacten die begrijpbaar en bruikbaar zijn voor het volledige team, inclusief de eindgebruikers.

Chunking werd toegepast opdat de ontwikkeling van de user interface zou passen binnen de korte iteraties gebruikt tijdens Agile Software Ontwikkeling. Hierbij is een grondige analyse van de user interface noodzakelijk om een efficiënte opdeling te kunnen maken. We merken nogmaals op dat de tijd van de eindgebruikers optimaal benut moet worden om zoveel mogelijk data en feedback te verzamelen in de daarvoor voorziene tijd.

Het uiteindelijke proces werd op een vrij hoog niveau beschreven, dit omdat de concrete invulling van de parameters en de keuze van methoden en technieken sterk afhankelijk is van het betreffende project, het type software, de omgeving, de ervaring van het Product Team, het doelpubliek, en dergelijke.

In dit hoofdstuk refereerden we op een reeks plaatsen naar een digitale tool ter ondersteuning van het gedefinieerde proces. In het kader van deze thesis, werd dan ook zulk een tool ontwikkeld. De uitwerking en ontwikkeling van deze tool (“StoryDraw” genaamd) wordt besproken in deel 3 van deze tekst, meerbepaald in hoofdstuk 5.

Deel III:

Ontwikkeling

Hoofdstuk 5

Ontwikkeling

5.1 Inleiding

In dit hoofdstuk wordt een softwaretool besproken die het proces dat beschreven staat in hoofdstuk 4 helpt te ondersteunen. De goede uitvoering van dit proces hangt, zoals bleek uit ons onderzoek, sterk af van de samenwerking tussen teamleden met verschillende kennis en achtergrond. Verder zagen we in hoofdstuk 4 dat deze samenwerking vereenvoudigd kan worden door het gebruik van aangewezen communicatiemedia of artefacten, zoals personas en prototypes. Deze artefacten worden al dan niet digitaal gecreëerd en opgeslagen.

Zoals toegelicht werd in sectie 4.5.1, heeft elke manier van beheer van artefacten zijn voor- en nadelen. Digitale documenten hebben onder andere het voordeel dat men deze eenvoudig en efficiënt kan opslaan, doorzoeken of uitwisselen, terwijl “klassieke” artefacten zoals paper prototypes of Post It notes een lage leercurve vereisen en makkelijk te hanteren zijn tijdens bijvoorbeeld een meeting of brainstormsessie. Beide klassen van artefacten hebben elk hun toepassingsgebied. Het gebruik van klassieke artefacten werd reeds besproken in vorige hoofdstukken, evenals de nood naar een digitale versie van deze artefacten. Verder bouwend op onze bevindingen uit hoofdstuk 4, zullen we in dit hoofdstuk dan ook een softwarematige tool introduceren aansluitend aan het proces dat we eerder definieerden. Concreet werd er als implementatie bij deze thesis een applicatie ontwikkeld die het toelaat een reeks van de belangrijkste artefacten uit Agile Software Ontwikkeling en User-Centered Design te beheren. Het “storyboard” (zie sectie 3.4.14) fungeert als centraal artefact in deze applicatie. De ontwikkelde tool werd “StoryDraw” gedoopt.

Het resultaat van de ontwikkelingen werd geïntegreerd in de UI design tool “Gummy” [87]. Doormiddel van Gummy kan de gebruiker volgens het WYSIWYG[12] paradigma in enkele stappen een gebruikersinterface ontwerpen die nadien getransformeerd kan worden voor gebruik op verschillende platformen, zoals een televisie of een mobiel apparaat. Deze functionaliteit kan een complement vormen voor een storyboarding tool zoals deze besproken in dit hoofdstuk. Een mogelijkheid zou kunnen zijn dat een interfacedesign voor een bepaald apparaat gekoppeld kan worden aan een scène uit een storyboard. Op die manier kan een storyboard verrijkt worden met bijvoorbeeld mockups van de uiteindelijke UI, wat de overgang naar de implementatie kan versoepelen. Derhalve dus deze integratie. Meer informatie over Gummy is te vinden in [87].

In de komende sectie zullen we het concrete doel van de ontwikkelde applicatie aanhalen. Vervolgens bespreken we reeds bestaande oplossingen voor het zojuist aangehaalde “probleem” in sectie 5.3, gevolgd door een bespreking van de verschillende artefacten en rollen waarvoor de applicatie ondersteuning biedt (sectie 5.4). In sectie 5.5 worden enkele voorbeelden besproken van het gebruik van de tool in de praktijk, om uiteindelijk af te sluiten met een conclusie in sectie 5.6.

---

[12] http://en.wikipedia.org/wiki/WYSIWYG

5.2 Doelstelling

Concreet kunnen we als doel een softwaretool stellen die een multidisciplinair team helpt tijdens de ontwikkeling van een softwareapplicatie. Hierbij moet het mogelijk zijn om in teamverband te werken aan de artefacten die - in dit geval - digitaal beheerd worden. Verder is het nodig dat alle artefacten onafhankelijk beheerd moeten kunnen worden, maar ook koppeling en een zekere interactie tussen de artefacten moet mogelijk zijn.

In sectie 4.5 drukten we de wens uit om over een digitale medium te beschikken voor het beheren van de geproduceerde artefacten tijdens ons ontwikkelingsproces. Dit medium kan verscheidene vormen aannemen. De eenvoudigste oplossing is simpelweg voor elk artefact een digitale versie te creëren. Voor elk persona kan er bijvoorbeeld een index card opgemaakt worden die uitgewerkt wordt in samenwerking de eindgebruikers en de klant, om vervolgens dit persona in te voeren in een digitaal medium (bijvoorbeeld een PDF bestand). In sectie 4.5.1 haalden we reeds de belangrijkste voordelen aan van digitale verwerking van deze data. Deze aanpak resulteert echter in weinig voordelen ten opzichte van de traditionele manier van werken:

- Het is niet eenvoudig om wijzigingen bij te houden tijdens het werken in teamverband
- Alle personen binnen het multidisciplinaire team werken steeds met één en dezelfde representatie van de data, onafhankelijk van hun achtergrond
- Geen mogelijkheid voor het linken van artefacten onderling

Een mogelijke oplossing zou kunnen zijn om een zogenaamd “versioning” systeem te gebruiken voor het bijhouden van de artefacten in bijvoorbeeld PDF formaat of als afbeeldingen. De mogelijkheid om referenties te maken tussen artefacten of om wijzigingen visueel voor te stellen, zijn echter niet mogelijk met zulk een systeem. Een gespecialiseerde tool zou hier beter geschikt zijn.

In het proces dat we definieerden in hoofdstuk 4, kwamen we tot de constatering dat het zogenaamde Product Team best opgedeeld kon worden in Usability Experts enerzijds en Developers anderzijds. De tool die we voor ogen hebben, zal twee gebruikersdoeleinden hebben, namelijk

1. het creëren van artefacten door het invoeren van verworven data en
2. het gebruik van de gecreëerde artefacten voor het ontwikkelen van het softwareproduct.

De creatie van de artefacten in de tool zal voornamelijk gebeuren tijdens de analyse- en designfase door de Usability Experts, terwijl de Developers deze artefacten gebruiken wanneer zij de software ontwikkelen. Deze overgang moet voldoende ondersteund worden door de tool. Door het iteratief en incrementeel karakter van softwareontwikkeling, is het verder noodzakelijk dat StoryDraw het toelaat flexibel om te gaan met de artefacten.

Zoals reeds aangehaald in de inleiding, zal er gewerkt worden met het storyboard als hoofdartefact. Andere ondersteunde artefacten zijn:

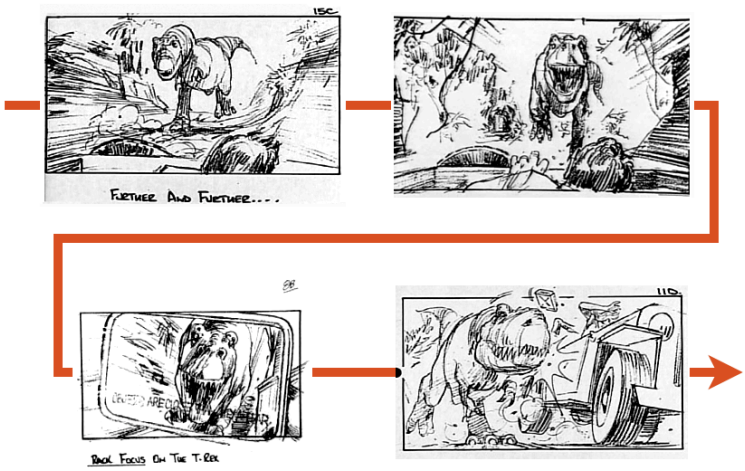
- **Scenario’s**
  Scenario’s worden vaker gebruikt binnen UCD dan bij Agile Software Ontwikkeling. Ze

worden aangewend bij de analyse van de workflow van eindgebruikers. (zie ook sectie 3.4.8)

- **Personas**
  Een persona is een typisch artefact van een UCD proces. Het is een fictief persoon die een bepaalde gebruikersgroep voorstelt of representeert. (zie ook sectie 3.4.8)
- **User Stories**
  Bij Agile Software Ontwikkeling gebruikt men User Stories als een manier om de requirements voor een bepaalde iteratie duidelijk te maken. (zie ook sectie 2.4)
- **Annotaties**
  Een annotatie is een meer algemeen artefact, dat zowel binnen UCD als bij Agile Software Ontwikkeling gebruikt wordt om details van bijvoorbeeld een gebruikersinterface te verduidelijken. Ontwikkelaars kunnen bijvoorbeeld aan de hand van annotaties feedback geven op het werk van de designers.

De keuze voor deze artefacten vloeit voort uit onze bevindingen uit sectie 4.5, waar we een plaats gaven aan een digitale oplossing voor het beheer van deze artefacten.

Vooraleer we verder gaan, geven we eerst een korte uitbreiding op onze bespreking van storyboards uit sectie 3.4.14. Storyboards vinden hun oorsprong in de animatie- en filmindustrie, waar ze ingezet worden om een verhaallijn te vertellen in de vorm van zogenaamde scènes of “snapshots”. Figuur 42 geeft een voorbeeld van een storyboard gebruikt bij het produceren van de film “Jurassic Park”. Door het gebruik van storyboards wordt het onder andere mogelijk om als regisseur op een goedkope en relatief snelle manier bepaalde concepten uit te werken of om discontinuïteiten in het verhaal te identificeren. De eenvoudige visuele representatie maakt het mogelijk om als team aan een film te werken en om zich in de schoenen van de kijker te stellen. [88] Gezien de kenmerken van deze techniek is deze ook inzetbaar in de software engineering, zoals we reeds zagen in sectie 3.4.14.



Figuur 42: Storyboard voor bepaalde scène uit Jurassic Park [89]

De gebruiksvriendelijkheid en efficiëntie van de applicatie werd uiteindelijk geëvalueerd tijdens een gebruikerstest. De resultaten van deze test zijn terug te vinden in hoofdstuk 6.

5.3 Gerelateerd werk

Vooraleer we verder gaan met de bespreking van onze tool, is het nuttig om eerst even stil te staan bij bestaande applicaties die een oplossing kunnen bieden.

Een voorname speler op het gebied van productie van diagrammen en schema's is Microsoft Visio[13]. Een storyboard is in essentie ook een diagram, dus dit softwarepakket kan tevens ingezet worden voor het maken van storyboards. Echter zal het resultaat zich slechts beperken tot afbeeldingen van scènes die men eventueel kan verbinden. Deze werkwijze biedt weinig meerwaarde ten opzichte van een storyboard "op papier". Andere gelijkaardige softwarepakketten zoals OmniGraffle[14] of Dia[15] bieden dezelfde mogelijkheden. Vaak wordt er ook presentatiesoftware zoals Microsoft PowerPoint[16] gebruikt om eenvoudige storyboards te creëren, met dezelfde nadelen als de eerder genoemde pakketten.

Er bestaat echter een plugin voor Microsoft Visio, namelijk Stpsoft Storyboarding[17]. Deze uitbreiding maakt onder andere prototyping van gebruikersinterfaces mogelijk die vervolgens kunnen gebruikt worden als onderdeel van een storyboard. Verder kunnen requirements hiërarchisch worden bijgehouden. Er wordt echter geen ondersteuning geboden voor twee belangrijke artefacten, namelijk scenario's en personas. Deze documenten zijn essentieel wanneer de noden en eisen van de eindgebruiker hoog in het vaandel worden dragen, zoals dat bij User-Centered Design het geval is. In onze implementatie zullen we dan ook deze artefacten zo goed mogelijk proberen te ondersteunen.

Buiten de ondersteuning voor een reeks essentiële artefacten, is de opzet van StoryDraw een totaaloplossing bieden die voldoende ondersteuning kan bieden aan een multidisciplinair projectteam (zoals datgeen in uit het proces in hoofdstuk 4). Zoals vermeld werd in sectie 4.5.1, bestaat zo een team uit personen met een verschillende achtergrond. Dit kan de samenwerking en communicatie binnen het team bemoeilijken. Met StoryDraw willen we een tool bieden die het projectteam helpt samen te werken.

5.4 Ondersteunde artefacten en rollen

Zoals we reeds aanhaalden in sectie 5.3, is het hoofdartefact van StoryDraw het storyboard. In sectie 3.4.14 zagen we dat een storyboard bestaat uit een verzameling "snapshots" of "scènes" die zorgen voor een illustratie van de uiteindelijke gebruikersinterface. Ook zagen we dat storyboards het mogelijk maken om een context van gebruik te schetsen door het

---

[13] http://office.microsoft.com/nl-nl/visio/

[14] http://www.omnigroup.com/applications/OmniGraffle/

[15] http://projects.gnome.org/dia/

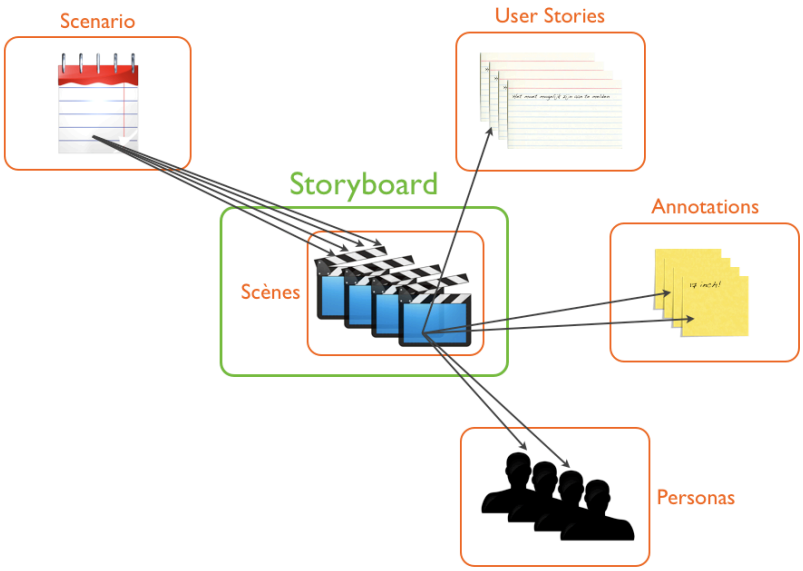
[16] http://office.microsoft.com/nl-nl/powerpoint/

[17] http://www.stpsoft.co.uk/stpbadeveloper3.html

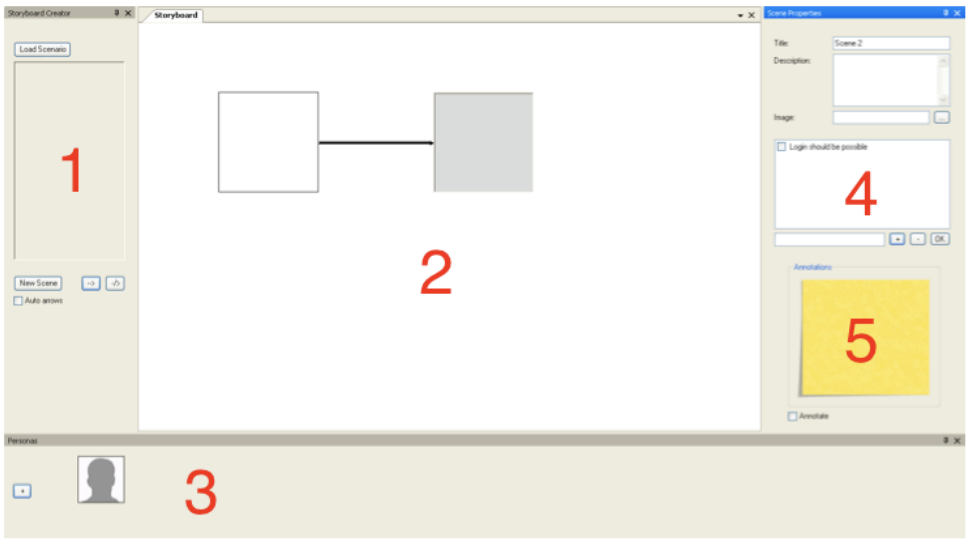
afbeelden van bepaalde situaties (zie figuur 27). Mede vanwege de integratie met Gummy is deze laatste mogelijkheid het hoofddoel van StoryDraw, aangezien Gummy ontwikkeld is met het oog op het produceren van software die ingezet kan worden op meerdere platformen, en dus meerdere contexten. Verder zien we dat er tegenwoordig meer en meer vraag is naar toepassingen die in verscheidene omgevingen (zoals bijvoorbeeld onderweg of in de woonkamer) inzetbaar zijn [87]. Omwille van deze feiten is StoryDraw ontworpen met deze filosofie in het achterhoofd.

5.4.1 Algemene structuur applicatie en gebruikersinterface

Vooraleer we verder gaan met het bespreken van StoryDraw, zullen we eerst een overzicht geven van hoe de verschillende artefacten ondersteund worden. Figuur 43 toont de hoofdelementen van StoryDraw. Figuur 44 beeldt de gebruikersinterface van de tool af. Details worden verduidelijkt in volgende secties.



Figuur 43: Koppeling van artefacten in StoryDraw

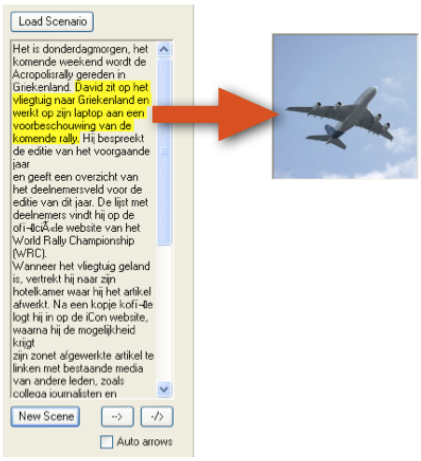


Figuur 44: Overzicht van de UI van de StoryDraw

5.4.2 Scenario's

Tijdens de analysefase stellen de Usability Experts een scenario op voor de te ontwikkelen software. Dit stuk tekst wordt samengesteld in samenwerking met de eindgebruikers met behulp van informatie verkregen tijdens het gebruikers- en taakonderzoek (zie ook sectie 3.4.8 en sectie 4.5.2). Het scenario beschrijft de stappen die de eindgebruikers doorlopen wanneer ze met de software omgaan. Dit houdt niet enkel informatie in die handelt over rechtstreekse interacties met de software, maar ook andere informatie die invloed kan hebben op de interactie van de gebruiker met de software (bijvoorbeeld een evenement of gesprek). Na eventuele brainstorming met de klant en de eindgebruikers met behulp van enkele visuele artefacten (bv. index cards), kan het scenario ingeladen worden uit een tekstbestand op de harde schijf (nr. 1 in figuur 44). Onderzoekers hebben dus de mogelijkheid om op basis van veldstudies een scenario uit te werken en in te voeren op een apparaat naar keuze (bijvoorbeeld een laptop of een netbook), om uiteindelijk dit scenario in te laden in StoryDraw.

Zoals we zagen in hoofdstuk 3, is een scenario echter niet het meest geschikte artefact om te communiceren over software met onder andere klanten, eindgebruikers en ontwikkelaars. Hiervoor is een visuele representatie zoals een storyboard beter geschikt. De overgang van scenario naar storyboard kunnen we als volgt zien: een scène uit een storyboard is eigenlijk niets meer dan een illustratie van een bepaald stuk tekst uit het scenario. In de tool wordt dit concept naar voren gebracht doordat de gebruiker een deel van het scenario kan markeren en vervolgens een nieuwe scène kan aanmaken gekoppeld aan die selectie (zie figuur 45). Vervolgens kan er een titel en een afbeelding toegevoegd worden aan de scène.

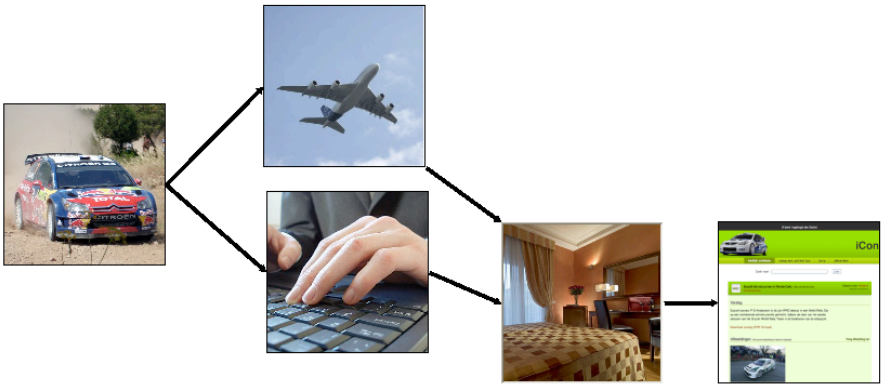


Figuur 45: Het creëren van een scène aan de hand van informatie uit een scenario

Om de communicatie met de klant en de eindgebruikers te versoepelen, is het aangeraden om initieel te werken met een storyboard “op papier”, dat later omgezet wordt in een digitale versie met behulp van StoryDraw. Deze werkwijze vereenvoudigd en versnelt het ontwikkelproces, terwijl de voordelen van digitaal beheer behouden worden.

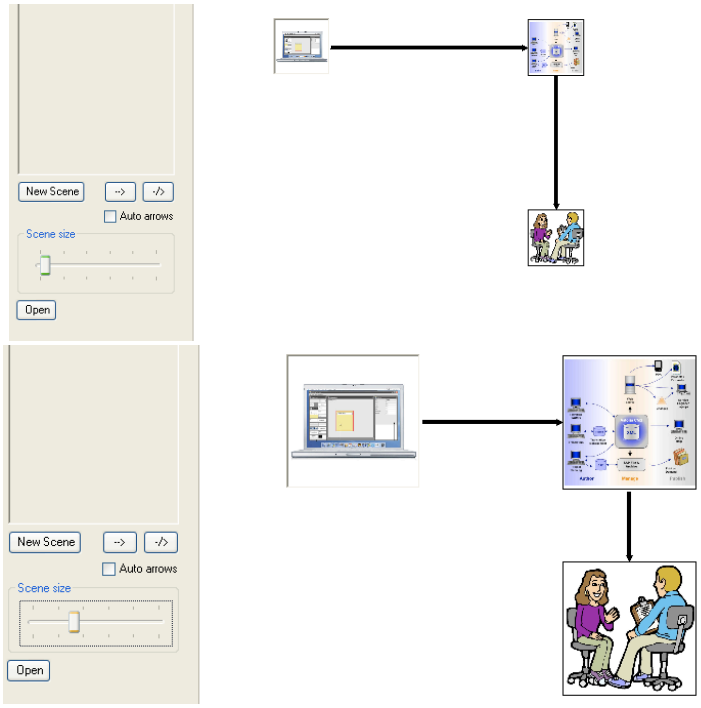
5.4.3 Scènes

Alle gecreëerde scènes worden op de centrale canvas (nr. 2 in figuur 44) geplaatst, waarna de flow van het storyboard kan worden aangeduid door het toevoegen van pijlen. Hiervoor zijn er interface-elementen voorzien. Er kan echter ook gekozen worden om automatisch de scènes te verbinden op basis van de volgorde van hun gekoppelde stukken tekst uit het scenario. De gebruiker kan de schikking van de scènes aanpassen door deze te slepen naar de gewenste positie. Tijdens het verplaatsen van scènes worden de pijlen automatisch herschikt, waardoor het resultaat correct en overzichtelijk blijft. Het grote voordeel van deze werkwijze is dat de gebruiker kan kiezen hoe het storyboard er zal uitzien om zo te voldoen aan het mentale model wat hij of zij heeft van de flow van de software (zie figuur 46).



Figuur 46: voorbeeld van een storyboard in StoryDraw

Indien gewenst kan de gebruiker de grootte van de scènes dynamisch aanpassen. Dit kan door de daarvoor voorziene slider te verschuiven (zie figuur 47).

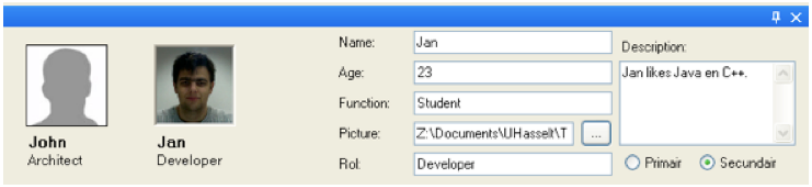


Figuur 47: de grootte van scènes aanpassen

5.4.4 Personas

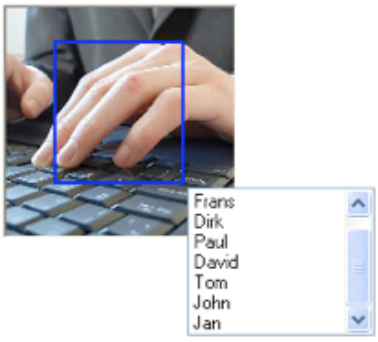
Zoals vermeld werd in sectie 3.4.8, bestaat er een belangrijke relatie tussen scenario's en personas. De actoren die handelen in een scenario, zijn namelijk de personas van de software. In StoryDraw is een aparte panel voorzien voor het beheren van de personas (nr. 3 in figuur 44, figuur 48). Een persona in StoryDraw kan volgende eigenschappen hebben:

- Naam
- Leeftijd
- Functie
- Rol
- Beschrijving
- Niveau (primair of secundair)
- Afbeelding



Figuur 48: Beheren van personas

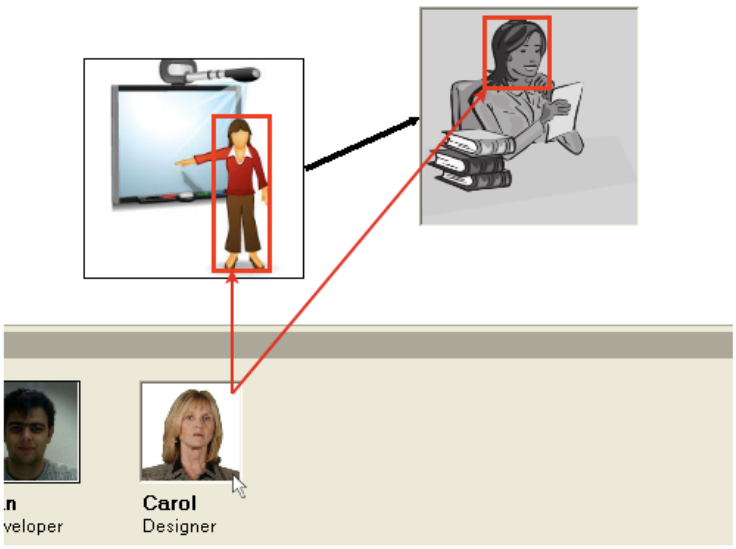
De relatie tussen een scenario en bijbehorende personas wordt duidelijk gemaakt in StoryDraw door de namen van personas automatisch aan te duiden in het scenario door de deze in het vet te zetten. Een tweede koppeling is de koppeling tussen de personas en de aangemaakte scènes. Het komt namelijk vaak voor dat een persona afgebeeld staat in één of meerdere scènes. Het segment in een scène waarin een persona voorkomt kan vervolgens aangeduid worden door de gebruiker en gekoppeld worden met een persona (zie figuur 49).



Figuur 49: Koppelen van een persona aan een scène

Eénmaal een persona geassocieerd is met een subdeel van een scène, is het op twee manieren mogelijk om informatie over deze koppeling te verkrijgen. De gebruiker kan enerzijds alle personas die bij een bepaalde scène horen opvragen; anderzijds kan voor elke scène opgevraagd worden of een bepaald persona erin voorkomt en indien dit zo is, tevens de juiste locatie van die persona in de scène. Deze functionaliteit maakt het mogelijk snel een overzicht te krijgen van bijvoorbeeld de frequentie van voorkomen van een bepaald persona in het storyboard of welke personas vaak samen communiceren. Dit is als volgt geïmplementeerd in de tool:

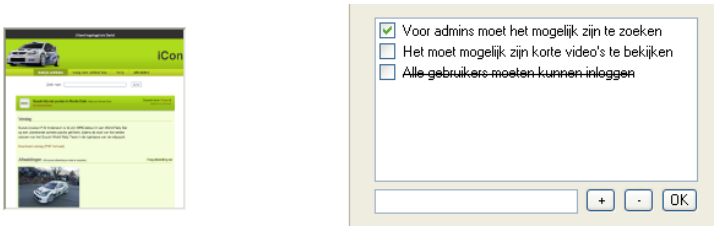
1. Door met de aanwijzer over een bepaalde scène te zweven, worden alle personas die voorkomen in die scène aangeduid.
2. Op dezelfde manier is het mogelijk om over een persona te zweven. Vervolgens worden alle plaatsen in het storyboard aangeduid waar deze persona voorkomt (zie figuur 50).



Figuur 50: Koppeling van een persona met twee scènesx

5.4.5 User Stories

Een vierde artefact dat StoryDraw ondersteunt, is User Stories. Zoals beschreven in sectie 2.4, is het mogelijk om de requirements voor een project (of voor een bepaalde iteratie binnen een project) voor te stellen door een reeks User Stories. In StoryDraw is het mogelijk een reeks User Stories op te slaan, en deze vervolgens te associëren met één of meer scènes uit het storyboard (nr. 4 in figuur 44). Wanneer de gebruiker vervolgens een scène selecteert, worden de gekoppelde User Stories aangeduid (zie figuur 51).



Figuur 51: De mogelijkheid om User Stories bij te houden en te koppelen aan scènes

Wanneer het storyboard wordt doorgegeven aan de Developers ter implementatie, kunnen zij een User Story markeren als voltooid (story 3 in figuur 51).

5.4.6 Annotaties

Tenslotte is het mogelijk om annotaties toe te voegen aan subdelen van een scène. Deze functionaliteit is omvat in StoryDraw met het oog op het geven van extra details aan de developers wanneer deze de software implementeren. Dit kan bijvoorbeeld gaan over annotaties met betrekking tot hardwaremogelijkheden en dergelijke. Ook de developers zelf kunnen dit artefacten gebruiken om feedback te geven aan bijvoorbeeld de designers of analisten, onder meer wanneer een bepaald ontwerp technisch moeilijk, of niet, te implementeren valt. De implementatie in StoryDraw is gelijkaardig aan de functie voor het aanduiden van personas in een scène: de gebruiker kan een gedeelte van een scène selecteren en hierbij een notitie voegen (zie figuur 52). Doordat een annotatie in StoryDraw bestaat uit een stuk tekst zonder een vaste vorm, is deze uiterst flexibel en kunnen mensen met verschillende achtergrond en kennis eenvoudig gebruik maken van dit artefact.

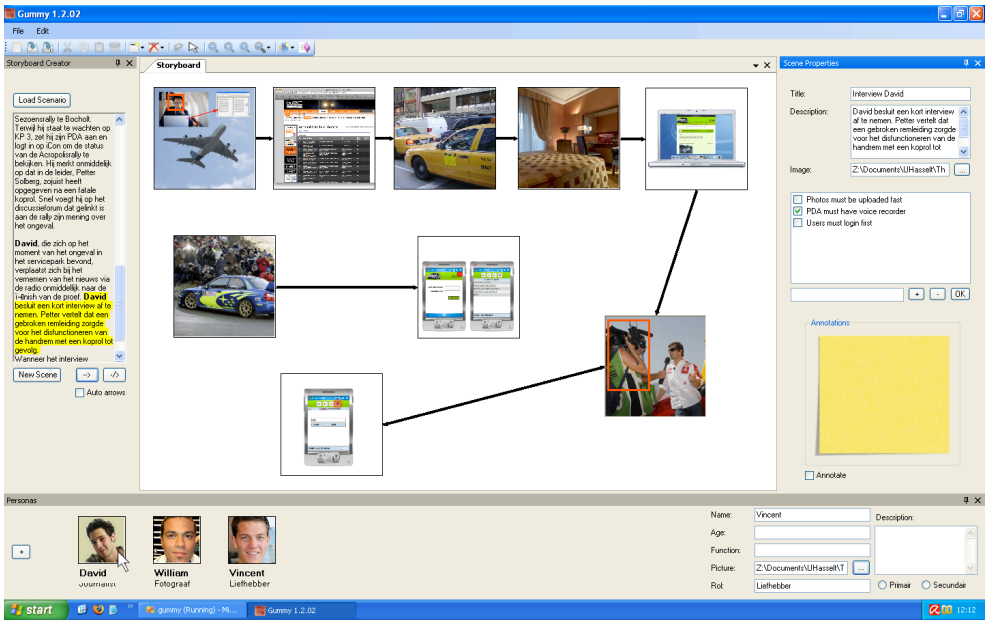
Figuur 52: Annotaties toevoegen bij een deel van een scène

5.5 Toepassingen

In deze sectie zullen we enkele voorbeelden geven van hoe de ontwikkelde tool gebruikt kan worden in praktische situaties. We gaan van start met een storyboard voor het project van het vak Gebruikersgerichte Systeemontwikkeling [40]. Vervolgens zullen we een storyboard bouwen voor het gebruik van StoryDraw.

5.5.1 Storyboard GSO

In dit project werd een scenario uitgewerkt voor een systeem waarin verschillende gebruikers een applicatie (“iCon” genaamd) gebruiken voor het uitwisselen van tekst, foto’s en videobeelden. Deze applicatie kan opereren op een PDA of desktopsysteem. Bijlage A bevat het scenario en de personas die gecreëerd werden voor gebruik tijdens het project. Vertrekkende van deze artefacten, werd voor een bepaald deel uit het scenario het storyboard in figuur 53 uitgewerkt doormiddel van StoryDraw.

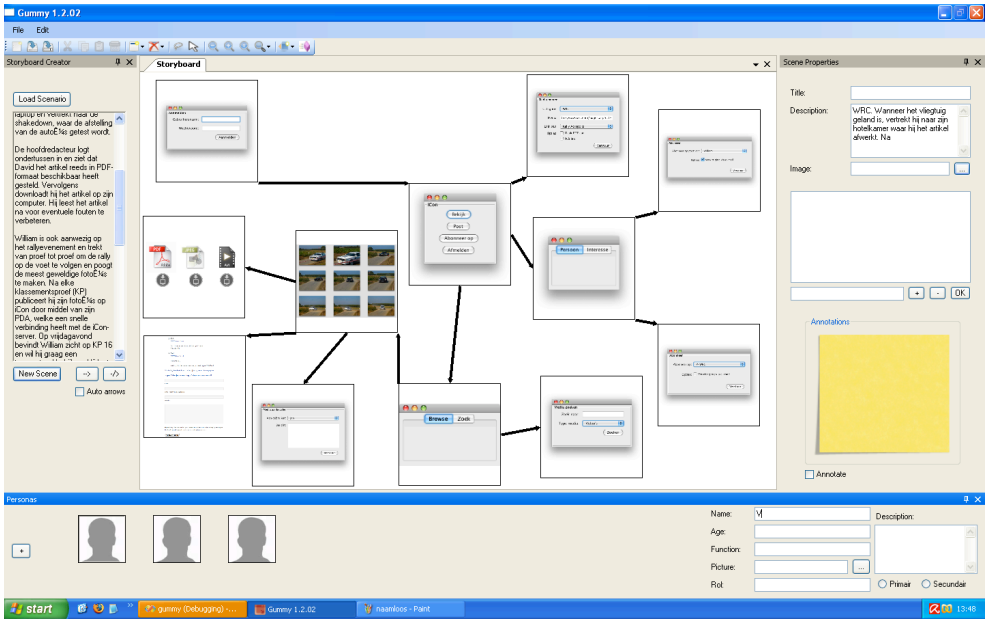


Figuur 53: Storyboard voor het project GSO

We zien in figuur 53 dat er drie personas geassocieerd zijn met het storyboard. Verder zijn er 9 scènes aangemaakt vertrekkende van het ingeladen scenario. De locatie van de muisaanwijzer, namelijk op het persona “David”, zorgt ervoor dat de plaatsen waar deze persona voorkomt in het storyboard aangeduid worden (zie scène linksboven en rechtsonder). Verder zien we dat in het scenario een gedeelte tekst gemarkeerd is. Dit is het gedeelte van het scenario dat gekoppeld is aan de geselecteerde scène. In het scenario zijn ook de namen van de personas gemarkeerd in het vet.

In de eerste paragraaf van sectie 5.4, haalden we aan dat het hoofddoel van StoryDraw was om storyboards met context-informatie te produceren. Daarnaast is het mogelijk om storyboards te maken voor de gebruikersinterface van de applicatie. Figuur 54 beeldt een

storyboard af voor een low-fidelity gebruikersinterface prototype van iCon. De elementen van het storyboard zijn afbeeldingen van de low-fidelity prototypes van iCon. Merk op dat, ondanks er relatief veel scènes aanwezig zijn, het overzicht op het storyboard acceptabel blijft. Dit is mogelijk enerzijds doordat de gebruiker zelf de schikking van scènes kan bepalen, anderzijds doordat de verbindingen tussen de scènes automatisch aangepast worden aan de schikking.



Figuur 54: Storyboard met low-fidelity user interface prototype van iCon

5.5.2 StoryDraw scenario

Als extra voorbeeld van het gebruik van StoryDraw, zullen we in deze sectie de opbouw van een storyboard bespreken dat de manier waarmee er met StoryDraw gewerkt wordt binnen een multidisciplinair team. Het storyboard dat we zo bekomen, biedt een mooi overzicht van hoe StoryDraw kan geïntegreerd worden in een praktisch softwareontwikkelingsproces zoals bijvoorbeeld het proces in hoofdstuk 4.

Eerst en vooral moeten we de personas definiëren, om daarna een scenario uit te werken. Vervolgens kunnen we een reeks User Stories opstellen en eventueel enkele notities bijvoegen. Uiteindelijk maken we de bijbehorende scènes aan en koppelen we deze met de personas, het scenario en de User Stories.

5.5.2.1 Personas

Het Product Team zorgt voor de uitvoering van het project. Zoals we zagen in sectie 4.3.1, is het Product Team opgedeeld in de Developers enerzijds en de Usability Experts anderzijds. Daarnaast zijn er nog de eindgebruikers van de software. Zij zijn ook betrokken bij de ontwikkeling. Deze drie groepen zijn dan ook de personas voor ons storyboard.

5.5.2.2 Scenario

Zoals hierboven vermeld, gaan we een scenario opstellen dat een praktische situatie beschrijft waarin een Product Team werkt aan een softwareproduct, gebruik makend van StoryDraw. Het scenario is als volgt:

> *Tijdens Cycle Zero gaan de Developers samen met de Usability Experts van start met onder andere het opstellen van de scope van het project, het maken van een kostenraming en het onderzoeken van de algemene context van gebruik van de applicatie. In overleg met de Developers, werken de Usability Experts parallel aan enkele hoogniveau gebruikersinterfaces. Omdat de applicatie zal draaien op twee platformen (PDA en Desktop), beslissen ze deze prototypes te maken in Gummy. Op die manier bekomen ze snel een gebruikersinterface die ingezet kan worden op de twee platformen. Om een idee te krijgen van de algemene workflow van de applicatie, wordt er een eerste storyboard opgesteld in StoryDraw. Dit gebeurt op een SmartBoard[18] zodat er in team gebrainstormd kan worden over de storyboard.*

> *Wanneer Cycle Zero afgelopen is, wordt het Product Team gesplitst. De Developers werken de basisarchitectuur uit, terwijl de Usability Experts de gebruikersdata verzamelen die zal dienen als basis voor het ontwerp in de volgende iteratie. Hiervoor voeren zij een etnografisch onderzoek bij de eindgebruikers van de software. Vervolgens maken ze de personas op en stellen ze een scenario van gebruik op voor de huidige working version. In StoryDraw wordt het scenario ingeladen en de personas aangemaakt. Ook de requirements worden in de vorm van User Stories ingegeven.*

> *Verder is het ook de taak van de Usability Experts om op basis van de verzamelde data een design voor de gebruikersinterface op te stellen. Om een overzicht te krijgen van het systeem, werken ze in StoryDraw een storyboard uit voor de huidige iteratie. Met behulp van dit storyboard wordt de gebruikersinterface ontworpen. Er worden annotaties toegevoegd als extra informatie voor de Developers. Wanneer de vereisten voor de huidige iteratie ingevuld zijn wordt het storyboard, samen met het gecreëerde design, doorgegeven aan de Developers. Zij zorgen voor de codering. Uiteindelijk geven de Developers de implementatie aan de Usability Experts, die deze onderwerpen aan een gebruikerstest. De voltooide User Stories worden gemarkeerd in StoryDraw.*

5.5.2.3 User Stories

Ter illustratie zullen we volgende User Stories invoeren in StoryDraw:

- Uitvoeren van Cycle Zero
- Basisarchitectuur uitwerken
- Gebruikersdata verzamelen

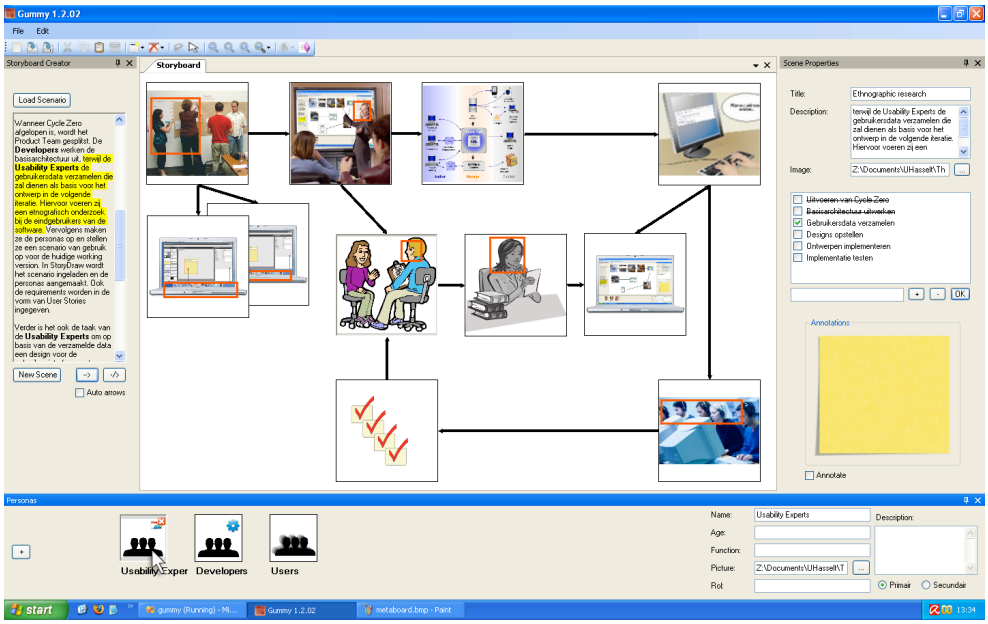
---

[18] http://www.smartboard.nl/

- Designs opstellen
- Ontwerpen implementeren
- Implementatie testen

5.5.2.4 Storyboard

Figuur 55 toont het storyboard dat geproduceerd werd aan de hand van bovenstaande artefacten. We zien dat het scenario ingeladen is en de personas en User Stories toegevoegd zijn. Wanneer we het storyboard zelf bekijken, merken we duidelijk op dat het beschreven proces iteratief werkt. Dit is te zien aan de lus rechts beneden. We zien dat de muisaanwijzer zich bevindt op de persona “Usability Experts”, met als gevolg dat de plaatsen in het storyboard waar deze persona voorkomt aangeduid worden.



Figuur 55: StoryDraw storyboard

5.6 Conclusie

In dit hoofdstuk werd StoryDraw besproken, een tool die werd ontwikkeld ter ondersteuning van het proces uit hoofdstuk 4. Een zoektocht naar bestaande oplossingen leverde weinig op, in die zin dat er geen tool kon gevonden worden die ondersteuning biedt voor enkele belangrijke artefacten zoals personas en User Stories. StoryDraw probeert een flexibele oplossing te bieden aan Product Teams voor het aanmaken en beheren van storyboards. Door de ondersteuning en onderlinge koppeling van een reeks essentiële artefacten wordt het makkelijker voor het Product Team om onder andere gebreken in de workflow van de software te ontdekken, rekening houdend met de eindgebruikers van de software. Deze eindgebruikers kunnen eenvoudig betrokken worden bij de productie van storyboards. Verder is het mogelijk om de tool in te zetten tijdens bijvoorbeeld de dagelijkse Scrum-

meeting door de applicatie weer te geven op bijvoorbeeld een SmartBoard. Doordat het mogelijk is annotaties toe te voegen bij een subdeel van een scène, is de kans kleiner dat er fouten gemaakt worden wanneer er wordt overgegaan naar de implementatie.

Ook werd het feit aangehaald dat het gebruik van een digitale oplossing zoals StoryDraw, het gebruik van klassieke artefacten niet overbodig maakt. Tijdens het proces in hoofdstuk 4 is het van belang om een combinatie te gebruiken van StoryDraw samen met de klassieke artefacten. Elk van de twee types van artefacten heeft zijn voor- en nadelen. Het correct combineren ervan kan echter de nadelen wegwerken en zo een ontwikkelingsproces efficiënt ondersteunen.

Hoofdstuk 6

Gebruikerstest

6.1 Inleiding

Met het oog op een evaluatie van de gebruiksvriendelijkheid van StoryDraw (zie hoofdstuk 5) enerzijds en de bruikbaarheid ervan binnen een multidisciplinair team anderzijds, werd een test uitgevoerd met een reeks representatieve gebruikers. In dit hoofdstuk zullen we de resultaten van de uitgevoerde gebruikerstest bespreken. Het testplan voor deze test is te vinden in bijlage B. We gaan verder met het beschrijven van de concrete doelstellingen van de test in sectie 6.2. De manier waarop de test werd uitgevoerd en de selectie van de deelnemers wordt besproken in sectie 6.3. In sectie 6.4 volgt de bespreking van de resultaten van de test. In sectie 6.5 volgt dan een discussie van de resultaten, om in sectie 6.6 te besluiten met mogelijke toekomstige uitbreidingen.

6.2 Doelstellingen

Voor een succesvolle tool zijn in essentie twee factoren van belang:

- **De tool moet gebruikersvriendelijk zijn**
  Wanneer gebruikers problemen hebben met het gebruiken van de tool omdat bijvoorbeeld sommige functies te ingewikkeld zijn of moeilijk te bedienen zijn, zullen zij de tool snel als een last beschouwen en deze links laten liggen. Gebruiksgemak is essentieel voor het slagen van een tool.
- **De tool moet de gebruiker efficiënt ondersteunen bij het uitvoeren van zijn of haar activiteiten en moet voldoende meerwaarde bieden**
  De tool moet naadloos geïntegreerd kunnen worden in de workflow en moet het werk van de gebruiker in zekere zin eenvoudiger en efficiënter maken.

Voor het toetsen van deze twee doelstellingen in de context van StoryDraw, werd een informele gebruikerstest uitgevoerd bestaande uit twee delen. In het eerste deel werd diens gebruiksvriendelijkheid geëvalueerd, terwijl in het tweede deel gevraagd werd aan de testgebruikers welke meerwaarde StoryDraw kan leveren aan een multidisciplinair projectteam en of de tool voldoende ondersteuning kan bieden tijdens een ontwikkelproces zoals hetgeen besproken in hoofdstuk 4.

6.3 Testopstelling

Voor het testen van StoryDraw werden de testpersonen opgedeeld in twee groepen, namelijk technische personen enerzijds en de personen zonder technische achtergrond anderzijds. In het totaal namen er 6 personen deel aan de test, gelijk verdeeld over de twee testgroepen (3/3). Vervolgens werd er gevraagd aan deze personen om het testplan (zie bijlage B) te doorlopen en uiteindelijk enkele vragen te beantwoorden in de vorm van een questionnaire.

6.3.1 Evaluatie

Bij het opstellen van de test, werd er beslist om de bruikbaarheid van StoryDraw te evalueren in twee stappen, als volgt:

- Als eerste stap is het nuttig om te kijken naar de noden van de verschillende teamleden binnen een multidisciplinair ontwikkelproces. Welke problemen ondervinden zij tijdens hun dagelijkse activiteiten? Hebben ze moeilijkheden met de communicatie naar de andere teamleden toe? Op welke manier wordt er gewerkt en welke artefacten worden er gebruikt voor welke doeleinden? Voor dit deel van de evaluatie zijn er personen nodig die een concrete (gespecialiseerde) rol uitoefenen binnen een projectteam. Uit hun ervaring en werkwijze kunnen we de bruikbaarheid van StoryDraw gedeeltelijk evalueren.
- Tijdens een tweede stap gaan we in op de manier waarop StoryDraw kan bijdragen aan een ontwikkelproces in zijn totaliteit, zoals het proces besproken in hoofdstuk 4. Ten opzichte van van de eerste stap gaan we in deze stap dieper in op eventuele problemen binnen een ontwikkelproces en op welke manier een tool zoals StoryDraw kan bijdragen tot de werkwijze binnen zo een proces.

6.3.2 Testpersonen

Zoals reeds vermeld, namen er zes personen deel aan de gebruikerstest. Deze personen werden geselecteerd op basis van hun technische achtergrond. De samenstelling van de testgroepen was als volgt:

- Drie personen met ervaring binnen UCD met een niet-technische achtergrond (waaronder ergonomie en grafisch ontwerp)
- Drie medestudenten met een technische achtergrond, maar ook met enige ervaring in UCD en SE. (Elk van hen volgde het vak “Gebruikersgerichte Systeemontwikkeling” [40] en twee personen namen ook deel aan het vak “Geavanceerde Software Engineering [14])

Elk van de deelnemers volgde hetzelfde testplan, maar namen achteraf deel aan een aparte questionnaire per testgroep.

6.3.3 Testplan

Het testplan opgesteld voor de gebruikerstest, is terug te vinden in bijlage B en is als volgt opgebouwd:

- **Inleiding**
  Tijdens een korte inleiding wordt er een overzicht gegeven van het doel, de gebruikersinterface en de mogelijkheden van StoryDraw.
- **Korte handleiding van StoryDraw**
  In deze sectie worden de verschillende functies besproken, hoe deze gerelateerd zijn aan elkaar en de manier waarop gebruikers deze functies kunnen bedienen.
- **Testplan**
  Het testplan omvat een reeks van 12 stappen die de testpersonen moeten doorlopen. Doorheen deze stappen wordt een storyboard opgebouwd voor een concreet scenario met bijbehorende personas, User Stories en annotaties (zie bijlage A). Tijdens het volgen van het testplan komen alle hoofdfuncties van StoryDraw aan bod. Deze opzet zorgt ervoor dat de testgebruikers een oordeel kunnen vellen over zowel de gebruiksvriendelijkheid van de tool, alsook over de bruikbaarheid ervan in een reële situatie.
- **Questionnaire 1**
  Deze eerste questionnaire is algemeen en stelt aan alle testgebruikers een reeks vragen waarmee de gebruiksvriendelijkheid van StoryDraw geëvalueerd kan worden.
- **Questionnaire 2**
  Deze questionnaire is enkel bedoeld voor de niet-technische testpersonen en is bedoeld voor het evalueren van de bruikbaarheid van StoryDraw binnen een concrete fase binnen een ontwikkelproces (bv. het design). Dit wordt onder andere gedaan door te polsen naar mogelijke problemen die de testpersonen hebben bij het uitvoeren van hun activiteiten binnen een ontwikkelproces.
- **Questionnaire 3**
  Enkel de technische testpersonen nemen deel aan deze questionnaire. De opzet is om te controleren welke plaats een technisch persoon een tool zoals StoryDraw geeft binnen een multidisciplinair ontwikkelproces zoals het proces besproken in hoofdstuk 4. Er wordt gepolst naar de mogelijkheden die StoryDraw biedt bij het ondersteunen van een ontwikkelproces in zijn totaal.
- **Mogelijkheid voor op- en/of aanmerkingen**
  In deze sectie kunnen de deelnemers opmerkingen en/of suggesties geven.

De vragen binnen elk van de questionnaires bestaan uit een reeks statements waar de deelnemers hun akkoord over moeten uitspreken. De mogelijke antwoorden per vraag werden opgesteld volgens het principe van een Likert scale[19]: telkens zijn er vijf mogelijke antwoorden, gaande van “volledig niet akkoord” tot “volledig akkoord”.

[19] http://en.wikipedia.org/wiki/Likert_scale

6.4 Resultaten

In deze sectie bespreken we de resultaten van de uitgevoerde gebruikerstest. Achtereenvolgens behandelen we de resultaten van questionnaire 1, 2 en 3.

6.4.1 Questionnaire 1

Zoals eerder vermeld werd, was het doel van questionnaire 1 het evalueren van de gebruiksvriendelijkheid van StoryDraw. De resultaten van de questionnaire zijn afgebeeld in tabel 8.

	Volledig niet akkoord	Niet akkoord	Noch akkoord, noch niet akkoord	Akkoord	Volledig akkoord
Vraag 1			1	5	
Vraag 2				4	2
Vraag 3			1	2	3
Vraag 4		2		4	
Vraag 5		1		4	1
Vraag 6		1	1	3	1
Vraag 7				3	3

Tabel 8: Resultaten gebruikerstest (questionnaire 1, gebruiksvriendelijkheid)

De beoordeling van de gebruiksvriendelijkheid van StoryDraw was, zoals te zien is in tabel 8, overwegend positief. Twee personen hadden moeilijkheden met het manueel verbinden van twee scènes, terwijl één persoon een andere manier voorstelde om de eigenschappen van een persona of scène aan te passen en om annotaties en personas te koppelen aan een scène binnen het storyboard.

Over het algemeen hadden de testpersonen nog enkele opmerkingen bij de user interface van StoryDraw:

- Automatische schikking van toegevoegde scènes (eventueel met een “klevend” raster) was een gewenste feature.
- Opbouw van het paneel voor het toevoegen van User Stories is soms verwarrend.
- Wanneer een storyboard groeit kan het overzicht van de koppeling tussen de verschillende artefacten bemoeilijkt worden. Een mogelijke oplossing hiervoor is het voorzien van kleine symbolen bij elke scène die ondermeer aanduiden welke personas bij welke scènes horen. Een handige uitbreiding zou zijn dat er aparte views konden opgevraagd worden van een storyboard, bijvoorbeeld een view voor het weergeven van enkel de annotaties.
- Bij het werken in teamverband is het nuttig om StoryDraw uit te voeren in combinatie met een SmartBoard. Deze werkwijze zorgt er verder ook voor dat bij een groot storyboard met veel scènes, toch het overzicht bewaard blijft. Het gebruik van een monitor van bijvoorbeeld 17” met een beperkte resolutie kan problemen opleveren bij het weergeven van grote storyboards.

- De tekst in het scenariopaneel is moeilijk leesbaar omwille van de korte regelafstand. Een breder paneel zou het aangenamer maken om de tekst te lezen.
- Het koppelen van een persona aan een scène kan eenvoudiger door het slepen van deze persona op de scène in kwestie, en vervolgens aan te duiden bij welke figuur in de scène deze persona geassocieerd is.

6.4.2 Questionnaire 2

Questionnaire 2 ondervraagde de niet-technische testpersonen over hun noden binnen een ontwikkelproces. Enkel voor vraag 4, 5, 6, 7 en 9 zijn er resultaten verzameld doormiddel van een Likert scale. De resultaten van deze vragen zijn te zien in tabel 9. Voor de overige vragen volgen er verder nog andere grafieken.

	Volledig niet akkoord	Niet akkoord	Noch akkoord, noch niet akkoord	Akkoord	Volledig akkoord
Vraag 4		1		2	
Vraag 5		2		1	
Vraag 6		1	1	1	
Vraag 7			1	2	
Vraag 9				1	2

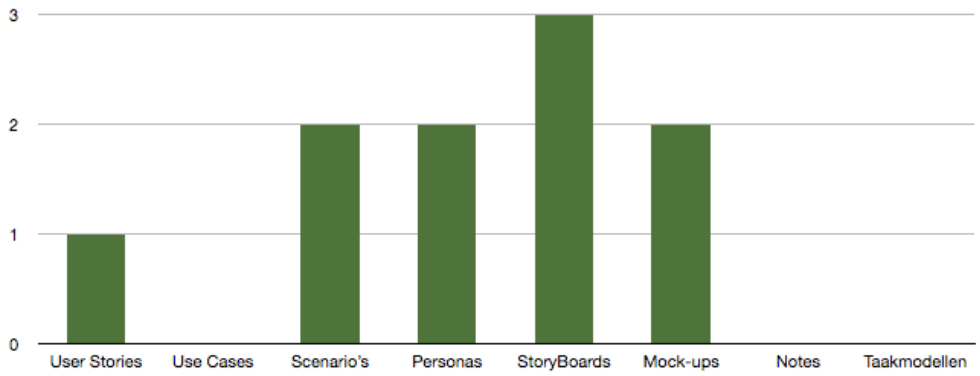
Tabel 9: Resultaten voor questionnaire 2

Uit de resultaten van deze questionnaire blijkt het volgende:

- Twee personen ondervinden problemen bij de correcte interpretatie van de resultaten van hun werk door andere teamleden (vraag 4). Deze problemen ontstaan hoofdzakelijk doordat personen uit verschillende vakgebieden samenwerken tijdens een project. Door het verschil in terminologie en kennis, is het mogelijk dat bepaalde details van bijvoorbeeld een UI design over het hoofd gezien worden. Beide testpersonen lossen dit probleem op door het gebruik mondelinge conversatie: aangezien de leden van de projectteams in kwestie zich in beide gevallen in hetzelfde gebouw bevinden, wordt de communicatie veelal mondeling gevoerd.
- Twee personen gingen niet akkoord met de stelling in vraag 5. Ze hebben namelijk geen problemen met het beheer en de uitwisseling van de gebruikte artefacten. Ze maken namelijk veelal gebruik van grote schema's op papier die mondeling verduidelijkt worden binnen het team.
- De antwoorden voor vraag 6 zijn gelijk verdeeld. Dit verschil is te wijten aan de situatie van het project in kwestie. Ter verduidelijking van de ontvangen data, geven de testpersonen vaak de voorkeur om bijvoorbeeld zelf op locatie aanwezig te zijn tijdens een gebruikerstest of observatie. Hierdoor krijgen zij een beter beeld van de dagelijkse activiteiten van de eindgebruikers.
- Twee testpersonen gingen akkoord met de stelling in vraag 7. Een visueel schema zou volgens hun de samenhang tussen de verschillende artefacten kunnen verduidelijken. Deze samenhang is niet aanwezig wanneer er gewerkt wordt met een reeks losstaande documenten.

- Alle testpersonen stelden vast dat StoryDraw een duidelijke meerwaarde kan bieden bovenop de traditionele manier van werken. (vraag 9)

Figuur 56 geeft een overzicht van de verschillende artefacten gebruikt door de testpersonen tijdens hun dagelijkse activiteiten (vraag 2).



Figuur 56: dagelijks gebruik van verschillende artefacten (questionnaire 2, vraag 2)

We zien dat het storyboard door alle testpersonen gebruikt wordt. Daarlangs worden ook vaak scenario's, personas en mock-ups gebruikt. De ondersteuning voor deze artefacten in StoryDraw, werd dan ook positief onthaald. Twee testpersonen gaven aan dat ze onbewust gebruik maken van scenario's en personas. Door het feit dat deze artefacten in de tool ondersteund worden, is het intuïtief voor deze personen om de tool te gebruiken.

Alledrie de testpersonen maken gebruik van een digitaal medium of tool voor het beheer van artefacten (vraag 3). Dit hield meestal het gebruik in van digitale bestanden zoals PDF, Photoshop, Illustrator of PowerPoint. Bij twee van de drie testpersonen werden deze bestanden beheerd met behulp van SVN. Deze werkwijze was gelimiteerd in het toevoegen van voldoende detail aan de artefacten, zonder dat het overzicht verloren gaat. Elk van de testpersonen maakt ook gebruik van artefacten op papier. Dit betreft vooral kleine aantekeningen of schetsen. Het gebruik van een digitaal medium of tool brengt in die gevallen vaak te veel overhead met zich mee. Het grote voordeel van papieren artefacten is het de mogelijkheid om eenvoudig aan deze artefacten te werken in teamverband tijdens bijvoorbeeld een brainstormsessie. Eén testpersoon gaf aan dat StoryDraw interessant kan zijn ter ondersteuning van de communicatie naar de klant toe. Een papieren storyboard geeft het gevoel dat de ideeën vast staan op papier en moeilijk of niet meer aan te passen zijn. StoryDraw zou hier een duidelijk meerwaarde kunnen bieden omwille van het feit dat een storyboard in StoryDraw eenvoudig gewijzigd kan worden. Dit geeft de klant meer het gevoel dat hij of zij ook een zekere inspraak heeft bij het creëren van de software.

Tabel 10 toont de mate waarin de verschillende testpersonen belang hechten aan bepaalde artefacten bij het uitvoeren van hun dagelijkse werkzaamheden (vraag 8).

	Niet in gebruik	Nutteloos	Neutraal	Belangrijk	Zeer belangrijk
Scenario's	1			2	
Personas				2	1
User Stories	1			2	
Annotaties				3	
Storyboards				1	2

Tabel 10: Belang van artefacten voor de verschillende testpersonen (questionnaire 2, vraag 8)

Uit tabel 10 blijkt dat de testpersonen veel belang hechten aan al de vermelde artefacten. Personas en storyboards werden bestempeld als het meest belangrijk. Alle testpersonen zagen dan ook een voordeel in de manier waarop StoryDraw al deze artefacten koppelt en samenvoegt in één interface. Hierdoor is het niet meer nodig om verschillende documenten, tools en applicaties langs elkaar te gebruiken, hetgeen resulteert in een vlottere workflow.

6.4.3 Questionnaire 3

Questionnaire 3 werd ingevuld door technische personen met kennis van UCD/Agile technieken en methoden. De resultaten van deze questionnaire zijn afgebeeld in tabel 11.

	Volledig niet akkoord	Niet akkoord	Noch akkoord, noch niet akkoord	Akkoord	Volledig akkoord
Vraag 1	1	2			
Vraag 2				3	
Vraag 3		2	1		
Vraag 4		3			
Vraag 5				3	
Vraag 6			1	2	
Vraag 7		1	1	1	
Vraag 8		2	1		
Vraag 9				3	

Tabel 11: Resultaten van questionnaire 3

Uit de oplossingen van deze vragenronde kon het volgende vastgesteld worden:

- Geen van de testpersonen ging akkoord met de stelling in vraag 1. Mondelinge conversatie is volgens hun van groot belang omdat deze manier van communiceren een grotere flexibiliteit en vrijheid biedt aan de teamleden ten opzichte van digitale communicatie (bv. via email of telefonie). Papieren artefacten hebben als voordeel dat ze het toelaten eenvoudiger samen te werken in groep omdat ze in die context makkelijker hanteerbaar zijn. Digitale tools laten het op hun beurt toe de artefacten met elkaar te koppelen en snel uit te wisselen binnen het projectteam. Verder is er automatisch versiebeheer mogelijk met een digitale werkwijze.
- Vraag 2 werd door alle deelnemers positief beantwoord. Net zoals in vraag 3 van questionnaire 2, stelden de testpersonen vast dat de flexibiliteit van de storyboards

gecreëerd in StoryDraw een pluspunt is wanneer men met de klant communiceert over het project. Zij hebben zo een groter gevoel van inspraak bij de uitvoering van het project.

- Vraag 3 werd overwegend negatief beantwoord. De leercurve van StoryDraw werd vrij laag bevonden: de deelnemers hadden gemiddeld een kwartier nodig voor het doornemen van de handleiding. Vervolgens konden ze alle stappen uit het testplan vlot doorlopen. Zelfs een stijlere leercurve zou geen probleem mogen vormen volgens de testpersonen: wanneer de tool voldoende meerwaarde kan bieden, brengt het aanleren ervan genoeg voordelen met zich mee.
- Vraag 4 werd negatief beantwoord door alle deelnemers. Binnen een Agile ontwikkelproces wordt de data geleidelijk aan verkregen en vervolgens eventueel ingevoerd in een tool zoals StoryDraw. De tijd nodig om de informatie in te voeren is dus gespreid over de volledige looptijd van het project en kan worden terugverdient door de snellere manier van samenwerken.
- Vraag 5 werd positief beantwoord door alle deelnemers. In elke stap van het ontwikkelproces kan er data worden ingevoerd in StoryDraw. Deze data blijft vervolgens zichtbaar in alle volgende fases en iteraties van het ontwikkelproces. Deze werkwijze biedt aan de teamleden een uitgebreide documentatie aan, vanaf de start van het project tot op het huidige moment.
- De antwoorden op vraag 6 waren over het algemeen positief. De visuele representatie van, en de interactiviteit tussen, de artefacten en het feit dat tekstuele informatie kort en bondig wordt gehouden, zorgt ervoor dat StoryDraw de verschillen in achtergrond tussen de verschillende teamleden helpt te overbruggen.
- De testpersonen antwoordden verdeeld op vraag 7. Ze zagen in de eerste plaats geen meerwaarde in het gebruik van een digitale tool zoals StoryDraw tijdens meetings e.d. ten opzichte van een klassiek whiteboard of projector. De bedoeling is dan ook meestal om aantekeningen te maken en snel bepaalde informatie toe te voegen. Met een onscreen toetsenbord is dit niet voor de hand liggend. Een zekere vorm van tekstherkenning is zeker nodig in zo een context. Het feit dat met deze werkwijze het projectteam steeds de beschikking heeft over de meest recente versie van het storyboard is dan wel een pluspunt.
- De antwoorden op vraag 8 waren neutraal tot negatief. StoryDraw zal het meest nuttig zijn binnen grote en gedelocaliseerde teams, maar ook binnen kleinere teams. Daar zal papier moeilijk te vervangen zijn.
- Vraag 9 werd positief beantwoord door alle deelnemers. Door de combinatie van alle data in één interface, wordt het eenvoudig voor de ontwikkelaars om snel een overzicht te krijgen van de data en deze correct te interpreteren.

6.5 Discussie

Uit de resultaten van de verschillende questionnaires en de extra op- of aanmerkingen van de testpersonen, kunnen we enkele dingen samenvatten in volgende punten:

- Buiten enkele opmerkingen over het design en het gedrag van StoryDraw, werd diens gebruikersinterface positief beoordeeld door de testgebruikers. De belangrijkste opmerkingen waren gericht op de breedte van het scenariopaneel, de beschikbare

schermgrootte wanneer de tool uitgevoerd wordt op een standaard monitor en de manier waarop personas en annotaties gekoppeld kunnen worden aan een storyboard.

- De testpersonen werken tijdens hun dagelijkse activiteiten veelal op een traditionele manier: ze maken gebruik van artefacten op papier of creëren deze met applicaties zoals een tekstverwerker of tekenprogramma. Uitwisseling gebeurt manueel of door middel van email of een SVN oplossing. Communicatie onderling gebeurt mondeling. Deze werkwijze is te wijten aan het feit dat zij steeds de mogelijkheid hebben om alle data mondeling door te praten en te bespreken. De gebruikte artefacten bieden voor deze situatie voldoende ondersteuning. De testpersonen opteren voor deze aanpak vanwege de grote flexibiliteit en snelheid waarmee er gewerkt kan worden. Toch zou een tool zoals StoryDraw volgens hun een grote meerwaarde kunnen bieden. Telkens wanneer er gebruikt gemaakt wordt van de artefacten, moet namelijk telkens weer de onderlinge link gelegd worden tussen deze artefacten. Dit vormt een obstakel voor de verloop van het proces. StoryDraw zou hier een goede oplossing kunnen bieden. Ook erkende de testpersonen het nut van StoryDraw binnen een groter project waarbij het team dikwijls gedelocaliseerd is.
- Alle teamleden zijn van mening dat mondelinge conversatie en papieren artefacten niet uit te sluiten zijn bij een ontwikkelproces zoals dat besproken in hoofdstuk 4. StoryDraw kan een traditionele werkwijze zeker niet vervangen, maar wel ondersteunen. Een goede combinatie tussen de traditionele manier van werken en het gebruik van een digitale tool brengt vele voordelen met zich mee. De manier waarop deze combinatie gemaakt wordt, hangt sterk af van het project in kwestie: over welke software gaat het? wie zijn de teamleden? welke mogelijkheden heeft het projectteam om samen te werken? etc.

6.6 Mogelijke uitbreidingen

Uit de resultaten van de gebruikerstest kunnen we ook nog enkele interessante uitbreidingen voorstellen:

- De mogelijkheid om een scène binnen StoryDraw te verrijken met één of meerdere UI designs, zou de overgang van designers naar developers kunnen ondersteunen. Op die manier hebben zij immers alle informatie bij de hand in één interface voor het ontwikkelen van de software. Een koppeling met Gummy zou hier zeker een interessante toevoeging zijn. De mogelijkheden die Gummy biedt met betrekking tot het creëren van genereren van gebruikersinterfaces voor gebruik binnen verschillende contexten, bieden een interessante toevoeging aan de functionaliteit van StoryDraw.
- Een tijdslijn waarop de momenten te zien zijn waarop er wijzigingen werden aangebracht aan het storyboard, samen met de persoon verantwoordelijk voor deze wijziging, zou de functionaliteit van een SVN systeem kunnen overnemen. Het voordeel hiervan is dat de teamleden op een visuele manier doorheen de “geschiedenis” van het project kunnen navigeren en voor elk moment in het verleden kunnen zien hoe het storyboard er op dat moment uitzag. Een SVN systeem kan bijvoorbeeld niet aanduiden wie een bepaalde koppeling tussen een reeks artefacten gewijzigd heeft en wanneer. (zie ook sectie 4.5.3.2)
- In de huidige versie van StoryDraw worden de namen van personas reeds aangeduid in het ingeladen scenario. Deze functionaliteit vereenvoudigd het overzicht op het scenario doordat men kan zien welke persona actief is op een bepaalde plaats in het scenario. Deze functionaliteit kan verder uitgebreid worden met een systeem dat bijvoorbeeld werkwoorden of voorwerpen detecteert en herkent. Deze werkwijze zou het mogelijk

maken om automatisch scènes te creëren voor bepaalde activiteiten binnen het scenario. Hiervoor kan er gebruik gemaakt worden van bijvoorbeeld WordNet[20].

6.7 Conclusie

We kunnen besluiten dat StoryDraw een zekere meerwaarde kan bieden aan een ontwikkelproces. Door het ondersteunen van de teamleden bij hun activiteiten, wordt het voor hen eenvoudiger om data over het project in kwestie te beheren en uit te wisselen. De visuele representatie van de artefacten binnen StoryDraw kan hun helpen de verschillen in hun achtergrond en kennisgebied te overbruggen.

[20] http://wordnet.princeton.edu/

Deel IV:

Conclusie

Hoofdstuk 7

Conclusie

In deze thesis werd een onderzoek verricht op het gebied van methodes voor het ontwikkelen van software. Onder meer door de evolutie van beschikbare hardware (hogere performantie, nieuwe *form factors*, e.d.) en de opkomst van communicatietechnologieën zoals het Internet, is de aard van software door de jaren heen sterk veranderd. Om kwaliteitsvolle softwareproducten te kunnen garanderen en om het proces van de ontwikkeling van software zo efficiënt mogelijk te kunnen uitvoeren, is er tevens een evolutie op het gebied van software ontwikkelingsprocessen nodig. Het doel in deze thesis was het uitwerken van een ontwikkelproces dat steunt op twee belangrijke pijlers, namelijk gebruikersvriendelijkheid van het eindproduct voor de gebruikers en een efficiënt verloop van het proces tijdens het tot stand komen van de software. De voorgestelde oplossing bestaat uit een integratie van twee bestaande ontwikkelmethodes, namelijk Agile Software Ontwikkeling en User-Centered Design.

In hoofdstuk 1 werden traditionele ontwikkelmethodes onderzocht met het Waterfall model als voorbeeld. De problemen met deze methodes werden besproken. Door het sterk uitgebreide en gecontroleerde proces, is er weinig flexibiliteit aanwezig. Deze flexibiliteit blijkt een cruciaal element te zijn voor moderne softwareprojecten. De snelle ontwikkeling op technologisch vlak, nieuwe mogelijkheden van snellere hardware en het feit dat de klant dikwijls moeilijk kan uitdrukken wat hij wil zijn katalysatoren voor het falen van inflexibele projecten.

Hoofdstuk 2 handelde over Agile ontwikkelmethodes, met als doel een oplossing te vinden voor het inflexibele karakter van traditionele methoden. Deze methodes bieden vele voordelen ten opzichte van het Waterfall model, in die zin dat er tijdens de ontwikkeling flexibeler kan omgesprongen worden met veranderingen van belangrijke parameters, bijvoorbeeld de eisen van de klant of de evolutie op technologisch vlak. Het gebruik van Agile methodes reduceert het risico, vergroot de zichtbaarheid op het project en zorgt voor snellere resultaten. Gebruiksvriendelijkheid van het eindproduct is echter een knelpunt bij Agile Software Ontwikkeling.

Hierna werd het proces van User-Centered Design uitgediept in hoofdstuk 3. User-Centered Design, met als doel het produceren van bruikbare en gebruiksvriendelijke producten door middel van een continue input vanwege de eindgebruiker, bleek een geschikt proces te zijn voor toepassing binnen softwareontwikkeling. Het gebruik van UCD tijdens het ontwikkelen van software resulteert in een bruikbaar en gebruiksvriendelijk product voor de eindgebruiker. Het uitgebreide onderzoek en het relatief trage proces, zorgen echter voor een langere duur van projecten. Ook de samenwerking met de eindgebruikers kan voor problemen zorgen.

Een oplossing voor deze problemen, werd aangereikt door de ontwikkeling van een proces dat Agile Software Ontwikkeling en User-Centered Design combineert. Het succes van deze combinatie hangt echter sterk af van de communicatie tussen de verschillende teamleden

en de gebruikte artefacten. Door de multidisciplinaire aard van het projectteam, is dit niet vanzelfsprekend.

Ter ondersteuning van het projectteam tijdens de uitvoering van het proces in hoofdstuk 5, werd StoryDraw ontwikkeld. StoryDraw is een softwaretool, met als doel de communicatie tussen de teamleden te vereenvoudigen. Door de ondersteuning en visuele representatie van een reeks artefacten eigen aan Agile Software Ontwikkeling en UCD, wordt het mogelijk om één en dezelfde notatie te gebruiken voor deze artefacten tijdens de volledige duur van het project. De interactiviteit van de tool zorgt ervoor dat personen met verschillende achtergrond zelf kunnen kiezen welke details zij te zien krijgen. Om de bruikbaarheid en gebruiksvriendelijkheid van StoryDraw te evalueren, werd er een informele gebruikerstest uitgevoerd. De testgebruikers reageerden overwegend positief op de test.

Toekomstig werk bevindt zich vooral op het gebied van het verfijnen van het proces beschreven in hoofdstuk 5 op gebied van de samenwerking tussen de teamleden. Hierbij speelt StoryDraw een belangrijke rol. Het uitbreiden van diens functionaliteit kan zorgen voor een betere ondersteuning van het proces.

Deel V:

Appendices

Bijlage A

Artefacten project GSO

Deze bijlage omvat een reeks artefacten die gecreëerd werden tijdens het vak “Gebruikersgerichte Systeemontwikkeling” [40]. Deze artefacten worden gebruikt in hoofdstuk 5 als voorbeeldgegevens bij het demonstreren van de functionaliteit van StoryDraw.

A.1 Personas

A.1.1 Vincent

Vincent is 19 jaar en studeert Informatica aan de universiteit van Hasselt. Hij zit op kot in Diepenbeek en heeft een vriendin.

Naast zijn studies is hij een fervent rallyliefhebber, dagelijks raadpleegt hij dan ook diverse bronnen over zijn favoriete sport, nl. kranten en online fora. Echter beschikt hij niet altijd over een computer, wel over een PDA. Deze gebruikt hij bij het opzoeken van nieuws op verschillende internetsites. Jammer genoeg zijn er weinig websites geschikt voor zijn PDA en mist hij dus verscheidene nieuwsberichten.

Wanneer hij een rallywedstrijd bijwoont, mist hij een overzicht van de wedstrijd, bijvoorbeeld een voorlopige tussenstand, of een lijst met opgaves. Tijdens het wachten wil hij ook graag toegang hebben tot foto’s en eventueel video’s van vorige proeven.

A.1.2 William

William is 20 jaar oud en is vrijgezel. Hij is woonachtig te Lommel en werkt als fotograaf bij een tijdschrift. In zijn vrije tijd gebruikt hij zijn fototoestel om zijn favoriete sport te fotograferen: hij is een rallyfan. Als hij terug thuis komt na de rally zet hij zijn foto’s op zijn fotoblog. Echter komen er niet veel bezoekers op zijn blog. Graag zou hij zijn foto’s kunnen linken aan nieuwsartikelen en andere fotosites over de rallysport, zo zouden zijn foto’s meer bekeken worden. Dikwijls is hij ver van huis en wil in contact blijven met zijn kennissen en zijn foto’s laten zien. Omdat hij niet over een laptop beschikt, kan hij zijn de foto’s pas online zetten als hij thuiskomt. Hij beschikt wel over een PDA die hij gebruikt om een agenda van de rally bij te houden. De manier om zijn foto’s te publiceren vindt hij wat omslachtig, hij werkt hiervoor met verschillende aparte tools.

Van computers heeft hij niet zoveel kaas gegeten. Graag zou hij beschikken over één tool die alle stappen ondersteunt.

A.1.3 David

David werkt als autosportjournalist bij een tijdschrift en is gespecialiseerd in de rallysport. Hij is aanwezig op alle grote rallywedstrijden, van het Belgisch en wereld kampioenschap. Meestal vertoeft hij in het servicepark, het rallycentrum of aan de finish van de proeven om de piloten te interviewen en om de stand bij te houden. Elke dag schrijft hij meerdere artikels waarin hij de rally bespreekt. ‘s Avonds stuurt hij zijn artikels door naar de redactie van zijn tijdschrift waar deze verwerkt worden en voorzien worden van beeldmateriaal. Het tijdschrift onderhoudt ook een website met sportnieuws. Graag zou David zelf de foto’s kiezen die best bij zijn artikels passen, nu doet de redactie dit. Hij houdt van de foto’s van zijn goede vriend William en zou deze graag gebruiken om zijn artikels te illustreren.

A.2 Scenario

Het is donderdagmorgen, het komende weekend wordt de Acropolisrally gereden in Griekenland. David zit op het vliegtuig naar Griekenland en werkt op zijn laptop aan een voorbeschouwing van de komende rally. Hij bespreekt de editie van het voorgaande jaar en geeft een overzicht van het deelnemersveld voor de editie van dit jaar. De lijst met deelnemers vindt hij op de officiële website van het World Rally Championship (WRC). Wanneer het vliegtuig geland is, vertrekt hij naar zijn hotelkamer waar hij het artikel afwerkt. Na een kopje koffie logt hij in op de iCon website, waarna hij de mogelijkheid krijgt zijn zonet afgewerkte artikel te linken met bestaande media van andere leden, zoals collega journalisten en fotografen. Hij zoekt naar foto’s van het vorige jaar en komt terecht op een fotoalbum van William. Zijn foto’s geven een mooi overzicht van de editie van vorig jaar, dus David beslist deze foto’s te verbinden met zijn huidige artikel. Daarna uploadt hij zijn artikel naar de iCon server en beslist ook een PDF bestand te laten genereren. David pakt zijn laptop en vertrekt naar de shakedown, waar de afstelling van de auto’s getest wordt.

Ondertussen is de redactie waar David werkt geïnteresseerd in een preview van de komende Acropolisrally. De hoofdredacteur van de sportafdeling vind in zijn mailbox het bericht dat David zojuist een nieuw artikel gepubliceerd heeft op iCon. Hij logt in en ziet dat David het artikel reeds in PDF-formaat beschikbaar heeft gesteld. Vervolgens downloadt hij het artikel op zijn computer. Hij leest het artikel na voor eventuele fouten te verbeteren, post het op de website van het tijdschrift en legt het klaar voor publicatie in de volgende uitgave van het tijdschrift.

Het hele weekend lang volgt David de rally, interviewt piloten, spreekt met teammanagers en schrijft verslagen en artikels die hij ook publiceert op iCon zodat geïnteresseerden deze kunnen raadplegen.

William is ook aanwezig op het rallyevenement en trekt van proef tot proef om de rally op de voet te volgen en poogt de meest geweldige foto’s te maken. Na elke klassementsproef (KP) publiceert hij zijn foto’s op iCon door middel van zijn PDA, welke een snelle verbinding heeft met de iCon-server. Elke foto bevat tags, zoals GPS coördinaten die de opnamelocatie voorstellen en het tijdstip waarop de foto gemaakt werd.

Op vrijdagavond bevindt William zicht op KP 16 en wil hij graag een tussenstand bekijken. Hij is namelijk een hevig supporter van Marcus Grönholm, die aan het begin van de dag voor de 2de plaats vocht met Sebastien Loeb. Hij logt in op iCon met zijn PDA en raadpleegt

het laatste nieuws over de Acropolisrally. Hij ziet dat Solberg van de baan is gegaan en is opgetogen dat zijn favoriet Gronhölm een plaatsje is opgeschoven.

Ondertussen bevindt Vincent zich in België op de Sezoensrally te Bocholt. Zijn budget liet het niet toe om naar Griekenland te trekken voor de Acropolisrally.

Terwijl hij staat te wachten op KP 3, zet hij zijn PDA aan en logt in op iCon om de status van de Acropolisrally te bekijken. Hij merkt onmiddellijk op dat in de leider, Petter Solberg, zojuist heeft opgegeven na een fatale koprol. Oorzaak was volgens geruchten een foute note bij zijn copiloot Phil Mills. Hij ziet ook dat William reeds een reeks foto's heeft gepubliceerd van de beschadigde auto van Solberg. Snel voegt hij op het discussieforum dat gelinkt is aan de rally zijn mening over het ongeval.

David, die zich op het moment van het ongeval in het servicepark bevond, verplaatst zich bij het vernemen van het nieuws via de radio onmiddellijk naar de finish van de proef. Hier treft hij de getakelde wagen van Solberg aan. Ook Petter en zijn copiloot Phil zijn aanwezig. David besluit een kort interview af te nemen. Petter vertelt dat een gebroken remleiding zorgde voor het disfunctioneren van de handrem met een koprol tot gevolg. Wanneer het interview afgelopen is, pakt David zijn laptop en post snel een artikel van het interview op iCon. Hij ziet ook dat er op het forum geruchten zijn over foute pacenotes bij zijn copiloot, en reageert hierop met een verwijzing naar zijn interview.

Bijlage B

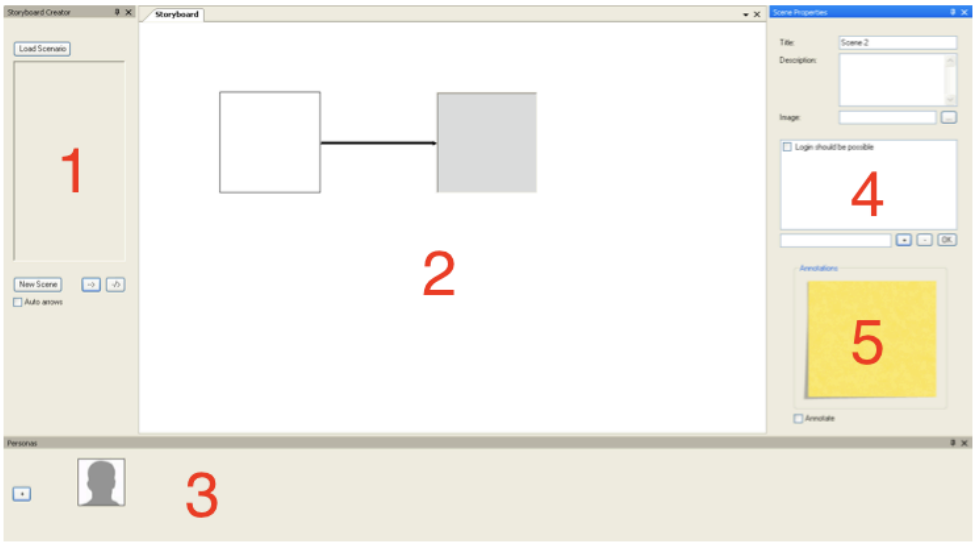
Testplan gebruikerstest

Introductie

StoryDraw is een tool die het toelaat storyboards te creëren voor gebruik binnen een softwareontwikkelingsproces waarin de principes van User-Centered Design en Agile Software Ontwikkeling aangewend worden. StoryDraw biedt ondersteuning voor volgende artefacten:

- Scenario's
- Personas
- User Stories
- Annotaties

Doormiddel van een storyboard, kunnen de gebruikers deze artefacten koppelen. In deze gebruikerstest zullen we de gebruiksvriendelijkheid en bruikbaarheid van de tool evalueren. De algemene gebruikersinterface van StoryDraw ziet er uit zoals in figuur 1.



Figuur 1: Gebruikersinterface van StoryDraw

We merken volgende hoofdgebieden op:

1. Weergave van het ingeladen scenario (tekst)
2. Hoofdpanel voor het storyboard
3. Weergave van de betrokken personas

4. Lijst met User Stories
5. Weergave van annotaties

De belangrijkste functies zijn:

Scenario's
Het is mogelijk een scenario in te laden vanaf een tekstbestand. Dit is mogelijk met de knop "Load Scenario".

Scènes
Een scène is altijd gekoppeld aan een tekstfragment uit het scenario. Om een scène aan te maken, moet er dus eerst een fragment uit het scenario geselecteerd worden. Dit kan door te slepen met de muis over de tekst en zo een fragment te selecteren. Vervolgens klikt men op de knop "New scene" en de scène wordt toegevoegd aan het storyboard.

Scènes kunnen verplaatst worden door deze te slepen, en verwijderen is mogelijk door te klikken op het sluitknopje rechtsboven op de scène (enkel zichtbaar wanneer men met de muisaanwijzer boven de scène zweeft).

Het verbinden van scènes (voor het aangeven van de sequentie binnen het storyboard) kan gebeuren in 3 stappen:

1. Druk op de knop [-->]
2. Selecteer een eerste scène
3. Selecteer een tweede scène

Vervolgens wordt er een pijl getrokken van de eerste naar de tweede scène. Deze verbinding weer opheffen gebeurt tevens in 3 stappen:

1. Druk op de knop [-/>]
2. Selecteer eerste scène
3. Selecteer tweede scène

Er kan ook gekozen worden voor de optie "Auto arrows". Deze optie zorgt ervoor dat een aangemaakte scène automatisch verbonden wordt met reeds aanwezige scènes aan de hand van de volgorde van de bijbehorende scenariofragmenten.
Het aanpassen van de eigenschappen van een scène, kan door deze te selecteren (1x klikken) en vervolgens de gewenste eigenschappen aan te passen in het "properties" panel. Het is niet nodig aanpassingen te bevestigen. Deze worden steeds opgeslagen.

Personas
Het aanmaken van een persona is mogelijk door te klikken op de knop [+] in het persona-paneel. Wanneer men vervolgens het aangemaakte persona selecteert, kan men diens eigenschappen aanpassen dmv de invoervelden uiterst rechts. Een persona kan op dezelfde manier verwijderd worden als een scène.
Personas kunnen gekoppeld worden aan scènes. Een scène beeldt namelijk vaak een situatie af waarin één of meerdere personas voorkomen. Deze plaats(en) kunnen gekoppeld worden aan de beschikbare personas: door met de rechtse muisknop een persoon in een bepaalde scène te selecteren (slepen), krijgt men een lijstje met personas waaruit men een keuze kan maken.

User Stories
User Stories kunnen aangemaakt worden door een User Story in te geven in het tekstvak en vervolgens te klikken op de knop [+]. Het verwijderen van een User Story is mogelijk door deze te selecteren (niet aanvinken) en op de knop [-] te klikken.
Wanneer men een scène selecteert en vervolgens een User Story aanvinkt, wordt die User Story gekoppeld aan de scène. Het is ook mogelijk om een User Story aan te duiden als voltooid door deze te selecteren en vervolgens te klikken op [OK].

Annotaties

Annotaties kunnen toegevoegd worden op een onderdeel van een scène. Dit gebeurt op dezelfde manier als het koppelen van een persona aan een scène, maar alvorens dient men de optie “Annotate” te activeren.

Usability test (de files die nodig zijn, zijn te vinden in de map “StoryDraw” op de Desktop)

1	Start StoryDraw op.
2	Laad het scenario in uit de file “scenarioTest.txt”.
3	Maak volgende 2 personas aan: **Persona 1** • Naam: William • Leeftijd: 37 • Functie: Researcher • Afbeelding: “William.jpg” • Rol: Analist • William is een primair persona **Persona 2** • Naam: David • Leeftijd 22 • Functie: Jobstudent • Afbeelding: “David.jpg” • David is een secundair persona

4	Maak een eerste scène aan, gekoppeld aan volgend fragment uit het scenario: *“David zit op het vliegtuig naar Griekenland en werkt op zijn laptop aan een voorbeschouwing van de komende rally.”* Sleep de gecreëerde scène naar de gewenste positie op het storyboard en selecteer deze. Stel de afbeelding van de scène in op “scene1.jpg”.
5	Zet de optie “Auto arrows” aan en selecteer vervolgens volgend fragment uit het scenario: *“De lijst met deelnemers vindt hij op de officiële website van het WRC.”* Maak ook voor dit fragment een scène aan. Stel de afbeelding in op “scene2.png”.
6	Schakel de optie “Auto arrows” weer uit, en selecteer volgend fragment uit het scenario: *“William is ook aanwezig op het rallyevenement en trekt van proef tot proef om de rally op de voet te volgen en poogt de meest geweldige foto’s te maken.”* Maak voor deze selectie een scène aan en stel “scene3.jpg” in als afbeelding.
7	Maak ook voor volgende selectie een scène aan en stel de afbeelding in op “scene4.png”: *“Na elke klassementsproef (KP) publiceert hij zijn foto’s op iCon door middel van zijn PDA, welke een snelle verbinding heeft met de iCon-server.”* Verbind nu manueel de twee laatst aangemaakte scènes met een pijl van scène 3 naar scène 4. Verbind ook op dezelfde manier scène 2 met scène 3. Er wordt nu een pijl toegevoegd van scène 3 naar scène 4. Merk op dat wanneer de scènes herschikt worden (door te slepen), de pijlen automatisch geschikt worden.
8	Voeg volgende User Stories toe: • Het moet mogelijk zijn online verslagen te typen • Toevoegen van bijlagen (PDF) moet mogelijk zijn

9	Associeer de eerste User Story aan scène 1, de tweede aan scène 2.
10	We gaan verder met het aanduiden van de personas in de scènes. Selecteer hiertoe het gezicht van de persoon in scène 1. Dit persoon stelt het persona “John” voor. Koppel vervolgens de selectie aan dit persona.
11	Activeer de optie “Annotate” en markeer vervolgens een gedeelte van de website in scène 2 en voeg volgende annotatie toe: *“De WRC website laat het toe PDF bestanden te creëren van de deelnemerslijsten”*

12	Bekijk nu de associaties binnen het storyboard door middel van volgende mogelijkheden:
	
	• Hover met de muis over een persona:
	  • alle plaatsen waar deze persona voorkomt worden aangeduid in het storyboard.
	
	• Hover met de muis over een scène uit het storyboard:
	  • De personas betrokken in die scène worden gemarkeerd.
	  • De annotaties die horen bij de scène worden gemarkeerd, wanneer er over deze annotaties zweeft met de aanwijzer, wordt de bijbehorende notitie getoond op de Post It Note.
	
	• Selecteer een scène:
	  • Het fragment uit het scenario waar deze scène bij hoort wordt geel gemarkeerd.
	  • De User Stories horende bij deze scène worden aangeduid.

Questionnaire 1

Algemene vragen over de gebruiksvriendelijkheid van StoryDraw

1. Het design van de tool was goed, de layout was overzichtelijk.

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

2. Het aanmaken van nieuwe scènes is eenvoudig (a.d.h.v. selectie uit scenario)

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

3. “Auto arrows” is een handige en nuttige functie

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

4. Scènes manueel verbinden gaat vlot en intuïtief

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

5. De eigenschappen van een scène of persona aanpassen gaat snel

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

6. Het koppelen van personas aan scènes en het toevoegen van annotaties is eenvoudig/ snel

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord

D. Akkoord
E. Volledig akkoord

7. Het is nuttig dat de schikking van de scènes manueel aangepast kan worden

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Geen mening
   D. Akkoord
   E. Volledig akkoord

Questionnaire 2

Vragen voor personen zonder technische achtergrond (designers, grafici, human experts, e.d.)

1. Wat is uw achtergrond? (slechts 1 mogelijkheid aanduiden)

   A. Ik ben een grafisch designer
   B. Ik ben een UI designer
   C. Ik ben een Human Expert

   D. Andere: .....

2. Tijdens mijn activiteiten maak ik gebruik van volgende artefacten (meerdere mogelijk):

   A. User Stories
   B. Use Cases
   C. Scenario's
   D. Personas
   E. Storyboards
   F. (Paper) mock-ups
   G. Index Cards/Post It Notes
   H. Taakmodellen

3. Ik maak gebruik van een digitaal medium/tool voor het beheren van artefacten

   A. Ja
   B. Neen

      Zo ja: welke artefacten beheerd u op welke manier? Welke beperkingen zijn hieraan verbonden?

---

4. Wanneer ik de resultaten van mijn werk overdraag naar de ontwikkelaars, zijn er soms moeilijkheden bij de correcte interpretatie van deze data.

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

5. Tijdens mijn dagelijkse werkzaamheden is het niet voor de hand liggend om data (bijvoorbeeld mock-ups, onderzoeksdata over eindgebruikers en hun taken, requirements, ...) te beheren en uit te wisselen met mijn teamleden

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

6. Wanneer ik data ontvang van mijn teamleden (bijvoorbeeld gebruikersdata) is het soms moeilijk om deze data duidelijk te begrijpen en om snel feedback te geven

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

7. Een visueel “schema” van de workflow van de eindgebruikers, zou me helpen bij het begrijpen van de onderzoeksdata afkomstig uit het gebruikers- en taakonderzoek

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

8. Duidt aan voor elk onderstaande artefacten hoe belangrijk deze zijn voor uw dagelijkse werkzaamheden

   Scenario’s
   A. Niet in gebruik
   B. Nutteloos
   C. Noch akkoord, noch niet akkoord
   D. Belangrijk
   E. Zeer belangrijk

   Personas
   A. Niet in gebruik
   B. Nutteloos

C. Noch akkoord, noch niet akkoord
D. Belangrijk
E. Zeer belangrijk

User Stories
A. Niet in gebruik
B. Nutteloos
C. Noch akkoord, noch niet akkoord
D. Belangrijk
E. Zeer belangrijk

Annotaties
A. Niet in gebruik
B. Nutteloos
C. Noch akkoord, noch niet akkoord
D. Belangrijk
E. Zeer belangrijk

Storyboards
A. Niet in gebruik
B. Nutteloos
C. Noch akkoord, noch niet akkoord
D. Belangrijk
E. Zeer belangrijk

9. StoryDraw kan duidelijke een meerwaarde bieden bovenop het gebruik van klassieke artefacten zoals paper mock-ups, visuele schema's op papier, ...

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

Questionnaire 3

Vragen voor personen met een technische achtergrond (personen met kennis van Agile methodes/UCD technieken, developers, e.d.)

1. Een digitale tool kan de traditionele manier van werken (mondelinge communicatie, artefacten op papier, e.d.) volledig vervangen

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

2. Een digitale tool zoals StoryDraw is geschikt voor communicatie naar de klant/ eindgebruikers toe

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

3. De leercurve van een digitale tool is een obstakel voor het gebruik ervan binnen een ontwikkelproces

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

4. De voordelen van een digitale tool binnen een ontwikkelproces wegen veelal niet op tegen de tijd nodig voor het invoeren van de data in de tool

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

5. De overgangen tussen de fases binnen een ontwikkelproces (analyse -> design -> ontwikkeling) zijn voldoende ondersteund in StoryDraw

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

6. StoryDraw kan teamleden voldoende ondersteunen bij het begrijpen van elkaars werk, ondanks hun verschillende achtergrond

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord
    D. Akkoord
    E. Volledig akkoord

7. Het gebruik van StoryDraw in combinatie met een SmartBoard tijdens meetings, vergaderingen, brainstorming, Daily Standup Meetings, e.d. biedt meerwaarde t.o.v. het gebruik van een klassiek whiteboard of flipchart

    A. Volledig niet akkoord
    B. Niet akkoord
    C. Noch akkoord, noch niet akkoord

D. Akkoord
E. Volledig akkoord

8. StoryDraw kan enkel meerwaarde bieden bij het gebruik ervan binnen grote teams of gedelocaliseerde teams

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

9. Het is voor developers eenvoudig om de data uit de analyse- en designfase te gebruiken als invoer voor hun activiteiten wanneer zij deze raadplegen via StoryDraw (t.o.v. de klassieke situatie waarbij zij een reeks individuele artefacten aangereikt krijgen)

   A. Volledig niet akkoord
   B. Niet akkoord
   C. Noch akkoord, noch niet akkoord
   D. Akkoord
   E. Volledig akkoord

Op- en/of aanmerkingen

.....

.....

.....

.....

.....

.....

.....

Bedankt!

Bibliografie

(1) Statistics over IT Failure Rate - IT Cortex, http://www.it-cortex.com/Stat_Failure_Rate.htm, Visited June 2008

(2) IEEE Spectrum: Why Software Fails, http://www.spectrum.ieee.org/sep05/1685, Visited: June 2008

(3) Royce, W. 1987 Managing the Development of Large Software Systems, ICSE '87: Proceedings of the 9th international conference on Software Engineering (1987), pp. 328-338

(4) Larman, C., Basili, V. 2003 Iterative and Incremental Development: A Brief History, IEEE Computer Society, pp. 47-56

(5) Parnas, D., Clements, P. 1986 A Rational Design Process: How and Why to Fake It, IEEE Transactions on Software Engineering , Vol. SE-12 , Nr. 2

(6) Mills, H. 1976 Software Development, pp. 265-273

(7) Beedle et al. 2001 Principles behind the Agile Manifesto, http://agilemanifesto.org/, Visited: May 2008

(8) Beedle et al. 2001 Manifesto for Agile Software Development, http://agilemanifesto.org/, Visited: April 2008

(9) Gall, J. 1986 Systemantics: How Systems Really Work and How They Fail, 2nd edition, The General Systemantics Press, pp. 65

(10) Cockburn, A. 2008 Using Both Incremental and Iterative Development, Software Engineering Technology, pp. 27-30

(11) Spence, I., Bittner, K. 2005 What is iterative development?, Part 1, 2 & 3, IBM

(12) Schwaber, K. 2006 Agile Software Development with Scrum - Scrum FAQ, version 1, Conchango

(13) Larman, C. 2003 Agile and Iterative Development, A Manager's Guide, Addison-Wesley Professional

(14) Luyten, K. 2008 Geavanceerde Software Engineering, Cursustekst

(15) Iprofs - Timeboxing, http://www.iprofs.nl/article/Timeboxing.pdf, Visited: May 2008

(16) Jalote et al. 2003 Timeboxing: A Process Model for Iterative Software Development, Infosys Technologies Limited, Journal of Systems and Software v70, pp. 117-127

(17) MoSCoW Project prioritisation technique, http://www.coleyconsulting.co.uk/moscow.htm, Visited: May 2008

(18) May, E., Zimmer, B. 1996 The Evolutionary Development Model for Software, Hewlett Packard Journal

(19) IBM Rational Unified Process, http://www-306.ibm.com/software/awdtoolslivepage.apple.com/rup/, Visited: May 2008

(20) Ambrahamsson, P. 2002 Agile software development methods, Espoo, VTT publications 478

(21) Githens, G. 1998 Rolling Wave Project Planning

(22) Sy, D. 2007 Adapting Usability Investigations for Agile User-centered Design, Journal of Usability Studies, Vol. 2, Issue 3, pp. 112-132

(23) Constantine, L. 2001 Process Agility and Software Usability: Toward Lightweight Usage-Centered Design, Reprinted from Information Age, August/September 2002. Revised and expanded version of a column from The Management Forum, Software Development, vol. 9, no. 6

(24) Cockburn, A., Highsmith, J. 2001 Agile Software Development: The People Factor

(25) Ambler, S. Architecture Envisioning: An Agile Best Practice, http://www.agilemodeling.com/essays/initialArchitectureModeling.htm, Visited: August 2008

(26) McNeill, M. 2000 User Centred Design in Agile Application Development, ThoughtWorks Ltd.

(27) Ambler, S. User Stories, http://www.agilemodeling.com/artifacts/userStory.htm, Visited: August 2008

(28) Cockburn, A., Williams, L. 2000 The Cost and Benefits of Pair Programming, Humans and Technology, Technical Report

(29) Extreme Programming: A Gentle Introduction, http://www.extremeprogramming.org/, Visited: May 2008

(30) Scrum Alliance, http://www.scrumalliance.org/, Visited: May 2008

(31) Benefits of Agile Development, VersionOne, http://www.versionone.com/Resources/AgileBenefits.asp, Visited: August 2008

(32) Hudson, W. 2003 Adopting User-Centered Design Within an Agile Process: A Conversation, Cutter IT Journal, Vol. 16, part 10, pp. 5-12

(33) International Standards Organisation, ISO 13407, Human centred design processes for interactive systems, Geneva, Swiss, 1999

(34) Norman, D., Draper, S. 1986 User Centered System Design: New Perspectives on Human-computer Interaction, L. Erlbaum Associates Inc.

(35) Karat, J., Karat, C.M. 2003 The evolution of user-centered focus in the human-computer interaction field, IBM Systems Journal, vol. 42, no. 4

(36) Woletz, N., Zimmerman, D. 2005 Organizational Aspects of the Introduction of a User–centered Design Process, Proceedings of the 11th International Conference on Human-Computer Interaction, July 22-27, 2005, Las Vegas, Nevada USA

(37) Constantine, L. 1995 What Do Users Want? Engineering Usability into Software, Reprinted and revised (January 1999, June 2000) from Windows Tech Journal

(38) Raskin, J. 2000 The Human Interface: New Directions for Designing Interactive Systems, Addison Wesley

(39) McCracken, D., Wolfe, R. 2004 User-Centered Website Development, A Human-Computer Interaction Approach, Chapter 3, pp. 37-44

(40) Coninx, K. 2008 Gebruikersgerichte systeemontwikkeling, Cursustekst

(41) Crystal, A., Ellington, B. 2004 Task analysis and human-computer interaction: approaches, techniques, and levels of analysis, Proceedings of the Tenth Americas Conference on Information Systems, New York

(42) Gulliksen et al. 1999 User Centered Design in Practice - Problems and Possibilities, Proceedings of the CSCW ’98 conference

(43) Gulliksen et al. 2000 How to make User Centred Design Usable, CID-72, KTH, Stockholm, Sweden

(44) Lindström, H., Malmsten, M. 2008 User-centred design and the next generation OPAC – a perfect match?, Rethinking the Library, Wageningen, Netherlands

(45) Constantine, L. 2004 Beyond User-Centered Design and User Experience: Designing for User Performance, Cutter IT Journal, 17 (2)

(46) Mao et al. 2005 The State of User-Centered Design Practice, Commun. ACM, Vol. 48, No. 3. (March 2005), pp. 105-109.

(47) Gulliksen et al. 2003 Key Principles of User-Centred Systems Design, Desmarais M, Gulliksen J, Seffah A (eds) Human-centered software engineering: bridging HCI, usability and software engineering

(48) Maguire, M. 2001 Methods to support human-centred design, Int. J. Human-Computer Studies 55, pp. 587-634

(49) Dourish P. 2006 Responsibilities and Implications: Further Thoughts on Ethnography and Design, Proceedings of the 2007 conference on Designing for User eXperiences

(50) Beyer et al. 2004 An Agile User-Centered Method: Rapid Contextual Design

(51) Vredenburg et al. 2002 A Survey of User-Centered Design Practice, Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, April 20-25, 2002, Minneapolis, Minnesota, USA

(52) Kujala, S., Kauppinen, M. 2004 Identifying and Selecting Users for User-Centered Design, Nordic Conference on Human-Computer Interaction; Vol. 82; pp. 297 - 303

(53) Nebe et al. 2006 Integrating User Centered Design in a Product Development Lifecycle Process: A Case Study, International Conference on Software Engineering Research and Practice (SERP 06) 2006, Las Vegas, USA (NV)

(54) Hermann et al. 2000 Semistructured models are surprisingly useful for user-centered design, Designing Cooperative Systems, Proceedings of Coop 2000, pp. 159-174

(55) Kujala et al. 2001 Bridging the Gap between User Needs and User Requirements

(56) Abras et al. 2004 User-Centered Design, In Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications.

(57) Dayton et al. 1993 Skills needed by User-Centered Design Practitioners in real software development environments: Report on the CHI ’92 workshop

(58) Greene et al. 2003 Iterative development in the field, IBM Systems Journal, volume 42, Issue 4 (October 2003) pp. 594-612

(59) Raymaekers, C. 2008 Evaluatie van user interfaces, Cursustekst

(60) Rieman et al. 1995 Usability Evaluation with the Cognitive Walkthrough, Conference companion on Human factors in computing systems, Denver, Colorado, United States, pp. 387-388

(61) Nielsen, J. 1992. Finding usability problems through heuristic evaluation. Proc. ACM CHI'92 (Monterey, CA, 3-7 May), pp. 373-380

(62) Berling, T., Runeson P. 2003 Evaluation of a perspective-based review method applied in an industrial setting, EE Proceedings Volume 150, Issue 3, 24 June 2003, pp. 177-184

(63) Stockton et al. 2002 Using Focus Groups, version 2.0

(64) Burke, J., Kirk, A. Ethnographic methods, http://www.otal.umd.edu/hci-rm/ethno.html, Visited: June 2008

(65) Katz, G. 2006 The truth about ethnography

(66) Beyer, H., Holtzblatt, K. 1998 Contextual Design: A Customer-Centered Approach to Systems Designs

(67) Embrey, D. 2000 Task Analysis Techniques, Human Reliability Associates Ltd.

(68) Stuart, J., Penn, R. 2004 TaskArchitect: Taking the Work out of Task Analysis, ACM International Conference Proceeding Series; Vol. 86, Proceedings of the 3rd annual conference on Task models and diagrams, Prague, Czech Republic, session: Task analysis and diagrams for task models, pp. 145-154

(69) Paterno, F. 1997 ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, IFIP Conference Proceedings; Vol. 96, Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction, pp. 362-369

(70) Coninx, K. 2008 Multidisciplinaire Benadering van Human Factors, Cursustekst

(71) Hochstein, L. 2002 GOMS http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/printer/goms.html, Visited: July 2008

(72) Dix et al. 2003 Human-Computer Interaction, Section 15.6, pp. 532-533

(73) Pruitt, J., Grudin, J. 2003 Personas: Practice and Theory, Designing For User Experiences, Proceedings of the 2003 conference on Designing for user experiences, San Francisco, California, session: Informing DUX, pp. 1-15

(74) Bodker, S. 1999 Scenarios in User-Centred Design - setting the stage for reflection and action, Interacting with Computers 13, pp. 61-75

(75) Redmond-Pyle, D. 1995 Graphical User Interface Design and Evaluation (GUIDE): A Practical Process, Chapter 11, pp. 216-246

(76) Bernsen et al. 1994 Wizard of Oz prototyping: How and when? CCI Working Papers in Cognitive Science and HCI, WPCS-94-1. Centre for Cognitive Science, Roskilde University

(77) Zhang et al. 1999 Perspective-based Usability Inspection: An Empirical Validation of Efficacy, Empirical Software Engineering, Volume 4, Number 1, pp. 43-69

(78) TRUMP - The benefits of User Centered Design, http://www.usabilitynet.org/trump/methods/integration/benefits.htm, Visited: June 2008

(79) Soderston, C., Thyra, R. 2002 The Case for User-Centered Design

(80) Bygstad et al. 2008 Software development methods and usability: Perspectives from a survey in the software industry in Norway, Interacting with Computers 20, pp. 375–385

(81) Nelson, N. 2002 Extreme Programming vs. Interaction Design, http://www.fawcette.com/interviews/beck_cooper/, Visited: August 2008 (Internet Archive)

(82) Cecil, R. 2006 Clash of the Titans: Agile and UCD, http://uxmatters.com/MT/archives/000153.php, Visited: August 2008

(83) Patton, J. 2008 Twelve emerging best practices for adding UX work to Agile development, http://agileproductdesign.com/blog/emerging_best_agile_ux_practice.html, Visited: August 2008

(84) Miller, L. 2005 Case Study of Customer Input For a Successful Product, adc pp. 225-234, Agile Development Conference (ADC'05)

(85) Nielsen, J. 2000 Why You Only Need to Test With 5 Users, Jakob Nielsen's Alertbox, http://www.useit.com/alertbox/20000319.html, Visited: August 2008

(86) Scrum in five minutes, http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf, Visited: May 2008

(87) Meskens et al. 2008 Gummy for Multi-Platform User Interface Designs: Shape me, Multiply me, Fix me, Use me, Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy, session: Interaction environment design, pp. 233-240

(88) Film Education - About Storyboards, http://www.filmeducation.org/secondary/StudyGuides/storyboard.pdf

(89) Jurassic Park: Storyboard, http://jurassicpark.sk/jurassic_park/jp_storyboard_en.html, Visited: December 2008

(90) Memmel et al. 2007 Agile Human-Centered Software Engineering, British Computer Society Conference on Human-Computer Interaction, Proceedings of the 21st British CHI Group Annual Conference on HCI 2007: People and Computers XXI: HCI...but not as we know it - Volume 1, University of Lancaster, United Kingdom, session: From theory to technique, pp. 167-175

(91) Ambler, S. Agile Modeling: Effective Practices for Modeling and Documentation, http://www.agilemodeling.com

(92) CC Pace, 2001 Usability and User Interface Design in XP, http://www.ccpace.com/Resources/documents/UsabilityinXP.pdf, Visited: April 2009

(93) Patton, J. 2002 Hitting the Target: Adding Interaction Design to Agile Software Development, Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA 2002 Practitioners Reports