**REQ ANAL**

1.introduction (door lezen) / 2. Defenitie

**2. Wat zijn requirements?**

**Definitie** = Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a systemproperty or attribute. They may be a constraint on the development process of the system.

* Veel projecten zullen falen omdat de vereisten van de klant niet volledig werden bevraagd, begrepen of gedocumenteerd
* Een project heeft verschillende stakeholders (zie later) en in de requirements komen de belangen van al deze stakeholders vaak samen
* Belangrijk is dat we leren hoe we deze vereisten kunnen bepalen, valideren en documenteren

**Business Requirement** = Strategisch doel van het bedrijf dat de software implementeert*.( We hebben een webshop nodig om onze verkoop te kunnen stimuleren)* = driver

**Business Rule** = Een policy, richtlijn of standaard die een bepaalde voorwaarde stelt aan de software. Is geen requirement op zich, maar leidt tot één of meerdere requirements.(*vb. elke bestel bon ondertekend door manager).*

**Constraint =** Een beperking die wordt opgelegd waardoor de keuzes die we kunnen maken beperkter worden*.(vb. In ons bedrijf is enkel JAVA-kennis aanwezig).*

**External interface requirement** = Beschrijving van een link tussen verschillende systemen en/of hardware. *(Vb. “Betalingen online gebeuren via PayPal)*

**Feature** = Een set van functionaliteiten die een logisch geheel vormen en een meerwaarde vormen voor de klant. *(Vb. Er moet een productcatalogus ter beschikking zijn)*

**Functionele Requirement =** Een beschrijving van hoe het systeem zich zal gedragen onder bepaalde omstandigheden. *(vb. Als een product niet op voorraad is, moet de verwachte levertermijn getoond worden)*

**Nonfunctional Requirement** = Een beschrijving van een eigenschap van het systeem. *(vb. De webshop moet ook kunnen getoond worden op mobiele toestellen).*

**System requirement** = De systeemvereisten die nodig zijn om een softwareproduct te kunnen maken*.(vb. Het inloggen in de webshop vereist een tweestapsverificatie)*

**User requirement** = Taken die een user moet kunnen uitvoeren. *(vb. en verkoper moet alle bestelbonnen kunnen bekijken)*

**Belangrijkste Types**

User Requirements

* **Taken die de gebruikers moeten kunnen uitvoeren**
* Voegen waarde toe
* Use cases/User stories
* “Als een klant, wil ik de verschillende producten kunnen bekijken in een catalogus en meer informatie bekomen per individueel product

Functional Requirements

* **Beschrijft wat developers zullen maken om ervoor te zorgen dat de user requirements kunnen worden uitgevoerd en de business requirements worden ingelost.**
* “Wanneer de klant op een product in de catalogus klikt, zal er een foto, beschrijving, voorraad en prijs worden getoond op een nieuwe pagina.”
* Beschreven in de functionele specificatie en gebruikt voor development, testing, PM,… (SRS - Software Requirements Specification)

Nonfunctional Requirements

* Beschrijft niet wat het systeem moet doen, maar hoe het systeem het doet
* Security, performance, usability, design,..
* Ook de omgeving waarin het systeem draait kunnen aanzien worden als nonfunctional requirements
  + Portability, compatibility, platform, framework,...

Feature

* **Is een verzameling van logisch gerelateerde functies in een systeem en wordt beschreven door meerdere functionele vereisten**
* Feature = online catalogus

1. Een gebruiker kan de ganse catalogus bekijken
2. Een gebruiker kan zoeken binnen de catalogus
3. Een gebruiker kan gebruik maken van categorieën om te browsen
4. Een gebruiker kan klikken op een specifiek product om meer details te krijgen van dat product
5. Een gebruiker kan rechtstreeks vanuit de catalogus een bestelling plaatsen
6. ….

Deliverables

* Requirements moeten gedocumenteerd worden in 3 deliverables
* Scope statement
* User requirements document
* Software requirements specification -> Deze verschillende onderdelen zien we terugkomen in het analysedocument
* Ook requirements moeten iteratief bekeken worden
* Bij een nieuwe functionele/user vraag moet steeds bekeken worden of het onderdeel is van de scope/business need

Change requests

**In lopende projecten kunnen we niet zo maar requirements gaan toevoegen**

* Risico voor kost, tijd en kwaliteit
* Change request procedure
* Vraag wordt voorgelegd aan de change advisory board
* Soms krijgt PM beperkte mogelijkheid om kleine changes goed te keuren
* Afweging maken tussen klantvriendelijkheid en project scope

🡪Opletten voor Scope Creep

product requirements VS project requirements

* Product requirements geven aan hoe een stuk software er zal moeten uitzien, hoe het werkt,…
* Project requirements geven aan wat een project nodig heeft om te slagen
  + Training van gebruikers
  + Laptops
  + paswoorden
  + Documentatie
  + Licenties, ….
* In dit vak spreken we enkel over product requirements, project requirements zien we volgend jaar in Project Management

Requirements ontwikkeling

* Elicitation
  + Identificeren van de requirements via interviews, workshops, prototyping,…
* Analysis
  + Verder uitwerken van elke requirement
* Specification
  + Documenteren van de requirements naar geschreven requirements en diagrammen
* Validation
  + Reviewen van de specificaties om eventuele fouten er uit te halen • Opstellen van acceptance tests è Meerdere iteraties kunnen nodig zijn om het complete plaatje te begrijpen

Requirements management

* Vastleggen van de requirements baseline: het ijkpunt van waar de development zal starten
* Change request procedures
* Afchecken dat het project plan in lijn blijft met de afgesproken requirements
* Bepalen van de impact van veranderingen en de kost ervan in het project
* Definiëren van afhankelijkheden tussen requirements
* Opvolgen van de status van de verschillende requirements

Elk project heeft requirements…

• Kost van verandering wordt groter naarmate het project vordert

• Maar: het is niet altijd mogelijk om alle requirements vooraf vast te leggen

🡪 Iteratieve aanpak – Agile development

…maar deze worden niet altijd goed gedocumenteerd

• Onvoldoende betrokkenheid van de gebruikers

* Assumpties dat de developers weten wat de gebruikers willen/nodig hebben
* Gebruikers weten het vaak zelf niet
* Moeilijk om de juiste gebruikers te vinden

• Slechte planning

* Wanneer requirements onvoldoende duidelijk zijn, zal de planning vaak niet correct zijn

• Scope creep

* Het organisch uitbreiden van de scope omdat de klant steeds kleine extra’s vraagt
* Scope wijzigt altijd doorheen het project maar de PM moet impact ervan bepalen en managen

• Onduidelijke requirements

* Requirements moeten duidelijk én éénduidig zijn
* Onduidelijke requirements leiden vaak tot verschillende verwachtingen bij verschillende stakeholders

• Gold plating

* Developers implementeren soms functionaliteiten die niet gevraagd werden maar waarvan ze denken dat de gebruikers ze nodig hebben
* Leidt tot tijdverlies

• Vergeten stakeholders

* Soms worden bepaalde groepen gebruikers niet betrokken omdat ze op het eerste zicht weinig in contact komen met het product

**3. Processen & rollen**

**Requirements Elicitation**

**1. Definieer product visie en project scope**

* Beschrijf de business requirements
* Zorg ervoor dat alle stakeholders een gelijkaardig begrip hebben van wat het eindresultaat is van het project
* Scope bepaalt wat er zal opgeleverd worden en wat niet
* De visie blijft stabiel, de scope kan wijzigen (zeker bij een iteratieve projectmethodologie)

**2. Identificeer gebruikersgroepen en hun eigenschappen**

* Identificeer de verschillende gebruikers (ook diegene die minder voor de hand liggen)
* Beschrijf hun rol op basis van taken, gedragingen, locatie,…
* Maak personas

**3. Kies een key user voor elke groep**

* Identificeer welke gebruiker het beste op de hoogte is en op die manier de ganse groep kan vertegenwoordigen
* Het is makkelijker met één persoon te praten dan met een ganse groep

**4. Hou focusgroepen met typische gebruikers**

* Breng een groep representatieve gebruikers samen die de vorige software gebruikten of de taken uitvoeren die in de nieuwe software zullen worden uitgevoerd
* Deze gebruikers kunnen zowel functionele als kwalitatieve input geven

**5. Identificeer User Requirements**

* Welke taken moeten ze uitvoeren met de nieuwe software?
* Welke meerwaarde zoeken ze?
* User stories/Use cases/Scenarios

**6. Identificeer system events en responses**

* Identificeer events die zich voor kunnen doen en hoe de software hier op gaat reageren: signal (externe input) – temporal (geplande events) – business

7. Gebruik verschillende technieken om requirements te bepalen

* Interviews: Afzonderlijke interviews om persoonlijke requirements te bepalen
* Workshops: In groep om requirements documents op te stellen
* Job monitoring: Observatie van key users
* Questionnaires: Bevragen van grote groepen
* Document analyse: Bekijken van bestaande documentatie
* Problem reports: Huidige problemen kunnen inzicht geven
* Bestaande requirements: Bestaande documenten gebruiken als die er zijn

**Requirements Analysis**

**1. Modelleer de applicatie-omgeving**

* Beschrijf hoe de nieuwe applicatie in de omgeving past
* Context diagram/ecosystem

**2. Creëer use interface en/of technisch prototype**

* Wanneer het een totaal nieuwe applicatie betreft, kan een prototype helpen de requirements beter te bepalen

**3. Analyseer de haalbaarheid van de requirements**

* Kosten/baten moeten worden afgewogen
* Een functionaliteit die voor 1 user veel waarde oplevert, is niet altijd een must
* Wanneer een requirement een grote impact heeft op andere applicaties, moet dit afgewogen worden
* Technische haalbaarheid

**4. Prioteriseer de requirements**

* De belangrijkste requirements (noodzakelijk voor andere requirements of diegene met de hoogste toegevoegde waarde) moeten eerst gedaan worden
* Prioriteiten kunnen wijzigen!

**5. Maak een data dictionary**

* Data items en structuren moeten door iedereen gelijk opgesteld worden

**6. Modelleer de requirements**

* Modellen en schema’s visualiseren de geschreven requirements
* Geven vaak nieuwe inzichten
* Data flow diagrams, entity-relationship diagrams,…

**7. Analyseer de interfaces**

* De meeste applicaties staan niet op zichzelf
* Communicatie met user management systems, andere applicaties, hardware,….

**8. Wijs requirements toe aan subsystemen**

* Bij zeer complexe producten, kan het zinvol zijn de applicatie op te delen en de requirements toe te wijzen aan een subsysteem om de complexiteit te verminderen

**Requirements Specification**

1. Zorg voor standaard templates

* Deliverables moeten opgeleverd worden in een template

2. Indentificeer de oorsprong van requirements

* Het is belangrijk te weten wie de oorspronkelijke requirement heeft doorgegeven zodat de juiste persoon kan worden gecontacteerd in geval van vragen

3. Label elke requirement op unieke manier

* Zorg ervoor dat requirements makkelijk terug te vinden zijn

4. Documenteer de business rules

* Business rules overstijgen meestal het project/de software
* Documenteer deze dus apart

5. Specifieer non-functional requirements

* Kwaliteitsvereisten zoals performance, betrouwbaarheid, aanpasbaarheid, veiligheid,…
* Gebruikers denken hier zelden over na maar verwachte dit wel

**Requirements Validation**

**1. Review de requirements**

* Laat iemand anders de requirements en bijbehorende documentatie nalezen

**2. Test de requirements**

* Schrijf test scenario's
* Overloop deze met de gebruikers om zeker te zijn dat de verwachtingen correct zijn weergegeven

**3. Definieer acceptance criteria**

* Vraag de users wanneer een oplossing voldoet en in productie kan gezet worden

**4. Simuleer de requirements**

* Maak mock-ups of gebruik prototypes • Niet noodzakelijk maar kan een goede aanvulling zijn bij nieuwe software of processen

**Requirements opstellen: de rollen**

Stakeholders

* Stakeholders zijn alle personen/groepen die een impact kunnen hebben op het project of een impact kunnen ondervinden door het project
* We bespreken de belangrijkste voor deze fase
* Extra aandacht voor de business analyst/functionele analyst

Business analyst/Functional consultant

* Ondersteunt en adviseert de business rond de optimalisatie van de processen
* Inventariseren van de wensen, vereisten en bestaande problemen (AS-IS)
* Definiëren van het te bereiken streefdoel voor alle betrokken personen en processen (TO-BE)
* Werkt aan de harmonisatie van processen en procedures
* Schrijft de business case mee uit
* Betrokken bij het ganse proces van requirements ontwikkeling

Functionele manager

* Leidt een afdeling binnen een bedrijf waar het project wordt uitgevoerd
* Balans tussen dagelijks werk en project werk
* Input voor de scope en de vereisten
* Verantwoordelijk voor de bedrijfsprocessen die aan de basis liggen van de oplossing
* Zetelt vaak in de stuurgroep van het project

Proceseigenaar

* De expert aangaande het business proces dat we proberen te verbeteren met het project
* Levert cruciale input tijdens de concept- en ontwerpfase
* Identificeert de mogelijke wijzigingen en verbeteringen aan het proces
* Zorgt voor een aanvaarding van de wijzigingen in de organisatie

Sleutelgebruiker

* De sleutelgebruiker (of keyuser of kerngebruiker) is een ervaren gebruiker van een proces of subproces
* Leveren de vereisten in de concept- en ontwerpfase
* Spelen een cruciale rol bij de oplevering omdat ze eerste aanspreekpunt zijn voor collega’s
* Staan bij go-live in voor de verificatie van de applicatie
* Voeren de eerste (functionele) testen uit
* Zijn het klankbord op de werkvloer en dragen boodschappen van het project uit naar de collega’s

Sponsor

* Draagt de eindverantwoordelijkheid over het project
* Stelt de middelen ter beschikking om het project te kunnen uitvoeren
* Initieert de projectvraag
* Directe communicatie tussen de project manager en de sponsor
* Zetelt meestal in de stuurgroep van het project

Stuurgroep

* Centraal rapporterings- en beslissingsorgaan van het project
* Wordt in principe periodiek georganiseerd (maandelijks bvb) maar er is flexibiliteit nodig om de stuurgroep samen te roepen wanneer nodig
* Rapporteren van de status
* Nemen van belangrijke beslissingen ivm het project

Developer

* Schrijven de applicatie
* Staan in voor de technische oplevering van het product
* Dragen vooral bij tijdens de implementatiefase maar kunnen ook in het voortraject betrokken worden
* Schrijven de technische specificatie
* Voeren de (technische) testen uit en lossen eventuele bugs op
* Staan mee in voor de support na go-live

Architect

* Bewaakt de “big picture” inzake IT-architectuur
* Zorgt ervoor dat nieuwe keuzes passen in de bestaande architectuur
* Zorgt dat de communicatie tussen verschillende systemen op een uniforme manier gebeurd
* Is vooral tijdens de eerste fases betrokken bij het project in een adviesro

Tester

* Voert de testen uit
* Maakt in principe niet zelf de testscenario’s om objectiviteit te garanderen
* Rapporteert aan de testmanager
* Vooral aanwezig in de testfase
* Testers kunnen zowel interne als externe medewerkers zijn

Change Advisory Board (CAB)

* Wordt samengeroepen om wijzigingen te bespreken en goed te keuren
* Zorgt er dus voor dat scope creep onder controle wordt gehouden
* Frequentie van samenkomen is niet vast maar zal bepaald worden op basis van de vragen
* Bestaat uit de sponsor, architecten, functionele managers, team leads,…
* De projectmanager legt de vragen voor en verantwoordt ze

**4. Scope**

Business Requirements

* Een strategische nood die leidt tot één of meerdere projecten die een oplossing bieden voor deze nood
* Moet duidelijk zijn voor de functionele en niet-functionele requirements worden opgesteld
* Zorgt voor een kader waarbinnen beslissingen kunnen worden genomen en veranderingen kunnen worden afgestemd
* Context en doelen van het project
* Meestal gedefinieerd door sponsor, CEO, directie, management

**1. Background**

* Beschrijf de context voor het project

**2. Business Opportunity**

* Beschrijf welk business probleem er wordt opgelost of welk proces er verbeterd wordt
* Business case kan hier worden toegevoegd
* Beschrijf welke technologieën, processen en resources er nodig zijn om een totaaloplossing te kunnen bieden

**3. Business Objectives**

* Beschrijf de doelen van het project op een SMART manier
* **S**pecific = make your goals specific and narrow for more
* Measureable = define evidence that proves your making progress
* Attainable = make you sure u can accomplish the goals in a certain timeframe
* Relevant = goals should align with your vvalues and long-term obj
* Time-based = set realistic end-dates

**4. Success metrics**

* Business objectives kunnen vaak pas afgemeten worden op het einde van het project
* Success metrics meten het verloop van het project en of het project on track is
* Belangrijk is de business problems te linken met business objectives

**5. Vision statement**

* Beschrijf het lange termijn doel en intentie van het product
* Balanceert de verschillende verwachtingen van de verschillende stakeholders
* For [target customer]
* Who [statement of the need or opportunity]
* The [product name]
* Is [product category]
* That [major capabilities, key benefit, compelling reason to buy or use]
* Unlike [primary competitive alternative, current system, current business process]
* Our product [statement of primary differentiation and advantages of new product]

**6. Business Risks**

* Beschrijf de risico’s die de business loopt door het project al dan niet uit te voeren
* Geen projectrisico’s

**7. Business Assumptions en Dependencies**

* Assumpties zijn dingen die we aannemen, maar niet voldoende informatie over hebben om het zeker te weten
* Deze moeten goed gedocumenteerd worden

Scope & Limitations

* De scope goed definiëren, zal scope creep voorkomen
* Scope kan gedefinieerd worden op verschillende manieren en hangt af van het niveau
* Op niveau van het scope statement is het enkel een lijst van grote functionaliteiten

**1. Major Features**

* Beschrijft grote features vanuit het standpunt van de gebruikers
* Deze features moeten waarde toevoegen

**2. Scope of initial release**

* Beschrijft welke features in de eerste release zullen zitten (must-haves)

**3. Scope of subsequent releases**

* Beschrijft welke features zullen opgenomen in latere cycli of releases (nice-tohaves)

**4. Limitations and exclusions**

* Beschrijft welke features uitdrukkelijk niet in scope zullen zijn

Business Context

**1. Stakeholder profiles**

* Stakeholders are the people, groups, or organizations that are actively involved in a project, are affected by its outcome, or are able to influence its outcome.
* Een stakeholder profile beschrijft:
  + Het voordeel dat een stakeholder haalt uit het product
  + De attitude van de stakeholder tov het product
  + Belangrijkste features voor de stakeholder
  + Bekende beperkingen waar rekening mee moet gehouden worden

**2. Project priorities**

* 5 factoren: cost, scope, time, quality, resources
  + Constraint: limiterende factor
  + Driver: belangrijkste factor
  + Degree of freedom: factor die enige vrijheid heeft
* Bepalen welke factor de belangrijkste is en welke beperkt zijn/vrijheden hebben
* Voobeeld: klant vraagt maand vroeger op te leveren omwille van een nakende fusie

**3. Deployment Considerations**

* Beschrijft de informatie en activiteiten nodig om het project succesvol uit te kunnen voeren
* User access, hardware, data migratie,

**5. Personas**

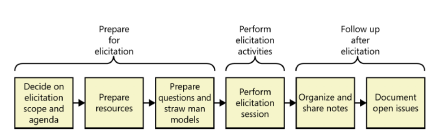
Theori + labo lezen

**6. Technieken**

Elicitation (uitlokken, achter halen)

* Het achterhalen van requirements bij users is meer dan enkel de requirements noteren
* Het is een proces van samenwerken en analyseren zodat requirements worden verzameld, ontdekt, achterhaald en gedefinieerd
* Door users te betrekken bij dit proces, zorg je voor betrokkenheid en buy-in
* De uitkomst van dit proces zorgt voor een gezamenlijk begrip van wat de nieuwe applicatie moet kunnen

Proces voor Elicitation



Technieken voor Elicitation

1. Interviews
2. Workshops
3. Focusgroepen
4. Observaties
5. Questionnaires
6. System Interface Analyse
7. User Interface Analyse
8. Document Analyse

1. intervieuw

• Kleine groepen, management, directie, key-user,…

• Tips

* Introduceer jezelf, de agenda, het doel van het interview
* Hou je aan het onderwerp
* Gebruik een vragenlijst en zo veel mogelijk draft documenten om het interview te structureren
* Maak suggesties en reik alternatieven aan
* Probeer steeds actief te luisteren (geef feedback en vat samen

2. workshops

* Gestructureerde meeting waar stakeholders en experts samen werken om finale deliverables te kunnen opleveren
* Wordt vaak gebruikt om een consensus te bereiken – vaak meerdere rollen tegelijk aanwezig
* Rol van facilitator: zorgt voor planning, deelnemers en stuurt de workshop naar de gewenste uitkomst
* De tips voor interviews gelden hier ook plus een aantal extra
  + Zorg voor een aantal basisregels
  + Zorg dat er iemand noteert, de tijd in de gaten houdt, scope wordt bewaard,…
  + Plan een agenda
  + Check elke requirement steeds met de business requirements
  + “Parkeer” belangrijke, niet-relevante items tijdelijk op een flipchart zodat ze niet vergeten worden
  + Timebox discussies om tijdverlies te vermijden
  + Beperk het aantal deelnemers tot de juiste deelnemers
  + Zorg ervoor dat iedereen gefocust blijft en dat alle deelnemers voldoende gehoord worden

3 focusgroepen

* Focusgroepen worden gebruikt om input te vragen van een grote groep gebruikers
* Kies de deelnemers goed
* Gebruikers met ervaring met een gelijkaardig product, vorige versie of proces
* Gebruikers met een gelijke achtergrond of net heel divers
* Balans vinden tussen “on topic” blijven zonder opinies te beïnvloeden
* Subjectieve feedback die verder moet geëvalueerd en geanalyseerd worden
* Meestal geen finale beslissingen

4.observaties

* Als je vraagt aan een gebruiker om zijn taken te beschrijven, is dat vaak onvolledig/onjuist
  + Sommige taken zijn zo vanzelfsprekend dat ze vergeten worden
  + Sommige taken zijn zeer complex
  + Sommige gebruikers zijn bang dat ze niet voldoende doen en overdrijven
* Gebruikers observeren kan hier een oplossing bieden
  + Kost veel tijd, dus niet voor alle gebruikers
  + Ter validatie van eerder verkregen informatie
  + Observaties kunnen stil of interactief zijn (eventueel opgenomen voor latere analyse en bespreking)

5.Questioinares

* Grote groepen bevragen
* Vrij goedkoop, geen fysieke grenzen
* De analyse van questionnaires kan vaak als input gebruikt worden voor andere technieken
* Stel goede vragen
  + Zorg ervoor dat alle antwoorden ter beschikking zijn
  + Vermijd vraagstelling waarbij een “correct” antwoord wordt gesuggereerd
  + Gebruik steeds dezelfde schalen
  + Gebruik gesloten vragen met antwoorden als je een statistische analyse wil doen
  + Test de questionnaire voor gebruik
  + Stel niet teveel vragen

6.System Interface Analyse

* Analyse van de systemen die verbinding zullen maken met het nieuwe systeem
  + Requirements over de uitwisseling van data
  + Context diagrams & Ecosystem maps

7.User Interface Analyse

* Analyse van bestaande systemen om reeds bestaande functionele requirements te achterhalen
* Op basis van screenshots, user manuals of toegang tot het bestaande systeem
* Eigen analyse kan basis zijn voor bevraging van gebruikers
* Ga er niet van uit dat een functionaliteit uit het oude systeem automatisch in het nieuwe systeem nodig is

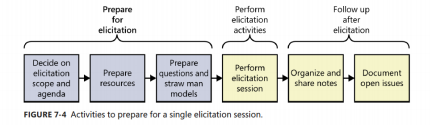
8.Document Analyse

* Analyseren van bestaande documentatie om requirements te achterhalen
* Bestaande requirements specificaties, business processes, user manuals,…
* Goede manier om zich in te werken in de context van het project
* Vaak een manier om requirements te achterhalen die de gebruikers niet kennen of te ontdekken dat de gebruikers het systeem anders gebruiken dan initieel voorzien

Plannen

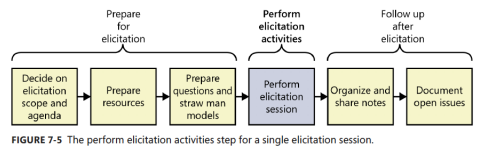
* Zorg voor een duidelijke planning zodat resources kunnen worden vrijgemaakt
  + Zet de doelen
  + Beschrijf de strategie en technieken
  + Maak inschattingen
  + Beschrijf documenten en systemen die nodig zijn
  + Beschrijf het gewenste resultaat en de deliverables
  + Identificeer de risico’s

Proces voor elecitation (uitlokken, achterhalen)



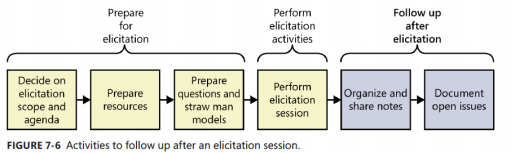
Voorbereiding

* Plan de scope en agenda voor de sessie
* Bereid de resources voor: zowel fysiek al menselijke resources
* Identificeer de verschillende stakeholders en leer hen kennen
* Bereid de vragen voor
  + Vragen moeten voldoende open zijn maar wel specifiek
  + Leef je in in de gebruiker (persona)
  + Stel ook vragen over uitzonderingen
* Bereid draft modellen en use cases voor om de discussie te voeden



Uitvoering

* Licht de stakeholders in over het elicitation proces: de technieken, de processen en de volgende stappen
* Neem goede notities
  + Zorg dat er iemand is die alles opschrijft
  + Aanwezigheden, beslissingen, to do’s, verantwoordelijken, open issues,…
* Gebruik de omgeving: muren om post-its op te plakken, white boards, stand-up sessies



Opvolging

* Organiseren en delen van de notities
  + Review notities na de sessie
  + Opletten met correcties nadien – vraag steeds validatie
* Documenteren van open issues
  + Issues die geparkeerd werden tijdens de meeting
  + Nieuwe issues die uit de notities kwamen
  + Oplijsten in een issue-tracking tool

Classificeren van requirements

* **Business Requirement**
  + Verhogen van marktaandeel in Vlaanderen met 10% binnen 12 maanden.
* **User Requirement** 
  + Ik moet een verzendlabel kunnen printen voor een pakket.
* Business Rules
  + Een nieuwe klant moet 30% voorschot betalen bij het ondertekenen van het contract.
* **Functionele Requirements** 
  + De gebruiker moet de productlijst alfabetisch kunnen sorteren.
* **Quality Attributes** 
  + De mobiele versie moet snel reageren op touch commando’s.
* **External Interface Requirements** 
  + De mobiele app moet de verificatie foto naar de bank sturen nadat ik een foto heb genomen van mijn identiteitskaart.
* **Constraints**
  + Files die ingediend worden mogen niet groter zijn dan 10MB.
* **Data Requirements** 
  + Een postcode heeft 4 numerieke karakters, de default is 0000

Wanneer ben je klaar?

* De users kunnen geen verdere use cases of user stories bedenken
* Nieuwe scenario’s leiden niet meer tot nieuwe functionaliteiten
* Users herhalen issues
* Nieuwe features of requirements zijn oud of scope of lage prioriteit
* Reviewers hebben nog weinig vragen

Aandachtspunten

* Balanceer de aanwezigheid van de stakeholders
* Zorg dat de scop e steeds goed gedefinieerd is
* Maak een onderscheid tussen requirements en design
* Probeer maximaal te onderzoeken zonder het proces onnodig te vertragen
  + kosten/baten afwegen
  + Belang van requirement weegt door

7. Use Case/ user stories

Introductie

* Essentieel onderdeel van software design is begrijpen wat de gebruikers zullen gaan doen met de software
* Productfocus vs userfocus
  + Productfocus start van de functionaliteiten in de hoop dat deze voldoen aan de verwachtingen van de gebruikers
  + Userfocus start van de gebruikers en hoe zij de software zullen gebruiken
* User requirements bevinden zich tussen business requirements en functionele requirements
* Twee zeer belangrijke technieken om deze user requirements te bepalen: use cases en user stories
* Use cases en user stories vragen de gebruiker wat hij wil doen ipv te vragen hoe hij het systeem wil gaan gebruiken
  + De analist zal dit moeten vertalen naar functionaliteiten
  + Testen worden opgesteld om te kijken of de functionaliteit goed was geïmplementeerd
  + Use cases en user stories werken goed voor “functionele applicaties” zoals websites/business applicaties/ERP-systemen
  + “Technische applicaties” zijn moeilijk te beschrijven omdat de userinteractie zeer beperkt is (data warehousing, batch processes,…)

Use Case

**Definitie:** Een use case beschrijft de opeenvolging van interacties tussen de applicatie en een externe actor waardoor de actor een zeker uitkomst bekomt

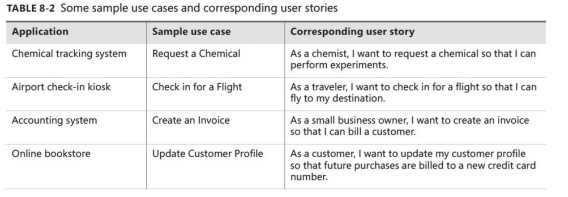
* Omschrijving steeds “Werkwoord” + “Onderwerp”



User story

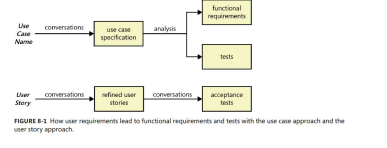
**Defintitie**: Korte, simpele beschrijving van een feature beschreven vanuit het perspectief van de persoon die de nieuwe functionaliteit nodig heeft, meestal een gebruiker of klant van een applicatie

* User stories worden meestal beschreven in de volgende format: Als: (klant) wil ik: (beschrijving van datgene dat ontwikkeld moet worden) zodat ik: (beschrijving van de reden waarom dat ontwikkeld moet worden)
* User stories geeft de reden weer achter een bepaalde vraag van de gebruiker
* User class komt hier terug in “type gebruiker”



**Use case VS User Story**

* Use cases en user stories lijken sterk op elkaar bij de start maar het vervolg qua uitwerking en deliverables is zeer verschillend

****

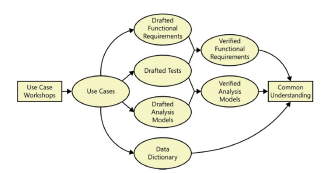
* **Use cases** worden meestal zeer nauwkeurig uitgewerkt via templates en diagrammen om op die manier functionele en niet-functionele requirements te kunnen bepalen
* **User stories** dienen eerder als placeholders om later op te nemen en verder te analyseren (Agile werken) en als basis om “Acceptance Criteria” te bepalen (Testing)
* Grotere user stories kunnen opgedeeld worden in kleinere user stories zodat ze binnen één epic kunnen worden afgewerkt
* User stories zijn simpel maar missen vaak context en detail om requirements te kunnen afleiden

Use cases

* Actoren bepalen
  + Wie (of wat) ontvangt een notificatie wanneer er iets in de applicatie gebeurd?
  + Wie (of wat) levert informatie of services aan de applicatie?
  + Wie (of wat) helpt de applicatie reageren en taken afwerken?
* Users vs. Actors
  + 1 gebruiker kan verschillende activiteiten uitvoeren
* Use case diagrams: zie software analyse
* Scenario’s
  + Verschillende activiteiten met een gemeenschappelijk doel
  + Use case is een verzameling van gerelateerde gebruikersscenario’s
  + 1 scenario is een specifieke instance van die use case
* Template
  + Structuur zodat alle info die nodig is voor een goede use case wordt verzameld
* Aannames
  + Wat moet er voldaan zijn alvorens de use case kan starten?
  + Trigger is geen aanname
  + Aanname beschrijft de staat van de applicatie maar niet de intentie van de gebruiker
    - Applicatie moet aannames kunnen testen
    - Zorgt ervoor dat bepaalde foutmeldingen vermeden kunnen worden
* Resultaten
  + Staat van de applicatie nadat de use case succesvol werd uitgevoerd
  + Observeerbaar, fysieke uitkomst, interne wijziging in de applicatie

--> Aannames en resultaten zijn niet altijd evident voor gebruikers dus de analist speelt hier ook een rol !

* Primair scenario, secundair scenario en uitzonderingen
  + 1 primair scenario en verschillende alternatieve (secundaire) scenario’s die een gelijkaardige uitkomst hebben
  + Vooraf reeds gekende uitzonderingen worden ook opgenomen
    - Exceptions handeling toevoegen is belangrijk zodat developers niet teveel veschillende interpretaties gaan maken
* Niet elke alternatief scenario dient te worden geïmplementeerd
  + Sommige kunnen worden uitgesteld
  + Uitzonderingen moeten steeds worden geïmplementeerd
* Use cases worden gebruikt om functionaliteiten op te sporen, het is niet noodzakelijk dat elke functionaliteit in een use case wordt gegoten

****

Identificeren van use cases

* Eerst de actoren identificeren, deze naast de business processen leggen en use cases definiëren voor activiteiten waar actoren en applicatie interageren
* Creëer een specifiek scenario om een business proces te illustreren en dan een generiek scenario maken
* Op basis van een business proces kijken welke taken nodig zijn binnen de applicatie om inputs naar outputs om te zetten
* Kijk naar het context diagram en kijk welke resultaten alle externe systemen uit de applicatie willen halen

Use cases vs. functional requirements

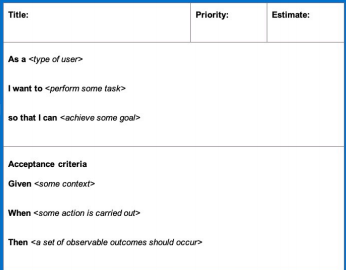
* Use cases zijn geen functional requirements!
  + Beschrijven het perspectief van de gebruiker op het extern zichtbaar gedrag van de applicatie
    - Geen informatie over interne parameters en processen
    - Geen informatie over wat de applicatie moet doen als aannames niet vervuld zijn
* De analist moet de functionele requirements oplijsten die nodig zijn om een bepaalde use case te implementeren
  + Beter apart gezien er nog andere functionele en niet-functionele requirements zijn die niet aan een use case hangen, of dat er functionele requirements zijn die aan meerdere use cases hangen (SRS)

Use cases – te vermijden

* **Te veel use cases:** probeer geen aparte use case te maken voor elk scenario (denk aan primaire en secundaire scenario’s)
* **Heel complexe use cases:** zelfs een use case met veel alternatieve scenario’s moet duidelijk en bondig beschreven worden
* **Design toevoegen in use cases:** use cases beschrijven conceptuele interacties tussen gebruiker en applicaties, niet hoe het in de applicatie zal gepresenteerd worden
* **Data definities toevoegen:** verschillende data types kunnen voorkomen in verschillende use cases dus beter ergens centraal bij houden om inconsistenties te voorkomen
* **Use cases die de gebruikers niet begrijpen:** beschrijf vanuit een gebruikersperspectief en niet vanuit applicatie-perspectief

User stories

* Ook in user stories worden actoren en een resultaat bepaald, maar wordt ook de reden waarom meegegeven
  + Dit kan helpen toegevoegde waarde te bepalen en prioriteiten te stellen
* Acceptance criteria
  + Wanneer is de doelstelling van een bepaalde user story behaald
  + Basis voor acceptance testing
* Wanneer user stories gebruikt worden in een agile project, zal ook de status, de inschatting, prioriteit, link naar de epic en eventuele takenlijst worden opgenomen.

****

User story – voorbeelden

* **Als** zorgverlener, **wil ik** inzicht in de medicatiehistorie van de patiënt, **zodat** ik bij twijfel de nieuw voorgeschreven dosering gemakkelijk kan verifiëren.
* **Als** reisorganisatie **willen wij**, naast de huidige autovakanties, ook vliegvakanties gaan aanbieden, **zodat** we een breder marktsegment kunnen bedienen.
* **Als** een geregistreerde gebruiker, **wil ik** een nieuw wachtwoord kunnen aanvragen, **zodat** ik toch toegang kan krijgen als ik mijn wachtwoord ben vergeten

**8. Documenteren en Modelleren**

Introductie

* Enkel informatie zijn geen requirements
  + Business requirements -> Scope Statement
  + User requirements -> Use case/User story
  + Functional/Non-functional requirements -> SRS (Software Requirements Specification)
* In deze les bespreken we de SRS
  + Kan een document zijn maar ook een spreadsheet, Requirements Management Tool,…
  + Niet iedereen is overtuigd van het nut van requirements documentatie maar de kost om het achteraf terug te gaan achterhalen is veel hoger

Software Requirements Specification

* Heeft verschillende namen: functionele specificatie, product specificatie, requirements document, onderdeel blueprint,…
* Beschrijft
  + de functies van een applicatie, met de karakteristieken en beperkingen
  + het gedrag van de applicatie onder verschillende omstandigheden
  + kwaliteiten van de applicatie zoals performance, security, useability,…
* Bevat geen
  + Design, implementatie, testing of project management details
* Wordt gebruikt door:
  + Klanten
  + Project Managers
  + Developers
  + Testers
  + Operations
  + Documentatie
  + Trainers
  + …

Agile & Functional Requirements

* Niet alles moet vooraf opgenomen zijn maar voor elke iteratie moeten de functionele requirements beschreven en geanalyseerd zijn
* Baseline requirements moeten duidelijk zijn
  + Proces van analyse en validatie
  + Minimaliseert miscommunicatie
  + Dient als ijkingspunt

Inhoud van de SRS

* Zorg voor een goede leesbaarheid
  + Gebruik een template
  + Gebruik consistente benaming en labels voor requirements
  + Gebruik visuele hulpmiddelen (onderlijnen, cursief,…) correct
  + Maak een inhoudstafel
  + Nummer figuren en tabellen
  + Gebruik links om naar andere documenten te verwijzen
  + Gebruik modellen en andere visuele voorstellingen

**SRS**

* **Lees alle puntjes van de SRS in PowerPoint**
* **En voor modeleren ook**

**9. Prioteriseren en Valideren**

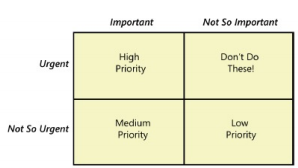
**Requirements Prioteriseren**

Pragmatisch prioriteiten stellen

* Prioriteiten stellen kan op feature niveau of op functionaliteit niveau
* Betrek zoveel mogelijk stakeholders wanneer prioriteiten worden gesteld
* Factoren om rekening mee te houden
  + The nood van de klant
  + De relatieve belangrijkheid van requirements
  + De timing wanneer het resultaat moet opgeleverd worden
  + Eventuele causale verbanden of andere relaties tussen requirements
  + Welke requirements in een ”groep” moeten worden opgeleverd
  + De kost van elke requirement

Enkele technieken

* **In or out:** voor elke release checken welke mee moeten en welke niet
* **Gepaard vergelijken en ranken:** steeds 2 requirements tegen elkaar afwegen en beslissen welke belangrijkste is
* **Three-level scale**: high, medium, low
  + **High**: zowel dringend als belangrijk
  + **Medium**: wel belangrijk, niet dringend
  + **Low**: niet belangrijk, niet dringend
  + **4de optie:** wel dringend maar niet belangrijk - meestal dringend voor één stakeholder maar levert weinig tot geen waarde op



* **M**o**SC**o**W**:
  + **Must**: essentieel onderdeel van de oplossing
  + **Should**: belangrijk maar niet essentieel
  + **Could**: interessant maar kan uitgesteld of weggelaten worden dus enkel te implementeren als de tijd en resources het toelaten
  + **Won’t**: niet voor deze release, eventueel voor een volgend

**Requirements Valideren**

Introductie

* Requirements zijn vaak onduidelijk of onvolledig
  + Dit leidt tot fouten bij de implementatie en frustratie bij de developers
  + Een fout herstellen na implementatie is duurder dan de requirements vooraf nakijken en valideren
  + Testen worden vaak pas op het einde ingepland waardoor de fout pas laat aan het licht komt
  + Requirements kunnen ook al “getest” worden zonder werkende software

Validatie en Verificatie

* Verificatie: nagaan of de requirements juist beschreven zijn
* Validatie: nagaan of je de juiste requirements hebt beschreven
* Factoren
  + De SR beschrijven correct wat het systeem moet doen zodat het aan de noden van de klant kan tegemoet komen
  + De SR zijn correct afgeleid van de business requirements, systeem requirements, business rules,…
  + De requirements zijn compleet, haalbaar en verifieerbaar
  + Alle requirements zijn nodig en voldoende om de businessdoelen te bereiken
  + Alle requirements werden op een consistent manier gedocumenteerd
  + De requirements leveren een goede basis voor design en development

Reviewing requirements

* Peer review
  + Nalezen door een collega
  + Een meeting met verschillende collega’s om door het document te gaan
  + Bespreking van het document met ruimte voor comments
* Zowel formeel als informeel
  + Inspectie: formeel nakijken van elke requirement volgens een bepaald proces
  + Voorkomt dat zaken vergeten worden, of verkeer geïnterpreteerd

Inspectie

* Deelnemers
  + De auteur van het document
  + De bronnen van de informatie in het document
  + De eindgebruikers
  + Gebruikers van de applicaties die een interface hebben met de applicatie
  + Max 7 inspecteurs
  + Niet alle profielen zullen altijd aanwezig zijn, maar te grote groepen leiden vaak tot discussies
* Rollen
  + Auteur: passieve rol
  + Moderator: plant en faciliteert
  + Lezer: vat de requirement samen voor de groep
  + Recorder: documenteert issues en fouten die worden gevonden
* Criteria voor documenten
  + Enkel afgewerkte documenten zonder grote fouten
  + Duidelijke structuur en nummering
  + Open issues zijn duidelijk aangegeven
* Afronding
  + Alle issues zijn besproken en uitgeklaard
  + Alle wijzigingen zijn correct doorgevoerd

Problemen met de review

* Grote requirement documenten
* Grote inspectieteams
* Niet alle reviewers zitten op de dezelfde locatie
* Reviewers zijn slecht of niet voorbereid

Validatie van Requirements

* Acceptatie criteria opstellen zorgt ervoor dat we zowel de requirements als de applicatie achteraf kunnen valideren
  + SMART (Specific, Measurable, Attainable, Relevant, Time-Sensitive)
  + Specifieke functionaliteit met hoge prioriteit moet aanwezig zijn
  + Essentiële niet-functionele criteria en kwaliteitseisen moeten voldaan zijn
  + Welke issues/defects er nog mogen open staan
  + Eventuele legale of contractuele vereisten
  + Aanwezigheid van transitie, infrastructuur of projectdeliverables
* Acceptatie criteria zijn geen testen, maar voorwaarden waaraan moet voldaan zijn om de applicatie als succesvol te beschouwen
* UAT (User Acceptance Testing) zal testen of aan de criteria voldaan werd

**10. Requirements for Agile projects**

**Agile developement**

Iteratief ontwikkelen

* Bij iteratief ontwikkelen gaan we het ganse project vanaf de ontwerpfase opsplitsen in kleinere projectjes
* In de eerste iteratie wordt de basisoplossing opgeleverd
  + Na elke cyclus is er een werkende oplossing en worden er nieuwe functionaliteiten toegevoegd
  + Klanten worden bij elk deelproject betrokken
* Risico dat de klant niet tevreden is met het eindproduct wordt geminimaliseerd
* Zeer flexibel omdat analyse in elk deelproject gebeurd

Agile ontwikkelen

* Agile is een variant op het iteratief model
  + MVP (minimum viable product) na elke iteratie
  + Zeer korte iteraties • Sterke direct communicatie
* 4 vuistregels om goede software te maken

1. Personen en interacties boven processen en tools.
2. Software die werkt boven lijvige documentatie.
3. Samenwerking met de klant boven onderhandeling over het contract.
4. Omgaan met verandering boven het volgen van een plan

* Geen concrete planning
  + Wel een overzicht van het werk per iteratie
  + Continu aanpassen aan de veranderende omgeving en de vereisten van de klant
* Toch belangrijk om beperkt ontwerp op te maken om budget en doorlooptijd te kunnen bepalen
* Bekendste agile methodologie is SCRUM

SCRUM

* Zoveel mogelijk waarde creëren op een zo kort mogelijke termijn
* Elke sprint (iteratie) duurt 2- 4 weken
* De klant bepaalt de prioriteiten, het projectteam organiseert zichzelf om de prioriteiten te kunnen uitvoeren
* **Product backlog:** alle functionaliteiten die moeten geïmplementeerd worden in het project
  + Wordt vooraf vastgelegd maar kan nog wijzigen
* **Sprint backlog:** alle functionaliteiten die moeten geïmplementeerd worden tijdens een sprint
  + Functionaliteiten worden in detail geanalyseerd en ingeschat door de ontwikkelaars (planning poker)
* **Sprint planning:** op basis van de prioriteiten van de klant
* Tijdens de sprint mag er niets meer gewijzigd worden
* Verschillende communicatiemomenten
  + **Daily standup:** dagelijkse meeting waar elk teamlid vertelt wat hij heeft gedaan, wat hij zal doen en welke problemen die heeft
  + **Sprint review:** na elke sprint wordt er getoond wat er gerealiseerd werd aan de klant en andere stakeholders
  + **Sprint retrospective:** na elke sprint worden de goede en slechte punten besproken en worden er actiepunten gedefinieerd voor de volgende sprints te verbeteren
* Er is geen projectmanager maar een SCRUM master in het team
* Het team bestaat uit maximum 9 personen
* Opvolging gebeurt via een scrum-board of burndown chart
* Ook in SCRUM worden alle fases nog doorlopen:
  + **Analyse gebeurt in 2 delen:** een stuk vooraf voor budget en planning te bepalen en meer gedetailleerd tijdens de sprints
  + **Implementatie:** functionaliteiten worden tijdens de sprint toegevoegd
  + **Test:** gebeurt volledig tijdens de sprints, vaak geautomatiseerd
  + **Turnover:** na elke sprint wordt er afgewerkte software opgeleverd dus dient er ook steeds een turnover te gebeuren

**Requirements for Agile projects**

Agile Requirements vs. requirements for agile projects

* De requirements bij Agile development zijn niet anders dan die bij een andere ontwikkelstrategie
  + De informatie en format is dezelfde
  + De timing en uitwerking is anders
* Problemen met waterfall
  + Projecten worden vaak te laat opgeleverd
  + Belangrijke functionaliteiten missen
  + De applicatie voldoet niet aan de wensen van de klant

🡪 Requirements zullen altijd veranderen gedurende een project en Agile werken geeft meer flexibiliteit

Belangrijke aspecten van de agile aanpak van requirements

* Klanten betrekken
  + Dit is de sleutel tot succes in elke project ongeacht de ontwikkelstrategie
  + Maar: bij waterfall is de klant enkel betrokken in het begin en tijdens de UAT, terwijl ,bij agile de klanten altijd betrokken blijven
  + Feedback loops, product owner,…
* Detail van documentatie
  + Omdat bij waterfall er enkel contact is met de klant bij de start van het project, moeten de requirements tot in het grootste detail gedocumenteerd zijn
  + Bij agile kunnen deze nog verfijnd worden in de sprints omdat de klant continu samenwerkt met het development team
* Backlog en prioriteiten stellen
  + Elk agile project heeft een backlog met user stories (en soms ook andere requirements en bug fixes)
  + Elke sprint wordt beslist welke user stories prioriteit hebben en dan worden deze ook verder verfijnd
* Timing
  + Alle requirements worden nog steeds opgesteld aan het begin van het project maar de verfijning gebeurt later
  + Er kunnen ook makkelijker requirements worden toegevoegd of verwijderd op basis van de feedback loops
* Epics, user stories and features
  + User stories zijn een beschrijving van wat een gebruiker zou willen doen in de applicatie, meestal gelinkt aan een use case
  + User stories moeten kunnen geïmplementeerd worden in 1 iteratie
  + Een epic is een user story die te groot is voor 1 iteratie en kan dus opgesplists worden in meerdere kleinere user stories
  + Een feature is een verzameling van user stories en/of epics die waarde oplevert voor een gebruiker
* Verwacht veranderingen
  + Omdat er gewerkt wordt met constante feedback loops, zal een agile project altijd meer veranderingen kennen dan een waterfall project
  + Requirements kunnen dus toegevoegd en verwijderd worden aan/van de scope
* Agile development vraagt ook een agile aanpak
  + Volg niet blindelings de “regeltjes” maar kijk wat er kan in de organisatie en behoud wat praktisch en haalbaar is