Analyse II

Lorenz Verschingel

21 december 2014

Samenvatting

Samenvatting Analyse II HoGent

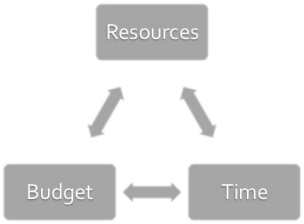
Inhoudsopgave

1 Inleiding

1.1 Waarom Analyse?

Om kwaliteitsvolle projecten op te leveren.

- Kwaliteit
  - uitbreidbaar
  - binnen parameters budget, tijd, mensen
  - ...
- Project
  - scope
  - tijd
  - resources
  - budget
  - éénmalig
- Opleveren
  - Tot de klant tevreden is.



Het grootste probleem in het verleden met ICT-projecten waren de tijd en het budget. Deze vormden problemen omdat men vroeger weinig voeling had met ICT-projecten.

Het is belangrijk dat in de analyse de requirements duidelijk vast gelegd worden. M.a.w. dat de scope duidelijk vastligt.

Big Data: enorme data die niet in 1 relationele databank kan.

2 Requirements

2.1 Situering

Ieder systeem wordt ontwikkeld om een bepaalde reden:

- Vooraf eisen helder en eenduidig formuleren: (= bussiness case)
  - Identificeren van de stakeholders
  - Formuleren van de Kritische SuccesFactoren (KSF)
  - Formuleren van meetbare acceptatiecriteria

2.1.1 Kritische succesfactoren

De KSF worden redelijk abstract geformuleerd.

Voorbeelden:

- begrijpbaarheid
- snelheid
- werkbaarheid onder verschillende systeemsoftware

2.1.2 Meetbare acceptatiecriteria

De acceptatiecriteria zijn concreet en leggen uit hoe de KSF gemeten zal worden.

De acceptatiecriteria worden SMART geformuleerd:

- Specifiek
- Meetbaar
- Acceptabel
- Realistisch
- Tijdsgebonden

Men kan niet altijd voldoen aan alle voorwaarden van SMART.

2.2 Soorten Requirements

2.2.1 Niet-Functionele Requirements

Soorten niet-functionele requirements:

- Look and feel requirements
  - bv: In lijn met huisstijl.
- Usability and humanity requirements
  - bv: Bruikbaar voor slechtzienden.
- Performance requirements
  - bv: Binnen 5 seconden opstarten.

- Operationele en omgevingsrequirements
  - bv: Leesbaar bij laaghangende zon.
    - beperkingen opleggen aan kleuren.
- Onderhoudsrequirements
  - bv: Systeem-updates moeten automatisch verlopen.
- Security requirements
  - bv: Aanmelden via e-ID.
- Culturele en politieke requirements
  - Oppassen met religieuze symbolen.
  - Meertaligheid van de applicatie.
- Legal requirements
  - De applicatie moet beantwoorden aan de geldende wetgevingen.

2.2.2 Functionele requirements

Met functionele requirements wordt het eenduidig vastleggen van de functionaliteit. Hiervoor worden use cases gebruikt.

Use cases

Enkele regels bij use cases:

- "My use case is not your use case"
  - de manier van noteren is niet altijd dezelfde. De beschreven functionaliteit wel.
- "Manage your energy"
  - juiste plan op het juiste tijdstip.
- "Keep it simple"
  - ca. 10 stappen
- Communiceer duidelijk

Proces van een use case:

1. Brainstorming [groep]
   - Elementaire businessprocessen = use case
   - Primaire actoren
   - User goals
   - Prioriteiten
2. Standaard vastleggen om de use case uit te schrijven [groep]
3. Schrijf de use case uit [max 2 personen]
4. Geef de use case door [max 2 personen]
5. Voorstelling van de use case aan de groep met discussie

Tips:

- Blijf in dezelfde kamer
- Juiste info door juiste mensen
- Kleinere groepen werken efficiënter
- Spendeer max. een halve dag met de eindgebruiker
- Management moet mee in de boot
- Actor is verschillend van de jobtitel
- Aantal use cases beperken
- Wees minimalist

Stakeholders zijn belanghebbende/deelnemende partijen

De primaire actor is de belanghebbende partij die het systeem vraagt een taak uit te voeren. De primaire actor is de persoon die de use case activeert.

Een use case is een elementair businessproces, duurt 2 à 20 minuten en wordt door 1 persoon uitgevoerd.

Structuur:

1. Situering
2. Korte beschrijving
3. Primaire actor
4. Precondities
    - Wat moet er gebeurd zijn om de use case te kunnen starten.
    - Geeft in veel gevallen aan dat een andere use case moet uitgevoerd zijn.
5. Postcondities
6. Normaal verloop
    - Top-to-bottom-beschrijving van een eenvoudige situatie, waarbij het doel van de actor wordt gerealiseerd.
7. Alternatieve verlopen

Tips bij het maken van de actiestappen in het normaal of alternatief verloop:

- Gebruik eenvoudige zinnen.
- Bij elke actie heeft 1 actor een boodschap.
- Vogelperspectief
- Toon de vooruitgang in het proces.
- Toon de bedoeling van de actor, niet de beweging.
    - Geen interactie met de GUI.
- Aanvaardbare set van actiestappen:
    - Transactie bestaat uit 4 delen: vraag, validatie, wijziging, resultaat.
- Valideer: vermijd if-statements
- User laat systeem A, systeem B aansturen

3 Ontwikkelstrategiën

3.1 Inleiding

3.1.1 Waterval-methode

1. Behoeften analyse
2. Analyse
3. Ontwikkeling
4. Testen
5. Oplevering

Bij de waterval-methode kan er nooit teruggegaan worden naar de vorige stap.

3.1.2 Agile

Agile is een filosofie gebaseerd op het Agile Manifesto. Het is een reactie op de waterval-methode bracht een mentaliteitswijziging op gang.

Enkele principes achter het Agile Manifesto:

- Hoogste prioriteit is de tevredenheid van de klant.
- Verwelkom wijzigingen.
- Lever geregeld werkende software op.
- Business en ontwikkelaars werken samen.
- Bouw producten rond gemotiveerde individuen.
- "Face to face" communicatie binnen het team.
- Werkende software is een meting van vooruitgang.
- Eenvoud is essentiëel.
- Zelforganiserende teams
- Zelfbespiegelende teams

Alle methodes die hierna besproken worden zijn Agile-methodes

3.2 Scrum

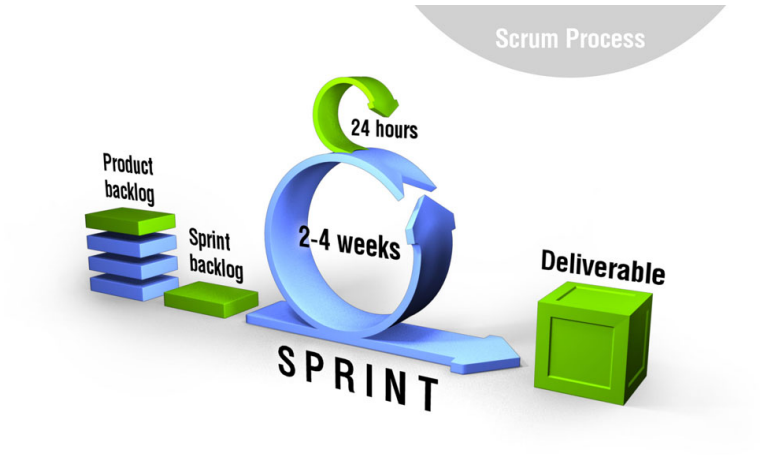
Scrum is een open framework gebaseerd op CAS (Complex Adaptieve Systemen), met als doel CAS te evolueren naar intellegentie.

Scrum is een iteratief en incrementeel proces.

- Iteratief:
  - Het zal niet allemaal meteen correct zijn.
- Incrementeel:
  - Verticaal bouwen i.p.v. horizontaal bouwen.

3.2.1 Scrum regels

- Werken in gesloten iteraties (geen scopewijzigingen)
- Elke iteratie productiewaardige software
- Zelforganiserende en zelfreflecterende teams
- Alles is geprioritiseerd



3.2.2 Rollen

- Product-owner
  - vertegenwoordigt de belanghebbende partijen
  - 1 stem, liefst 1 persoon
  - bepaalt wat er gebouwd moet worden
  - bepaalt wat er prioritair is.
  - doet commitment om het team met rust te laten tijdens de sprint
- Scrum-master
  - bewaakt het proces:
    - Scrumconcepten toepassen
      - Zelforganiserende en zelfbespiegelende teams
      - Werken in gesloten iteraties (Geen scopewijzigingen).
      - Elke sprint productiewaardige software
      - Alles is geprioritiseerd
    - Kwaliteitscontrole
      - Verantwoordelijk voor softwarekwaliteit.
  - zorgt voor continuous improvement
    - Alles een vraag stellen.
    - Mensen uit te dagen.

        * Niet zelf met oplossingen komen.
        * Eerst: Brandjes blussen.
        * Later: Structureel verbeteren.
- Team
    - Typisch 5-9 mensen.
        * Hoe meer mensen, hoe groter de communicatie overhead.
        * Indien groter team nodig; Werken met meerdere teams.
    - Leden zouden fulltime aan het project moeten werken.
        * Uitzondering bij ondersteunende functies.
    - Teams zijn zelforganiserend
        * Leden brengen ook een achtergrond aan,
        * Leden zijn multidisciplinair
        * Leden delen hun kennis
        * Leden werken mee aan domeinen waar ze geen specialist op zijn.

3.2.3 Overlegmomenten

Projecten worden uitgevoerd en een aantal sprints.

Een sprint duurt 2-4 weken. Een constante duur leidt tot een beter ritme.

Het product wordt ontworpen, geschreven en getest tijdens de sprint.

Voorbereiding sprint

- Auto takken worden opgemaakt door de product-owner en de klant.
- Oplijsting in een document (Product backlog)
- Uniek geprioritiseerd (dus niet in blokken)

Sprint planning deel 1

- Ruwe inschatting van het aantal items dat opgenomen zal worden
- Inschatting door het team
- Product-owner kan antwoorden op alle vragen van het team.

Sprint planning deel 2

- Backlog items worden opgesplitst door het team.
- Geen managers
    - Manager kent geen taken van individuen.
    - Managers maken geen beslissingen voor het team.
- Sprint Backlog wordt gemaakt.
    - Taken van 4-16 uur

Schattingen

- Planning poker in uren of punten.
- Door mensen die het werk zullen gaan doen.
- Liefst verbale communicatie in plaats van gedetailleerde neergeschreven specificaties
- Opgelet voor anchoring

Stand-up

- Parameters:
  - Dagelijks
  - 15 minuten
  - Rechtop staan
  - Niet om problemen op te lossen
- Drie vragen:
  - Wat deed je gisteren?
  - Wat ga je vandaag doen?
  - Welke obstakels liggen je weg?

Demo/sprint review

- Het team stelt voor wat het bereik heeft gedurende de sprint.
- Informeel: 2 uur voorbereidingstijd
- Deelnemers:
  - Klanten
  - Management
  - Product-owners
- Validatie: door testen opgesteld tijdens product plannen

Retrospective

- Deelnemers:
  - Team
  - Scrum master
  - Product-owner (Optioneel)
- Vragen:
  - Wat ging goed?
  - Wat kan er verbeterd worden?
- Resultaat:
  - Altijd acties bepalen
  - Tips:

    * Geen beschuldigingen
    * Gebruik nooit het woord "JIJ"
    * Gebruik een spreekstok

3.2.4 Documenten

Crum heeft heel weinig documenten.

- Product backlog
- Sprint backlog
- Burndown statistieken

Dit alles kan men opnemen in een Excel spreadsheet. Er bestaan ook meer geavanceerde tools.

Product backlog Een lijst van al het gewenste werk van het project. Dit is meestal een combinatie van story-based werk en task-based werk.

De lijst wordt geprioritiseerd door de product-owner

Sprint backlog De sprint backlog is het werkdocument van het team. Het is een opvolging van individuele taken.

Er wordt ingevuld wat er nog te doen valt om zo te zien en te corrigeren om het doel te behalen. Men doet dit iedere dag opnieuw.

Hiervoor wordt vaak een bord gebruikt.

3.3 Kanban

- Geen rollen
- Visueel bord
- Laagdrempelig
- Aantal:
    - Bottle neck
    - Beperken
- Optimalisatie (lean)
- Gebruik in productieomgeving/systeembeheer
- Agile