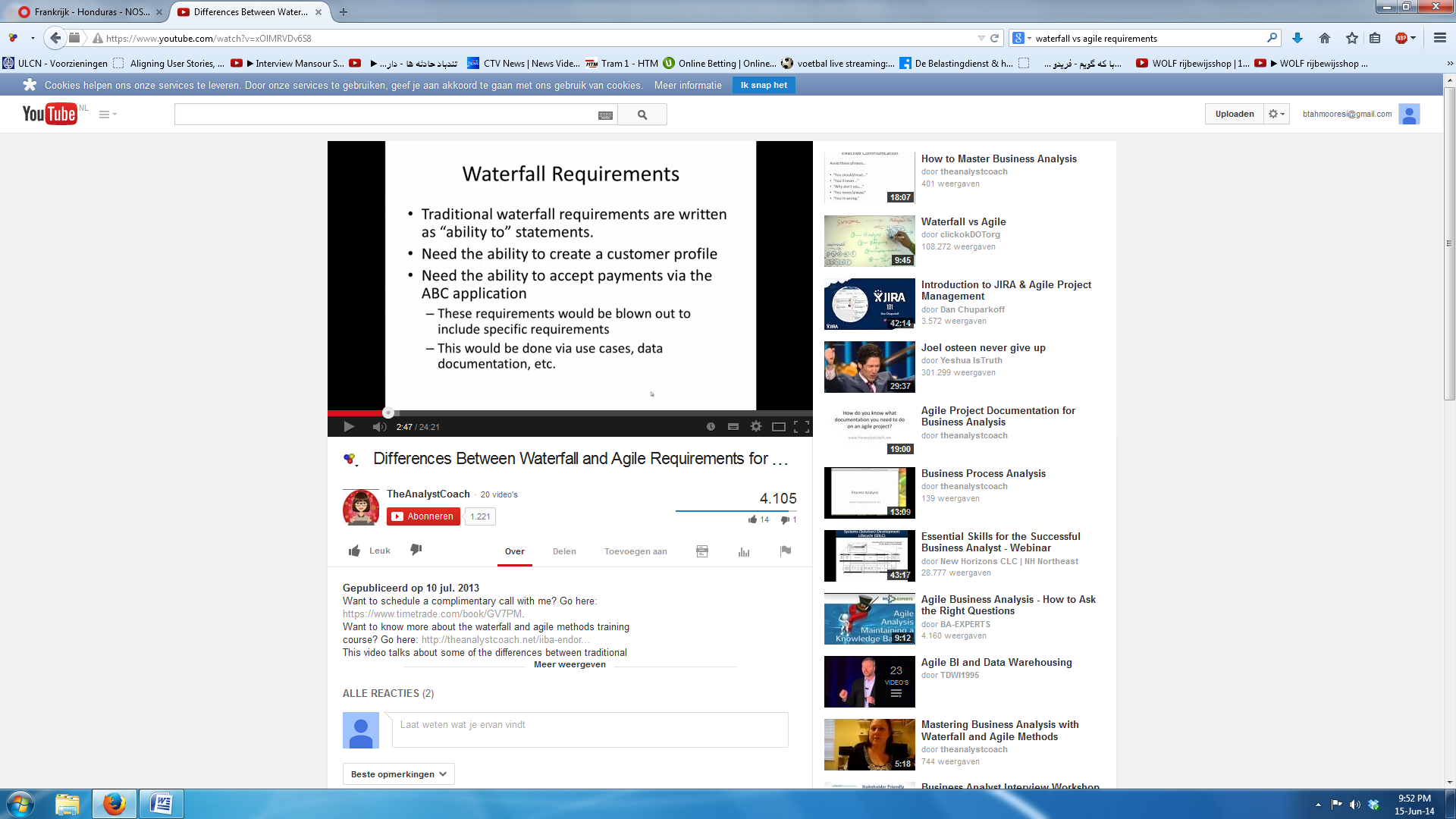
**Het vastleggen van Requirements**

**Dit document verschaft een overzicht van de meest gangbare manieren om Requirements op te schrijven, samen met de beschrijving van de belangrijkste attributen die voor elke Requirement minimaal vastgelegd dienen te worden. Dit document kan als naslagwerk worden gebruikt en heeft als doel om de wijze waarop Requirements binnen ons voortbrengingsproces worden vastgelegd te standaardiseren. Zonder concrete en eenduidige structuur is de grote hoeveelheid informatie die de specificaties van de gedetailleerde Requirements met zich meebrengen niet te begrijpen en ook niet terug te traceren naar de oorsprong.**

**A- Verschillende manieren om Requirements op te schrijven**

**1- Traditionele (klassieke)Requirement**

****Klassieke requirements worden geschreven vanuit het perspectief van de klant/gebruiker of het systeem. De traditonele requirements worden vaak geschreven als ''het moet mogelijk zijn om''.

Goede traditionele requirements schrijven kan tijdrovend zijn. Deze dienen namelijk in een eenvoudige, precieze, meetbare en eenduidige wijze, met een enkelvoudige betekenis te worden geschreven. Dit heeft ook als gevolg dat je lange opsomming van zinnen krijgt. De detaillering hangt af van de context van de wijziging, bv vroeg in het ontwikkelproces dienen de requirements minder gedetailleerd te zijn.

Traditionele Requirements kunnen zowel binnen een Agile omgeving als een traditionele omgeving worden gebruikt.

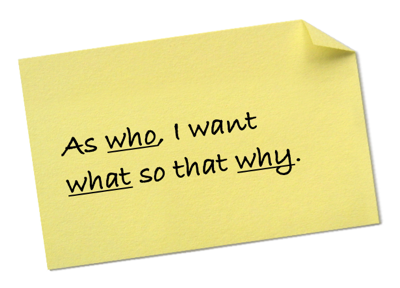
**Voorbeeld Traditional Requirements**

The candidate search system shall provide the ability to search for candidates by role, specialty, and geographic region.

The role shall consist of any role listed in the DB\_F01\_ROLE\_CAN database.

The specialties shall consist of any specialties in the DB\_F01\_SPEC\_CAN database.

The geographic regions must be specified via city, state, and optionally a radius in miles from the city.

**2- User Story**

User stories representeren requirements verteld vanuit het gezichtspunt van de gebruikers en geven de business values en behoeften van de gebruikers weer. Het geheel is weinig gedetailleerd en moet passen op een post-it. Hierin zegt men 'wie', 'wat', 'waarom' wil. User stories bestaan uit drie onderdelen:

1. Een korte beschrijving als aanduiding van de requirement. De requirements worden in korte eenvoudige zinnen beschreven, bij voorkeur op de volgende manier:

*Als een <type gebruiker> wil ik <iets doen> zodat ik < er iets aan heb>*

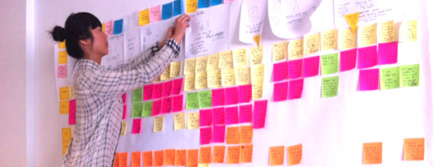
2. Mondelinge communicatie tussen alle relevante betrokkenen om de details te achterhalen

3. Acceptatiecriteria om juiste werking vast te stellen.

User stories worden vaak binnen een Agile omgeving gebruikt maar kunnen ook binnen een traditionele omgeving worden toegepast om zo bv de samenwerking en (mondelinge) communicatie tussen alle betrokken partijen te stimuleren.

Voordelen van het gebruiken van User Stories

* Om ze te schrijven is het nodig dat de gebruikers bij het systeem betrokken worden zodat de meest waardevolle zaken uit gebruikers oogpunt ook gerealiseerd worden.
* Ze zijn erg kort en nodigen uit tot face-to-face communicatie.
* Ze bevatten zaken die in een korte tijd gemaakt kunnen worden, eerder in dagen dan weken.
* Ze maken het mogelijk dat projecten in kleine onderdelen worden opgesplitst.
* Ze zijn erg geschikt voor projecten waar de eisen en wensen steeds wijzigen, of slecht begrepen worden.
* Ze maken het makkelijker om de tijd in te schatten die nodig is om het te maken.

Om User Stories overzichtelijk te kunnen documenteren worden er in het praktijk story maps gegenereerd. Een story map is in een Agile omgeving de grafische weergave van de productbacklog. De stories worden gegroepeerd in een logische volgorde waarmee men in staat zou moeten zijn om het systeem aan een onbekende uit te leggen. Op deze manier is het zelfs mogelijk om grote systemen te beschrijven zonder het overzicht te verliezen.

Binnen ons voortbrengingsproces worden ook Story maps gemaakt, hierbij wordt er geen gebruik gemaakt van een tool.

**Voorbeeld User Story**

**Title:**  
Search for candidates by role, specialty, and geographic location.

**Description:**  
As a candidate search user, I need the ability to search for candidates by specialty so that I can more efficiently refer patients to specialists.

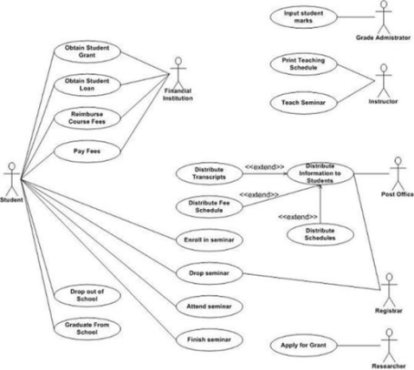
**Acceptance Criteria:**

* The candidate search mechanism has the ability to enter a role.
* The candidate search mechanism has the ability to enter a specialty.
* The specialty search will have a list of candidate specialties from which to select.
* Searching via the candidate specialty will return a list of matching specialists or a message indicating that there are no matches.
* If there are more results than can fit on one page, the system will provide the capability to view the list in pages or sections.

**3- Use Case**

Een use case beschrijft een systeem vanuit het gebruikersperspectief. Het beschrijft "wie" met het betreffende systeem "wat" kan doen. Het beschrijft de actor, de initiator van de interactie, en het systeem zelf als een opeenvolging van simpele stappen. Actoren kunnen eindgebruikers, andere systemen of hardware zijn, die bestaan buiten het te bestuderen systeem, en die deelnemen in de opeenvolgende activiteiten met het systeem om een bepaald doel te bereiken. Use cases kunnen zowel binnen een Agile als traditionele omgeving worden gemaakt en gebruikt.

Men kan een use case als uitwerking van een user story gebruiken, zo’n use case is dan de functionele beschrijving van wat er gebouwd is (wordt tijdens de sprint on-the-move gemaakt door beheerders of functioneel ontwerpers).



Use Cases bestaan in praktijk vaak uit twee componenten, namelijk:

**1-** Use Case diagrams, die de functionaliteiten van een systeem in termen van actoren, hun bedoeling uitgedrukt als use cases en de onderlinge relaties tussen deze use cases grafisch weergeven (zie figuur hiernaast).

**2-** Tekstuele beschrijving van de Use Case.

**Voorbeeld Use Case (tekstueel)**

The placement specialist selects a candidate search.

The system retrieves a list of candidate specialties and populates the candidacy list.

The system retrieves a predefined list of candidate specialties and populates the specialty  list.

The system retrieves a predefined list of geographic regions.

The system displays the candidate search screen.

The placement specialist selects a candidate specialty.

The placement specialist selects a candidate role.

The placement specialist selects a geographic region.

The placement specialist selects a geographic region.

The system identifies the region and populates the subregion.

The placement specialist selects the geographic subregion.

The placement specialist selects the search initiation button or mechanism.

The system retrieves a list of candidates who match the candidate specialty search. [Alt 1]

The system displays a list of candidates who match the search. [Alt 2]

**End use case**

Alt 1:  If there are no matches, the system displays a message indicating that no matches were found. End use case.

Alt 2:  If there are more matches than the user can view, the system will provide the capability to display multiple pages. End use case.

**B- Verschillende Requirements attributen die minimaal vastgelegd dienen te worden**

Hier volgt een overzicht van de belangrijkste attributen die per Requirement dienen te worden vastgelegd. Het is van groot belang om de Requirements op een gestructureerde wijze te voorzien van deze attributen. Dit zorgt er voor dat de grote hoeveelheid informatie die de specificaties van de gedetailleerde requirements met zich meebrengen begrijpelijk zijn en dat de traceerbaarheid wordt gewaarborgd.

**Prioriteit:** Per Requirement is de prioriteit ervan bepaald en vastgelegd met als resultaat een samenhangende set requirements waarbij een werkend product ontstaat zodat de belangrijkste eisen het eerst ontwikkeld en ook getest kunnen worden. Een gangbare manier om de requirements te prioriteren is MoSCow. Het is een afkorting, waarvan de letters staan voor:

* **M- must haves:** Het uiteindelijke resultaat moet voldoen aan de eisen met dit predicaat. Zonder een dergelijke eis is het resultaat onbruikbaar.
* **S- should haves:** Indien mogelijk moet het uiteindelijke resultaat voldoen aan de eisen met dit predicaat. In feite zijn deze eisen net zo belangrijk als de MUST, maar zijn er workarounds mogelijk waardoor deze een lagere prioriteit krijgt.
* **C- could haves:** Eisen met deze prioriteit zijn minder belangrijk, en worden ook wel "nice to have" genoemd. Deze eisen worden vaak meegenomen in het uiteindelijke resultaat indien ze eenvoudig en tegen lage kosten gerealiseerd kunnen worden.
* **W- would haves:** Eisen met deze prioriteit worden nu niet meegenomen in het projectresultaat, maar doorgeschoven naar toekomstige trajecten.

**Rationale:** Per Requirement wordt de rationale vastgelegd. Het vastleggen van de rationale is essentieel omdat het idee achter of de beweegreden voor een opgestelde of gewijzigde Requirement en de (ontwerp)beslissingen aan de hand van de rationale verklaard kunnen worden.

**Stakeholder/bron:** Per Requirement wordt een stakeholder/bron vastgelegd. Zonder een stakeholder/bron kunnen we niet van een ‘Requirement’ spreken. Een stakeholder/bron kan in het algemeen een individu, groep of organisatie zijn die invloed heeft op, beïnvloed wordt door, of van mening zijn dat zij worden beïnvloed bij een initiatief, (bijv. programma, project, risico). Binnen de BD zal de stakeholder/bron geen individueel persoon of enkel een organisatieonderdeel moeten zijn, maar een domein en rol > Bijvoorbeeld: IMB, TOEZICHT, ONTWERP. Tevens is het zo dat er bij verschillende Requirements niveaus verschillende belanghebbenden/bronnen kunnen horen.

**Unieke ID:** Elke Requirement zou gekoppeld moeten zijn aan een unieke identificatie. Zowel op hoog als laag niveau. Door een unieke naam, letters of nummer kan een Requirement worden getraceerd in het document. Hierdoor kan elk document dat refereert naar een specifieke Requirement een exacte referentie naar de Requirement maken.