INFANL01-5



*H. Aydogdu & M. Sozen*

INFANL01-5

|  |  |
| --- | --- |
| Module | Methoden voor software engineering |
| Docent | Mevrouw Tanja Ubert |
| Klas | Dinf2dt |
| Opleiding | Informatica |
| Auteur | H. Aydogdu  M. Sozen |

Inleiding

Als toekomstige software engineer is het belangrijk om bepaalde software & test methodes goed onder de knie te krijgen. Want hoe ga je de behoeftes van de klant aanpakken? Hoe ga je om met klanten die weinig budget hebben? Nou, dit verslag gaat over die bepaalde methodes. Hierbij kan je denken aan de Waterfall, incrementele en reuse methodes. Dit is erg belangrijk om te kunnen ontwikkelen volgens de wensen van de klant. Door de bovenstaande methodes toe te passen kan je heel veel informatie vergaren. Vanuit de vergaarde informatie is er een ruim beeld. De informatie kunnen we verwerken om vervolgens de gewenste applicatie te ontwikkelen door gebruik te maken van een van de bovenstaande methodes.

De *waterfall* methode wordt gebruikt om de ontwikkeling van een software vloeiend naar beneden te laten lopen. Er wordt namelijk ontwikkelt aan de hand van fases. Hierdoor is ook de benaming ontstaan. In dit verslag gaan we doornemen hoe de fases met elkaar communiceren en vloeien. Voorheen was het ontwikkelproces van een software onoverzichtelijk. Met deze methode is er meer duidelijkheid gekomen in ontwikkelprojecten.

Bij *incrementele* softwareontwikkeling bouw je het systeem geleidelijk en voegt er zodanig meer functionaliteit aan toe. Je begint met het ontwikkelen van de basis versie van het systeem, vervolgens laat je de omvang, functionaliteit en kwaliteit van het systeem langzaam groeien. Dit zou je bijvoorbeeld kunnen vergelijken met een tekening, naarmate je blijft tekenen ontstaat er een beeld. Hier zullen wij ons ook in verdiepen.

Daarnaast zullen we ons verdiepen in de *reused* methode. De reused methode is zoals zijn naam zich al omschrijft. Het gebruiken van eerder ontwikkelde software om een nieuwe software te ontwikkelen. Want waarom zou je helemaal beginnen van scratch?

Inhoudsopgave

[Inleiding 3](#_Toc463811784)

[Opdracht 1.a 5](#_Toc463811785)

[Opdracht 1.b 5](#_Toc463811786)

[Opdracht 1.c 5](#_Toc463811787)

[Opdracht 1.d - De waterfall methode 6](#_Toc463811788)

[Opdracht 1.d - de incrementele methode. 9](#_Toc463811789)

[Opdracht 1.e Change Management - Specification 9](#_Toc463811790)

[Opdracht 2.b Gebaseerd op etnografie 9](#_Toc463811791)

[Opdracht 1.d - reuse methode. 11](#_Toc463811792)

[Opdracht 2.a Use-case diagram 13](#_Toc463811793)

[Opdracht 2.c - vergelijken 15](#_Toc463811794)

[Opdracht 3.a 16](#_Toc463811795)

[Opdracht 3.b 22](#_Toc463811796)

[Opdracht 3.c 23](#_Toc463811797)

[Opdracht 4.a 23](#_Toc463811798)

[Opdracht 4.b 25](#_Toc463811799)

[Opdracht 4.c 25](#_Toc463811800)

[Evaluation and reflection 1 (Huseyin Aydogdu) 26](#_Toc463811801)

Opdracht 1.a

De pc-builder project is een “*Interactive transaction-based application*”.

De pc-builder is een toepassing welke uitgevoerd kan worden op een externe computer. In dit geval gaat het hierbij om de gebruiker die de website benaderd vanuit zijn eigen computer. De pc-builder is een webapplicatie waarbij er communicatie wordt gemaakt met een extern systeem. Hier kan de gebruiker componenten samenstellen en de producten aanschaffen. De pc-builder is een interactieve toepassing welke voorzien is van grote data die opgehaald wordt door verschillende websites. Componenten worden verzameld en er komt een configuratie als output.

Opdracht 1.b

Als we de pc-builder zouden moeten omzetten naar een andere soort type software. Dan hadden wij gekozen voor een *“Standalone application”*. De standalone applicatie bevat alle benodigde functionaliteiten en hoeft niet worden aangesloten op een netwerk. De standalone applicatie zou met updates voorzien kunnen worden van redelijk up-to-date informatie. De configuratie kan vervolgens worden afgedrukt om de onderdelen te bestellen bij de winkel.

Opdracht 1.c

De ontwikkelaar zou een afspraak kunnen maken met of ontwikkelen in opdracht van een grote partij zoals Bol.com en vervolgens gaan profileren dat het een onafhankelijk zoekmachines is. Nietsvermoedende gebruikers kunnen de dupe worden en moeten vervolgens wellicht meer betalen door deze afspraak.

De data dat wordt gecrawld kan gezien worden als een vorm van piraterij. De pc-builder kan data ophalen van websites welke niet worden bekend gemaakt door de pc-builder. Vervolgens wordt er wel een bestelling geplaatst op de desbetreffende website zonder dat er informatie over het bedrijf wordt gedeeld aan de gebruiker. Het kan dus zijn dat een bepaalde website niet wilt hebben dat er gesnuffeld wordt in zijn database waar de componenten toch goedkoper zijn waardoor de pc-builder aantrekkelijker is in gebruik.

Opdracht 1.d - De waterfall methode

**Requirement Gathering & Analysis;**

Alle mogelijke eisen van het systeem dat ontwikkeld zal worden, worden in deze fase gedocumenteerd. De gebruiker surft naar de webpagina van de pc-builder.

Vervolgens heeft de gebruiker de keuze om een nieuw configuratie te beginnen of een eerder opgeslagen configuratie te starten. Als de gebruiker kiest voor een nieuwe configuratie heeft de gebruiker de optie om te kiezen uit een component. Hierbij speelt de volgorde van de componenten geen rol. De gebruiker kan beginnen met het kiezen van een CPU, moederbord of een videokaart om de configuratie een draai te geven aan de gewenste eisen. Aan de hand van de gekozen componenten wordt door de pc-builder een filter gemaakt. De bedoeling van de filter is dat de uiteindelijke configuratie compatible is met de machine die wordt geconfigureerd. Spring Boot wordt gebruikt om de userinterface te genereren. De input van de gebruikers wordt gebruikt om de informatie van de database op te vragen. In feite is deze Spring-boot laag een service die de informatie in de database omzet in kleurrijke en gebruiksvriendelijke userinterface. In de backend wordt deze informatie eerst gefilterd op basis van compatibiliteit. De juistheid van de informatie wordt gezorgd door de verificatie proces die nodig is voordat de informatie op de website belandt. De crawler werkt onafhankelijk van de rest van het systeem en zorgt dat de database altijd wordt aangevuld met up-to-date informatie.

**System design;**

De specifieke eisen van de eerste fase worden bestudeerd en het design wordt voorbereid. Hierdoor beschrijven we kleine eenheden die geprogrammeerd worden. Bij het maken van de systeem design zijn we tegen bepaalde eisen aangelopen waarover we niet hebben gedacht.

Overzicht van de kleine eenheden die stapsgewijs geprogrammeerd moeten worden:

* **Crawler**

Een standalone service die zorgt voor input naar spring boot.

* **Spring Boot**

Service die zorgt dat de input wordt doorgevoerd naar de database en van de database naar de gebruiker.

* **Analytics**

De Koppeling tussen de website en externe partij die de statistieken bijhoudt.

* **Bestelling module(bezoeker)**

Een bezoeker kan de website benaderen en hoeft niet per se in te loggen. De bezoeker kan ook een pc configureren echter kan de bezoeker de configuratie niet opslaan indien de bezoeker niet inlogt. De gebruiker kan wel een bestelling plaatsen door alleen zijn NAW-gegevens in te voeren na een configuratie bij het bestel menu.

* **Bestelling module(gebruiker)**

Winkelmand, Bestelling en verwerken van betalingstransacties. De gebruiker heeft de mogelijkheid om in te loggen door middel van authenticatie. Hierbij kan de gebruiker een eerder geconfigureerde pc aanpassen en/of bestellen.

* **Userinterface m.b.t. admin**

Admin krijgt een speciale pagina te zien waar de informatie kan worden geverifieerd.

* **Database model**

De informatie die nodig is voor de matcher om te kunnen matchen en om een bestelling te plaatsen. Daarnaast informatie over transacties en bestellingen moeten voorkomen in de database model.

* **Matcher(backend)**

Matcher zorgt ervoor dat componenten die compatibel zijn worden getoond.

* **Bug meld module**

Gevonden bugs kan worden gemeld door met een knop de bug te melden.

**Implementation;**

Het systeem wordt eerst ontwikkeld in kleine eenheden zoals hierboven beschreven welke afkomstig zijn van de systeemarchitectuur. Nadat we de eisen hebben verzameld van de pc-builder zijn we begonnen met de opzet van de software. Het gaat om de eisen die beschreven zijn onder *“Requirement gathering & analysis”.* Vervolgens worden de eenheden getest op functionaliteit die ook wordt aangeduid als Testing Unit.

**Integration and Testing;**

Alles wat ontwikkeld is in de implementatiefase zoals hierboven beschreven worden geïntegreerd na het testen van elke eenheid. Het hele systeem wordt getest op eventuele fouten en mislukkingen. Alle modules worden getest met unittests om de afzonderlijke werking die verwacht wordt te waarborgen. Het systeem kent een onderscheid in test-, acceptatie- en productieomgeving. Hierdoor kan het systeem getest worden zonder dat de productie hier last van heeft. Bugs die worden gemeld vanuit de Bug meldt module worden geïmplementeerd in test en daarna in de acceptatieomgeving.

**Deployment of system;**

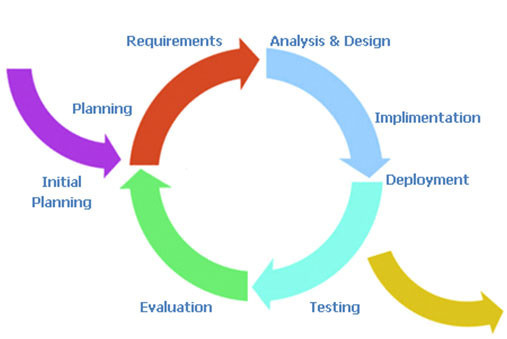
Zodra de pc-builder functioneel is en voldoet aan de functionele eisen wordt het systeem gepatcht in de productieomgeving. Tijdens deze patch wordt de website kort in maintenance gezet. De pc-builder wordt d.m.v. een go-live test getest zodat het volledig operationeel kan zijn en vervolgens wordt het vrijgegeven in de markt.

**Maintenance;**

Er zijn problemen die voorkomen in de omgeving van de klant. Om deze problemen op te lossen worden er patches gemaakt en uitgerold. Onderhoud wordt gepleegd om te voldoen aan de operationele eisen.

De crawler subsysteem staat los van de rest van het systeem en wordt ge-update d.m.v. patches wanneer daar behoefte voor is, zonder het systeem te sluiten voor maintenance. De userinterface en de backend worden gepatcht in de testomgeving en wordt pas live genomen nadat het volledig is getest.

Opdracht 1.d - de incrementele methode.



<http://www.testingexcellence.com/wp-content/uploads/2008/11/incremental-model-e1425812395282.jpg>

Opdracht 1.e Change Management - Specification

Aan de hand van etnografie wordt er een specificatie gemaakt. Voor de pc-builder is dit niet een juiste theorie om mee te werken. Dit heeft te maken met het feit dat de pc-builder een onderdeel is van de consumentenmarkt en geen bestaand beroep is, waarbij je kunt analyseren wat de gebruiker doet. Echter heeft de stakeholder voor deze methode gekozen omdat de stakeholder niet uit zijn comfortzone wilt stappen. Omdat wij een professioneel bedrijf zijn die een zakelijk houding hebben gaan wij dit aanpakken d.m.v. prototyping door de initiële versie te baseren op de outline definitie van het gesprek met de stakeholder. Vervolgens kiezen we een groep van 10 testgebruikers die met de executabele prototype gaan werken en wordt met deze resultaten het evaluatierapport opgesteld. Door gebruik te maken van prototyping zorgen we dat we een redelijk aangevulde requirements hebben om de initiële versie te kunnen bouwen. Als eerste heeft de stakeholder als voorbeeld de pc-configuratie van Alternate laten zien. Hieruit zijn een aantal vragen naar voren gekomen.

In de situatie van Alternate kan de gebruiker 3 soorten computers assembleren. Namelijk een Game pc, een pc van scratch of een server.

Opdracht 2.b Gebaseerd op etnografie

**Vraag**: Wat zijn uw eisen? Wilt u ook 3 verschillende soorten computers laten configureren? **Antwoord**: Nee, ik wil een pc van scratch laten builden. Echter wil ik de optie wel openhouden voor in de toekomst.

**Vraag**: Dient de gebruiker in te loggen?

**Antwoord**: Nee, dit hoeft niet persé. Echter als de gebruiker de configuratie wilt opslaan en later verder gaan, dan moet het wel mogelijk zijn.

Aan de hand van Alternate is er de mogelijkheid om de gewenste component als eerste te selecteren

**Vraag**: Moet er een bepaalde volgorde zijn bij het selecteren van een component. Dus moet een gebruiker als eerst kiezen voor een bepaald moederbord, CPU, etc., of is de gebruiker daar vrij in?

**Antwoord**: Nee de gebruiker kan gewoon beginnen waar hij/zij wilt beginnen.

**Vraag**: Dient de gebruiker een configuratie volledig af te maken, dus een compleet computer voordat hij uitcheckt?

**Antwoord**: Nee de gebruiker kan ook alleen op zoek naar een component en vervolgens uitchecken(betalen/bestellen)

**Development**

Aan de hand van de specificaties wordt er ontwikkelt. Hierbij hebben we een beter beeld gekregen aan de hand van de vragen en antwoorden. Er moet een inlogscherm gemaakt worden waarbij gegevens worden opgeslagen voor de gebruiker die dit wenst. De gebruiker kan dan zijn/haar configuratie opslaan en later verder gaan. Ook bezoekers kunnen een computer configureren. Echter kunnen zij niet bestellen zonder in te loggen. De bezoekers kunnen een overzicht van de componenten uitprinten.

**Validation**

Pc-builder wordt functioneel getest door de developer, er wordt een demo gegeven aan de stakeholders. Daarnaast wordt er een demo groep geselecteerd die met het systeem kunnen testen. Hierbij kunnen we stunten door de configuraties volledig gratis aan te bieden om het interessanter te maken voor het publiek.

**Initial Version**

De eerste versie bevat alleen de belangrijke features die gebruikt kunnen worden. De belangrijke features zijn, zoeken naar componenten, configuratie samenstellen. Hierbij laten we zoekresultaten opslaan, inloggen en bestelling module terzijde. Deze modules worden naderhand in de volgende releases gebouwd.

**Intermediate Versions**

Nadat de belangrijke features zijn ontwikkeld, worden de nieuwe features met een intermediate versie uitgebracht. De applicatie wordt in feite om de belangrijke features heen ontwikkeld. Hierin wordt gedocumenteerd wat er in de belangrijke features terzijde zijn gelaten. Hierin nemen we opslaan van configuratie, inloggen van de gebruiker en zoeken naar de componenten mee. Tevens is dit een referentie voor de stakeholder om aan te kunnen tonen dat alles op planning is.

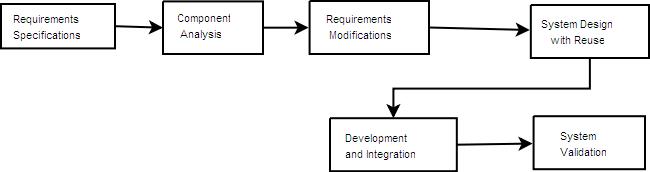
**Final Version**

De pc-builder moet volledig operationeel zijn en alle requirements moeten in de pc-builder verwerkt zijn. Wel wordt er een gehele system test gedaan, de zogenaamde black box test. Het doel van de integratie test is om het systeem in zijn geheel te testen. Vervolgens wordt er een final version afgeleverd.

Opdracht 1.d - reuse methode.

De reuse-oriented model ook wel bekend als reuse-oriented development (ROD) is een belangrijk onderdeel in de software engineering. Deze methode zorgt ervoor dat een nieuwe project niet volledig van scratch ontwikkeld moet worden, hierdoor kunnen we tijd en geld besparen.

De achterliggende gedachten van reused oriented developing is er voor zorg te dragen dat een software met een hoge kwaliteit wordt ontwikkeld en betrouwbaar is. Daarnaast worden de kosten ook gedrukt waardoor deze methode aantrekkelijk is voor de klant.

****

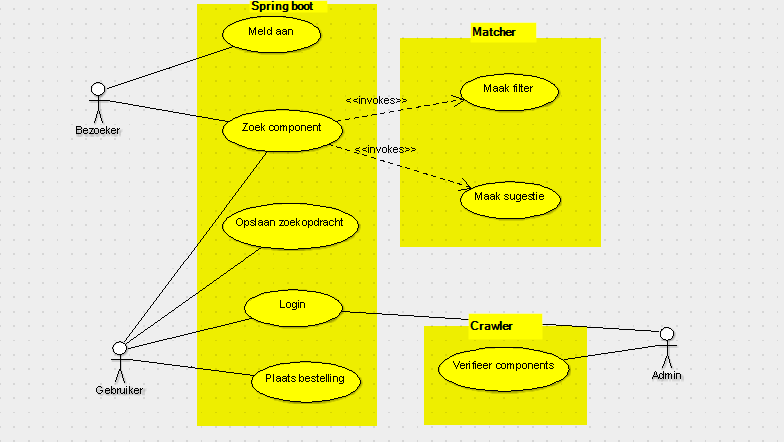
Requirements Specification

Voor het verkrijgen van de requirements hebben we gebruik gemaakt van de user story methode.

User story`s

|  |  |
| --- | --- |
|  | **Prio** |
| Als gebruiker wil ik op een willekeurig component kunnen zoeken zodat ik een keuze krijg gesorteerd op prijs, prijs/performance en performance. | 1 |
| Als gebruiker wil ik dat het systeem de gekozen componenten als filter gebruikt zodat ik de keuze kan maken die altijd compatibel is met elkaar. | 2 |
| Als gebruiker wil ik een master bestelling kunnen plaatsen die alle onderdelen besteld van verschillende websites zodat het gebruiksvriendelijk is. | 3 |
| Als gebruiker wil ik kunnen registreren zodat ik kan inloggen. | 1 |
| Als gebruiker wil ik kunnen inloggen zodat ik mijn bestellingen kan volgen. | 3 |
| Als gebruiker wil ik kunnen inloggen zodat ik mijn zoekopdracht kan opslaan. | 2 |
| Als admin wil ik componenten kunnen aanpassen en verifiëren zodat het systeem altijd up-to-date informatie bevat. | 1 |
| Als gebruiker wil ik een configuratie aanmaken en opslaan zodat ik een totaalprijs kan zien van mijn zoekopdracht. | 3 |
| Als gebruiker wil ik dat het systeem aan de hand van mijn configuratie  aangeeft welke component nog nodig zijn zodat ik die nog kan toevoegen. | 2 |

Opdracht 2.a Use-case diagram



Use case description

|  |  |
| --- | --- |
| Use Case Name | Een component zoeken |
| Participating Actors | Gebruiker of Bezoeker |
| Flow of Events | 1. Gebruiker logt in met een account  2.         Gebruiker zoekt naar een component  3.         Gebruiker kiest een of meer component(en)  4. Systeem gebruikt deze keuze als een filter  5. Systeem gebruikt deze keuze om een suggestie te geven  6. Gebruiker kiest een component vanuit de suggestie lijst  7. Gebruiker slaat zijn resultaat op  8.         Gebruiker plaatst geconfigureerde bestelling |
| Alternative Flows | 1.a Bezoeker logt niet in  6.a Gebruiker zoekt naar een component  7.a Bezoeker meldt zich aan om door te gaan |
| Entry Conditions | Gebruiker heeft juiste gebruiker authenticatie of meldt zich naderhand aan. |
| Exit Conditions | Gebruiker heeft zijn resultaat opgeslagen of bestelling geplaatst. |

|  |  |
| --- | --- |
| Use Case Name | Een component verifiëren |
| Participating Actors | Admin |
| Flow of Events | 1. Admin logt in met een account.  2.         Admin krijgt een lijst met componenten die ingevoerd is door decrawler.  3.         Admin kiest een component.  4. Admin controleert en verifieert deze component. |
| Alternative Flows |  |
| Entry Conditions | Admin heeft juiste gebruiker authenticatie.  Crawler heeft informatie ingevoerd. |
| Exit Conditions | Admin heeft een component geverifieerd. |

De requirements zijn merendeels beschreven maar niet tot in detail. Alle behoeftes van de gebruiker wordt geïmplementeerd in de pc-builder, hierbij gaan we gebruik maken van incremental delivery zodoende om er zeker van te zijn dat ook daadwerkelijk alles wordt geïmplementeerd. De requirements worden aangevuld en verfijnd tijdens de development van de pc-builder. Modules worden incrementeel opgeleverd.

***Component analysis -*** Om te beginnen kunnen we zoeken naar een kant en klare crawler die al eerder is ontwikkeld. Vervolgens kunnen we op zoek gaan naar een database. Met een beetje geluk kunnen we ook nog een interface vinden die aansluit op de pc-builder. Daarnaast kunnen we zoeken naar een manier om dit alles met elkaar te laten communiceren d.m.v. bijvoorbeeld Spring Boot.

***Requirements modification* -** Aanpassingen brengen aan de spring boot om met database te communiceren, aanpassen dat de userinterface wel aan de requirements voldoet(in de eerste instantie is het meestal een demo).

***System design with reuse -*** Bij het designen moeten de keuzes m.b.t. de requirements in acht worden genomen, bijvoorbeeld de userinterface moet multilingual zijn, zo niet kan het in deze fase worden gebouwd. User interface kan gewijzigd naar b.v. een responsive website. Daarnaast kan er een snellere crawler ingezet worden. Deze keuzes moeten in deze fase worden uitgewerkt.

***Development and integration* -** Als reused component(en) niet bestaan dan kan dit worden ontwikkeld. De bestaande componenten komen dan bij elkaar. In reuse oriented model is dit proces een onderdeel van de ontwikkeling en geen aparte activiteit.

Opdracht 2.c - vergelijken

**Gekozen methode:**

We hebben gekozen voor de reused methode. De reused oriented methode is zowel winstgevend voor de klant als comfortabel voor de ontwikkelaar. Hieruit kunnen we concluderen dat dit een win-win situatie is.

**Voordelen:**

De reused oriented method bedrukt de kosten t.o.v. andere methodes.

Het kan tijd besparen doordat elke fase van het proces al eerder is ontwikkeld.

Wanneer reused oriented method zorgvuldig wordt uitgevoerd, dan worden de kansen op fouten en/of bugs bedrukt. Hierdoor ben je sneller bij de final version.

Vele requirements zijn bekend doordat het al eerder is ontwikkeld.

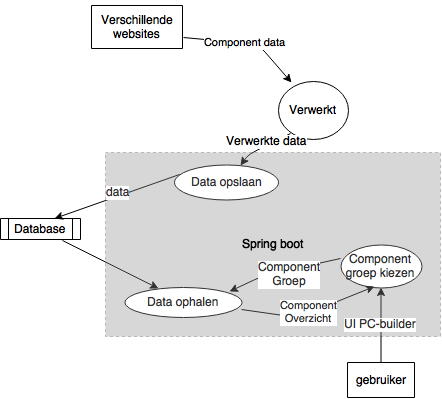
**Nadelen:**

De reused oriented model is niet altijd volledig, hierdoor moeten er nieuwe componenten worden ontwikkeld.

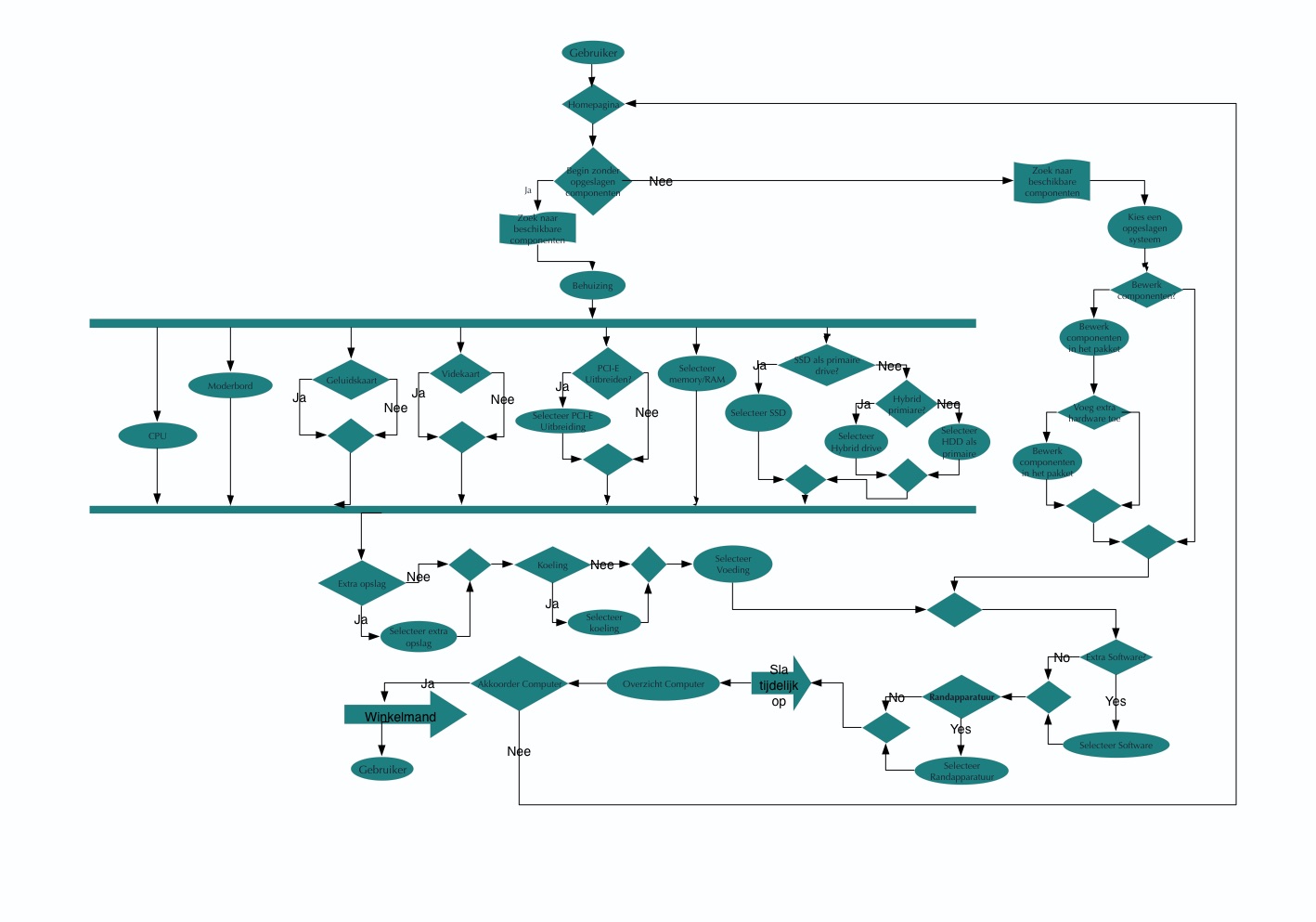
Als reused methode niet zorgvuldig wordt toegepast, dan kan het zijn dat je niet voldoet aan alle requirements. Dit resulteert in een onvolledige product.

Opdracht 3.a

DFD

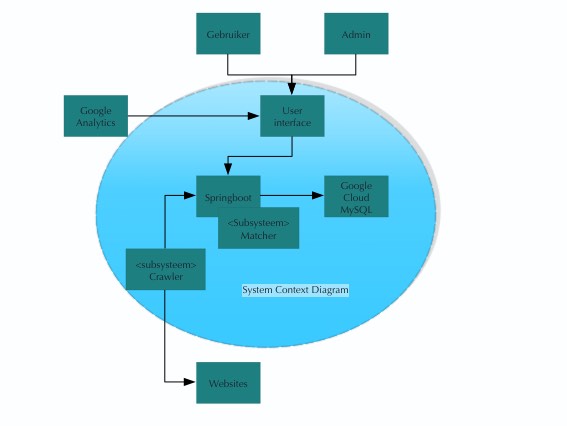


**Activity Diagram**

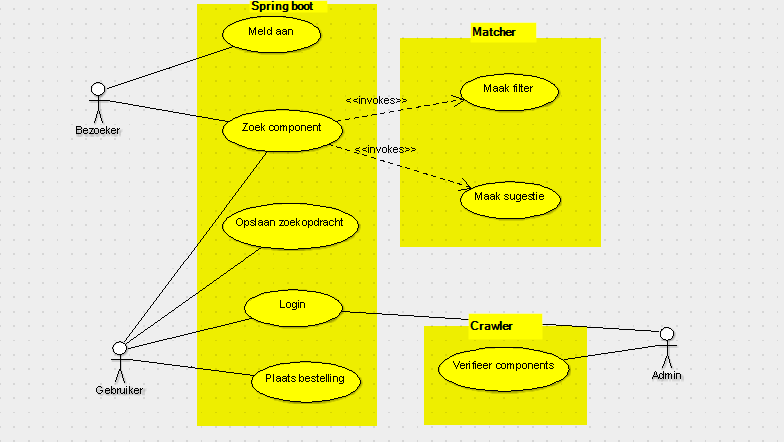


**Systeem context diagram**

In de onderstaande Context diagram kunnen we zien hoe de flow werkt van de pc-builder.



Use-case diagram

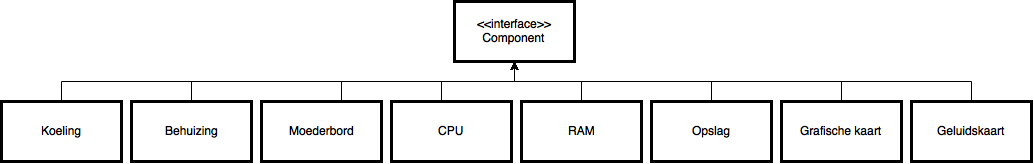


Use case description

|  |  |
| --- | --- |
| Use Case Name | Een component zoeken |
| Participating Actors | Gebruiker of Bezoeker |
| Flow of Events | 1. Gebruiker logt in met een account  2.         Gebruiker zoekt naar een component  3.         Gebruiker kiest een of meer component(en)  4. Systeem gebruikt deze keuze als een filter  5. Systeem gebruikt deze keuze om een suggestie te geven  6. Gebruiker kiest een component vanuit de suggestie lijst  7. Gebruiker slaat zijn resultaat op  8.         Gebruiker plaatst geconfigureerde bestelling |
| Alternative Flows | 1.a Bezoeker logt niet in  6.a Gebruiker zoekt naar een component  7.a Bezoeker meldt zich aan om door te gaan |
| Entry Conditions | Gebruiker heeft juiste gebruiker authenticatie of meldt zich naderhand aan. |
| Exit Conditions | Gebruiker heeft zijn resultaat opgeslagen of bestelling geplaatst. |

|  |  |
| --- | --- |
| Use Case Name | Een component verifiëren |
| Participating Actors | Admin |
| Flow of Events | 1. Admin logt in met een account.  2.         Admin krijgt een lijst met componenten die ingevoerd is door decrawler.  3.         Admin kiest een component.  4. Admin controleert en verifieert deze component. |
| Alternative Flows |  |
| Entry Conditions | Admin heeft juiste gebruiker authenticatie.  Crawler heeft informatie ingevoerd. |
| Exit Conditions | Admin heeft een component geverifieerd. |

Class Diagram



https://lh6.googleusercontent.com/_hJr-wiZK1vWwY-JoeoTmq-6TJJWdrEH2zoIDQLM70Nn-fKq2J7tcYYHn2_7AVY_2u5MLyTHASWYJ4sogjMxXasp4R5O65PAKz5HJq-rlrki9YL9D8AbbuL0hJFI7UhUq_mdIJ-H

Sequence Diagramhttps://lh4.googleusercontent.com/BU28jQFd3eJOXdcguMX2rewdK3xmQRvOc6W76Dz5vOJDEBenjITJFR6LV_ws0CB-njTMtfwLq7ggk_7s4RVqCRKDPegQ4qjnJvKDT2o6vtgf6dQoNBPUs6Rt144zwOdS8hDn2_rl

Opdracht 3.b

Passende architectuur

De hoogste waarde van layered architectuur is dat de verschillende layers, hun eigen verantwoordelijkheid hebben. Dit zorgt voor waarde bij de klant. Bijvoorbeeld: de klant komt alleen in contact met de userinterface waar de business logic zich niet in bevindt. De business logic bevindt zich in een andere layer. De data layer zorgt voor de afhandeling van de database.

Deze layer architectuur zorgt dat de code makkelijker begrijpbaar en te ontwikkelen is.

Bug fixen gaat dan ook makkelijker, sneller en overzichtelijker. Testen in deze architectuur is ook makkelijk omdat het modulair is. Veranderen en toevoegen van nieuwe technologie is hierdoor efficiënter.

https://lh3.googleusercontent.com/JlrsS5DKfZMrHiEHuoBeHDS3x1jBqWvfkMeKX5TdYdQKIttfBR7fkgs1vlXxonRfOEFaBKT8eq--gZ_m_K5TY8tWeXpruZnrJPaPlqkG0Q1R5UwIg7ppac3a558cg5UGnADOTVVB

Opdracht 3.c

Gebruik maken van design patterns onder andere singleton en strategy-pattern. Dit is een vorm van abstract level van reuse. Daarnaast zullen we gebruik maken van library`s zoals apache.common.io waar nodig. Deze object level reuse is een goede manier om een bepaalde functie sneller te ontwikkelen. Wij gaan gebruik maken van spring boot dit is een goed voorbeeld voor component level reuse, Het zorgt voor de database connectie door alleen de configuratie.xml aan te passen.

We geen gebruik maken commercial off the shelf producten maar dit system level reuse is in sommige situatie een goed alternatief. Het verkrijgen van expertise is ook makkelijker omdat er meer mensen zijn die met een COTS-systeem hebben gewerkt dan een nog te ontwikkelen systeem.

Opdracht 4.a

**Maak een risicoanalyse voor het project en beschrijf hoe je dit risico gaat managen.**

**Risico identificatie -** Risico identificatie is afhankelijk van de omgeving waarin het systeem wordt gebruikt. Risico’s kunnen ontstaan door interacties tussen het systeem en buitengewone condities van de operationele omgeving.

Risicoanalyse checklist

Stap 1: Identificeer kritieke onderdelen van het systeem

In een tabel zijn de kritieke onderdelen van de pc-builder opgenomen, met een motivatie waarom deze onderdelen kritiek zijn. Onder ‘kritiek’ wordt hier verstaan een veiligheidsrisico lopend.

Stap 2: In een tabel wordt weergegeven wat het risico is van een gebeurtenis, rekening houdend met de volgende variabelen:

* de impact van een gebeurtenis;
* de kans dat de gebeurtenis optreedt;
* de schade bij het optreden van de gebeurtenis.

Aan de impact van een gebeurtenis wordt een ‘rating’ toegekend tussen A (verwaarloosbaar risico) en E (hoog risico). De rating is in deze risicoanalyse al gecorrigeerd voor de reeds genomen maatregelen, waardoor het risico gereduceerd dan wel ondervangen is.

Schematisch kunnen we dit als volgt weergeven:

|  |  |  |
| --- | --- | --- |
| Impact | Schade | kans  (in %) |
| D | Mysql is down | 5 |
| A | Library wordt niet meer geupdate | 35 |
| B | Crawler werkt niet meer | 25 |
| C | Security leak in eigen code | 10 |
| B | Security leak in een library | 2 |
| E | Website werkt niet meer | 5 |

**Risk analysis and classification -** Elk risico wordt als afzonderlijk beschouwd. De serieus potentiele en de onwaarschijnlijke risico’s worden geselecteerd voor verdere analyse.

In deze fase worden bepaalde risico’s terzijde gelaten, dit heeft te maken met het feit dat ze onwaarschijnlijk zijn en niet detecteerbaar zijn door het systeem.

**Risk decomposition Each -** Risico wordt geanalyseerd om mogelijke dieperliggende oorzaken te ontdekken van die risico. De dieperliggende problemen zijn de oorzaak van het falen van het systeem. Dit kunnen zowel hardware als software error zijn.

**Risk reduction** - Er worden voorstellen gemaakt om risico`s te beperken of risico`s weg te schrappen. Dit draagt aan systeem betrouwbaarheidseisen. Dit zorgt voor risk preventie en riskmanagement.

Opdracht 4.b

In software development komen changes altijd voor. Hierdoor is change management absoluut essentieel. De algemene proces van het managen van changes noemen we *Configuration Management*.

**Versie management**

Versiebeheer houdt correcties/vernieuwingen in het software centraal, hierbij worden wijzigingen geplaatst in een centrale repository. Dit zorgt ervoor dat ontwikkelaars makkelijk kunnen samenwerken. Ontwikkelaars kunnen elkaars werk downloaden, wijzigingen inbrengen en vervolgens uploaden naar een nieuwe versie.

**System integration**

Als er onenigheden zijn in de ontwikkelde applicatie wil je de ontwikkelaarsteam hierbij niet betrekken. Je kan een eigen versie opslaan om dit probleem te omzeilen. Denk hier bijvoorbeeld aan versie x.x.1. We zullen git branch functionaliteit gebruiken om dit te doen.

**Problem tracking**

Eigen bug meldt functie, geïntegreerd in JIRA Software, zal gebruikt worden als basis voor probleem tracking. Bugs en probleem software helpt softwareontwikkelaars naar het vinden, opslaan en het volgen van bugs. Daarnaast kan het makkelijk gekoppeld worden aan de ontwikkelaar.

Opdracht 4.c

1. **De Process Measurement Attributes** van de pc-builder worden gemeten. Het is de bedoeling dat de maatregelen worden verbeterd volgens de doelstellingen van de organisatie die betrokken is bij de procesverbetering. We doen dit om te kijken of procesverbetering effectief is gebleken of niet.
2. **Process analysis –** Het huidige proces wordt beoordeeld en de gebrekkige processen en knelpunten worden geïdentificeerd. De procesmodellen die de werkwijze beschrijven kunnen in deze fase worden ontwikkeld.

3. **Process change -** Proces wijzigingen worden voorgesteld om geïdentificeerde zwakke punten aan te pakken. De cyclus wordt hervat en geïntroduceerd om gegevens te verzamelen over de effectiviteit van de veranderingen.

Evaluation and reflection 1 (Huseyin Aydogdu)

De stof is zeer leerzaam en makkelijk te volgen terwijl het op sommige punten wel veel thuisstudie vereist. Ik vond deze module uitdagend. Terwijl de nadruk was gelegd op de tijdsdruk vond ik dat het voor mij moeilijk in te schatten was hoeveel tijd eraan besteed moest worden, met name aan het einde van de module. De leerstof werd op sommige momenten door een aantal studenten gevolgd. Er was veel interactie binnen de groep, wat aangeeft dat de meeste de stof snapte. Het vragen werden goed beantwoord door de docent. De eindopdracht zou meer samenhang moeten hebben met andere lessen in het blok. Sommige studenten hadden dit niet begrepen en waren daardoor al gestart met een fictief project. Terwijl het veel leuker zou zijn als je de echte project zou gebruiken. Het gebruik van PowerPoint en video’s op scherm was goed, daarnaast is het gebruik van google classroom goed bevallen. We konden het lesmateriaal makkelijk vinden.

Evaluation and reflection (Mehmet Sozen)

Mijn eerste indruk van het stof was: Alweer een saai vak welke totaal niet relevant zal zijn in het zakenleven. Echter heb ik mij hierin zwaar vergist. De leerstof was ongelooflijk leerzaam en interessant. Met name hoe er interactief les werd gegeven en informatie werd verstrekt. In het begin vond ik het stof saai en veel. We moesten veel lezen en uitwerken. Naarmate er werd uitgewerkt volgens het boek begon het steeds interessanter te worden. In mijn huidige baan ben ik hierdoor anders gaan denken en doen door de leermaterialen. Ik merk hierdoor bijvoorbeeld op, dat ik in vergaderingen alles stap voor stap probeer te verwerken en te komen naar bepaalde oplossingen. Ik moet wel benadrukken dat het niet gaat om een software ontwikkel functie. Voor de rest was het interessant om informatie te vergaren in de les om iets te bouwen uit lego blokjes. Hierdoor bleef de motivatie er goed in. Ik moet wel eerlijk bekennen dat ik helemaal nieuw ben in Google Classroom en Google Docs. Dit is nog steeds even wennen voor mij omdat ik voorheen Word en Dropbox gebruikte. Hierdoor kan het zijn dat de layout van het verslag er een beetje slordig uit ziet. Lesmateriaal stond wel centraal en het was makkelijk terug te vinden. Al met al heb ik genoten van het vak INFANL01-5, Testen methoden en technieken.