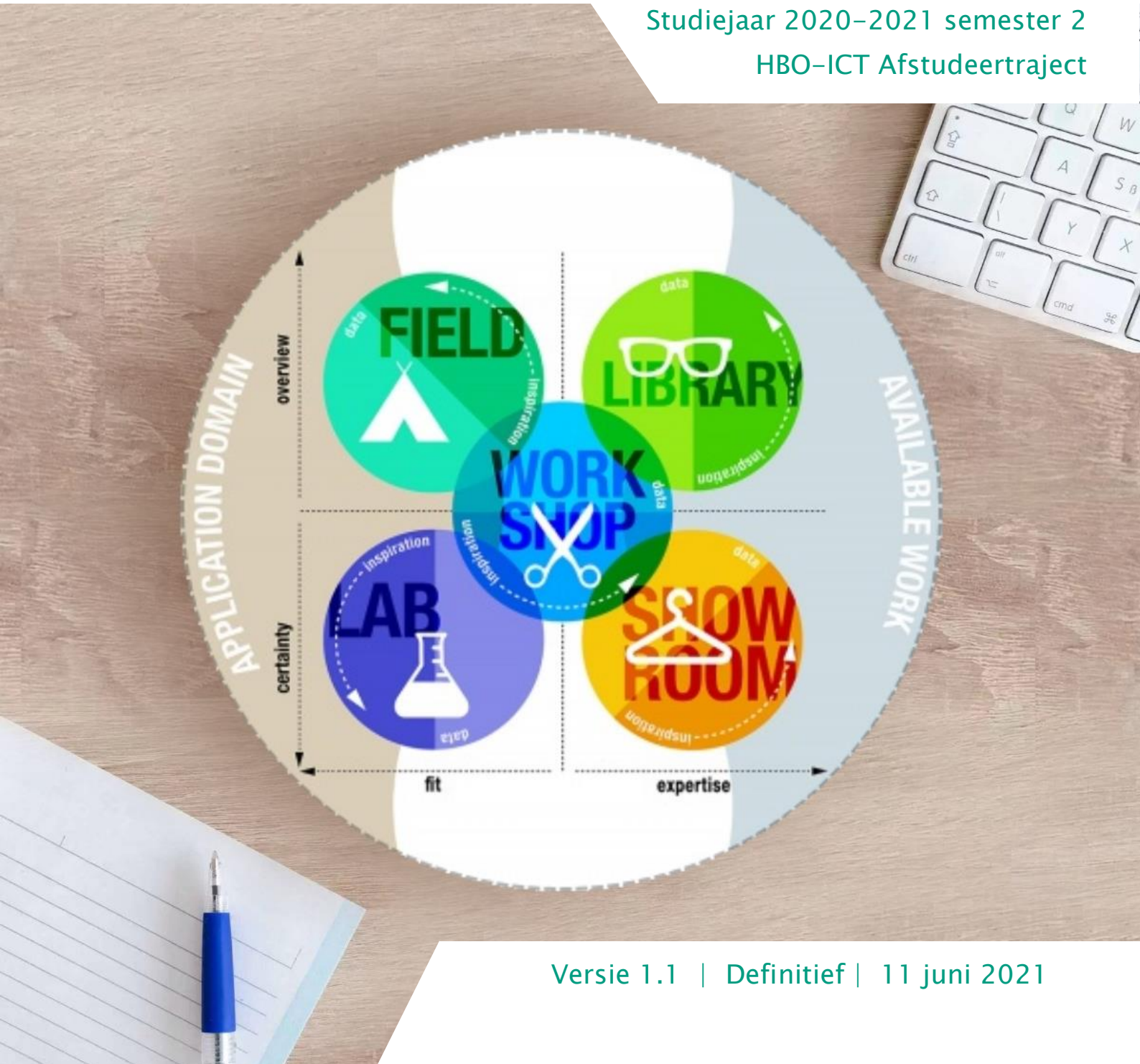
Afstudeerdossier

R.M. (Robin) van Alst – afstudeerder HBO-ICT Software Engineering

Studiejaar 2020-2021 semester 2

HBO-ICT Afstudeertraject



Versie 1.1 | Definitief | 11 juni 2021

Inhoudsopgave

1. Introductie

Het afstudeerproject betreft de ontwikkeling van een “Virtuele Docent” in de zogeheten “Project Approach Tool”. Dit is een webapplicatie van het lectoraat Ambient Intelligence van de Saxion Hogeschool. De Project Approach Tool biedt een visuele ontwerpomgeving waarin studenten samen een projectaanpak kunnen ontwerpen voor onderzoeks- en ontwikkelprojecten. Hierbij maken studenten gebruik van onderzoeksmethodes en activiteiten gebaseerd op de theorie van het “DOT Framework”. Dit is een theoretisch raamwerk van onderzoeksstrategieën en afwegingen om zo goed mogelijk onderzoek te doen. In het hoofdstuk Context vindt u een uitleg over deze en andere zaken die aan de afstudeeropdracht zijn gerelateerd.

De “Virtuele Docent” kan gebruikers van de Project Approach Tool begeleiden door hen feedback te geven over hun projectaanpak. Naast de virtuele docent zijn er drie kleinere, secondaire opdrachten om een aantal problemen aan te pakken in het huidige platform. Deze zijn minder complex dan de virtuele docent en dienen dus ter aanvulling op de hoofdopdracht.

In dit document beschrijf ik mijn project van start tot eind. Daarbij beschrijf ik zaken zoals:

- Context, probleemstelling, afstudeeropdracht & secondaire doelen;
- Methodiek om de afstudeeropdracht uit te voeren inclusief onderzoek, voortgangsbewaking en kwaliteitscontrole.
- Voorafgaand onderzoek, welke vragen zijn beantwoord en wat de resultaten van het onderzoek waren.
- Ontwerp van een gepaste oplossing, bestaande uit functioneel-, gebruikersinterface- en technisch ontwerp.
- Realisatie van de oplossing, inclusief enkele noemenswaardige oplossingen, en verantwoording van welke keuzes daarbij zijn gemaakt
- Testen van de oplossing, inclusief softwaretesten, UI testen en evaluatie van de bij het project betrokken stakeholders.
- Conclusies over de afstudeeropdracht.
- Samenvatting & advies over de voortzetting van het project.

Robin van Alst,

Afstudeerder, HBO-ICT Software Engineering

2. Context

Om de afstudeeropdracht goed te begrijpen, is het belangrijk om de context van de opdracht te kennen. In dit hoofdstuk worden een aantal belangrijke aanverwante zaken uitgelegd zoals de afstudeerorganisatie, Project Approach Tool en de theorie van het DOT Framework.

2.1. Afstudeerorganisatie

Ambient Intelligence (AmI) is een lectoraat van de Saxion Hogeschool. AmI focust zich op de "slimme wereld": het uitrusten van alledaagse dingen met slimme, innovatieve technologie. Concepten zoals Augmented - en Virtual Reality, Machine Learning, Artificial Intelligence en Internet of Things spelen hier vaak een rol bij.

Bij AmI werken ongeveer 24 vaste medewerkers van verschillende opleidingsniveaus en achtergronden. Veel van hen zijn daarnaast actief als docent bij HBO-ICT, CMGT en andere opleidingen van Saxion. AmI werkt veel samen met studenten van deze opleidingen om de onderzoeksvaardigheden van studenten te vergroten. Zo worden zij beter voorbereid op het professionele bedrijfsleven en eventuele (master) vervolgstudies. Projecten van AmI zijn voor studenten onder andere beschikbaar als Smart Solutions Semester en (afstudeer)stages.

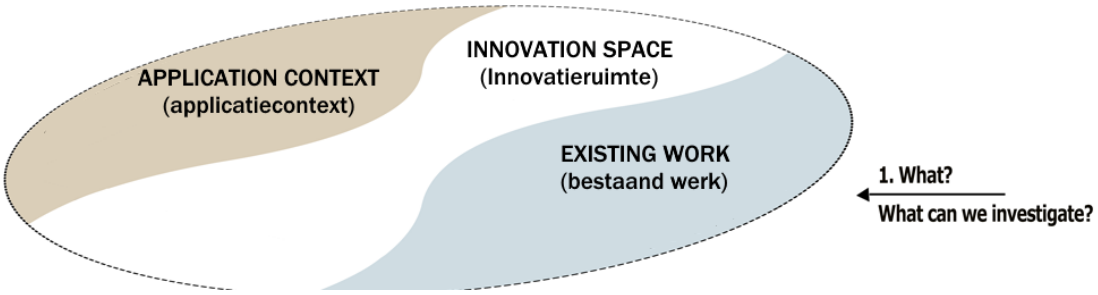
Bron: *Saxion Ambient Intelligence | Saxion Hogeschool (z.d.) | opgehaald van https://www.saxion.nl/onderzoek/smart-industry/ambient-intelligence*

2.2. DOT Framework

Het Development Oriented Triangulation (DOT) Framework is ontworpen door Dr. Koen van Turnhout (destijds docent bij de HAN) als raamwerk voor het uitvoeren van onderzoeks- en ontwikkelprojecten. Het DOT Framework bestaat uit drie kennisdomeinen, vijf onderzoeksstrategieën en drie afwegingen. Deze begrippen vormen een soort drielagenmodel.

Kennisdomeinen

Het DOT Framework kent drie kennisdomeinen: “applicatiecontext”, “bestaand werk” en “innovatieruimte”. De kennisdomeinen kan men zien als bronnen van verschillende soorten informatie die voor je project relevant kunnen zijn. Het domein “Applicatiecontext” bevat kennis over het probleem dat met je project wordt opgelost. Denk bijvoorbeeld aan stakeholders, gebruikers en de bestaande systemen die zij hebben. Daar tegenover staat het kennisdomein “Bestaand Werk”, dat meer algemene kennis, expertise en literatuur bevat. Tussen Applicatiecontext en Bestaand Werk zit het kennisdomein Innovatieruimte. Hier draag je zelf bij aan een oplossing.



Strategieën

Het DOT Framework kent vijf strategieën. Elke strategie heeft een bepaald focusgebied:

De veldstrategie: Ken je gebruiker.
Het veld dient om je gebruikers en toepassingscontext te verkennen.

De biebstrategie: op de schouders van reuzen.
De bieb dient om bestaand werk en expertise te verkennen.

De werkplaatsstrategie: Verbeter je oplossing.
De werkplaats dient om zelf te innoveren en proberen.

De labstrategie: Meten is weten.
Het lab dient om je oplossing te testen, meten en verbeteren.

De showroomstrategie: Toon je waarde.
De showroom dient om de waarde en kwaliteit van je oplossing te toetsen.

2.3.Afwegingen

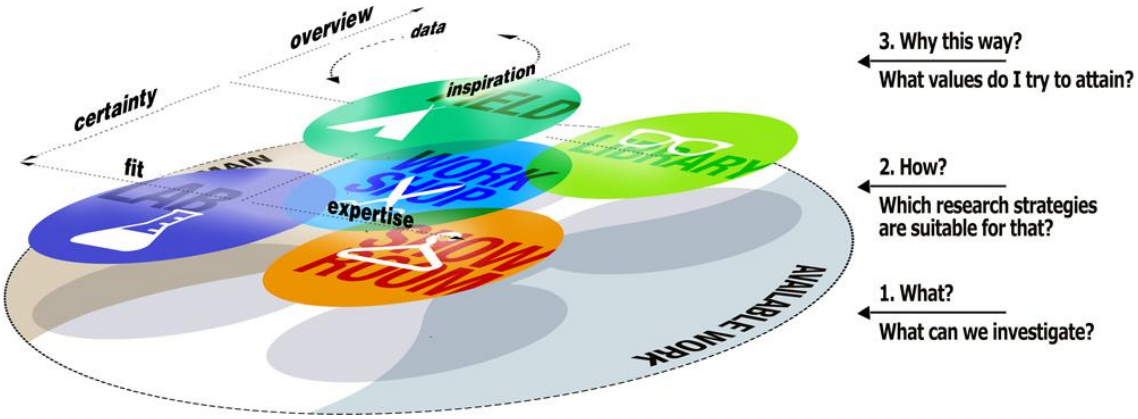
Als laatste kent het DOT Framework drie afwegingen:

- Overzicht of zekerheid?
- Expertise of fit?
- Data of inspiratie?

Overzicht is belangrijk om vooraf je probleem te definiëren en een oplossingsrichting te bepalen. Zekerheid is juist belangrijk om te verifiëren dat je oplossing goed werkt, van goede kwaliteit is en het probleem daadwerkelijk oplost.

Expertise is belangrijk om te zorgen dat je oplossing voldoet aan geldende kwaliteitsstandaarden. Fit wil zeggen dat je oplossing goed aansluit op de toepassingscontext, je gebruikers en het vakinhoudelijk domein dat je bedient.

Data dient met name om je keuzes te onderbouwen met feitelijke gegevens, onderzoek en observaties. Data is echter niet geschikt voor *elke* keuze: soms kan inspiratie helpen om een innovatieve oplossing te realiseren. Samengevoegd vormen de kennisdomeinen, strategieën en afwegingen een drielagenmodel. In de onderstaande figuur ziet u hoe het samen één geheel vormt. Let op de positie van de vijf onderzoeksstrategieën ten opzichte van de kennisdomeinen en afwegingen. Een uitvergrote versie is te vinden in 0.



Figuur 1: Het volledige DOT Framework drielagenmodel. Zie Bijlage 1 voor een uitvergrote versie.

2.4. Kaartensets & Project Approach Tool

Voor studenten zijn een aantal fysieke kaartensets beschikbaar waarmee zij een projectaanpak kunnen bepalen. Deze kaartensets zijn gebaseerd op het DOT Framework en bevatten voor elke onderzoeksstrategie een aantal onderzoeks-methodes zoals interviews, unit testing en literatuuronderzoek. Er zijn kaartensets met methodes voor ICT, Creative Media Design en Engineering. Daarnaast is er de "Stepping Stones" kaartenset die o.a. deelresultaten bevat voor een projectaanpak.

Project Approach Tool

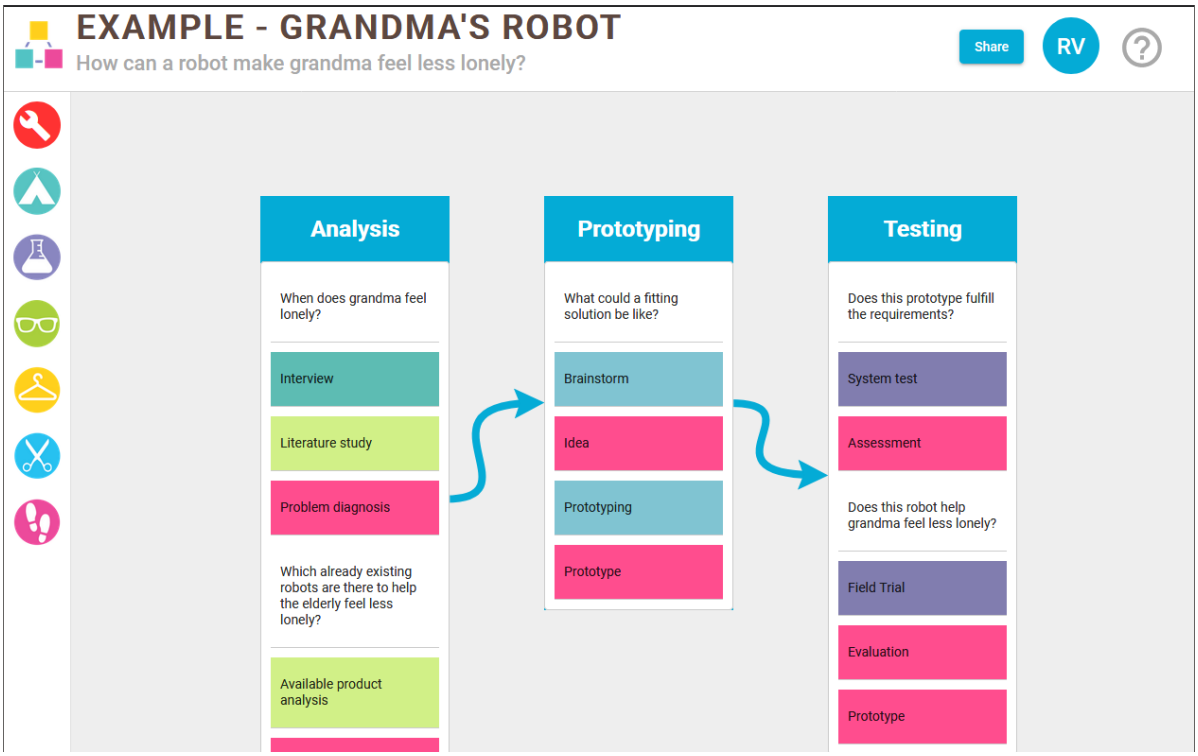
In de Project Approach Tool kunnen studenten in een visuele omgeving samen een projectaanpak ontwerpen. Hierbij hebben zij beschikking over digitale versies van de hiervoor genoemde kaartensets. De Project Approach Tool is op de eerste plaats bedoeld als brainstorming hulpmiddel zodat studenten alleen of samen een concept voor hun projectaanpak kunnen maken. De Project Approach Tool is een bestaand platform en kan worden geraadpleegd via de onderstaande link:

Open Project Approach Tool
http://www.projectapproachtool.nl

De Project Approach Tool is ontwikkeld door als digitaal hulpmiddel om een projectaanpak te ontwerpen. PAT maakt de hiervoor genoemde kaartensets digitaal beschikbaar. Daardoor is PAT geschikt voor wanneer fysieke aanwezigheid niet kan, zoals tijdens de COVID-19 pandemie op het moment van schrijven.

- **Stepping Stones**: deze kaartenset bevat in PAT alle deliverables die als resultaat uit een onderzoeksvraag naar voren zouden kunnen komen.
- CMD Research Methods: deze kaartenset bevat onderzoeksactiviteiten op basis van het DOT Framework, die zijn toegespitst op het vakgebied Communication & Multimedia Design.
- Engineering Methods Pack: soortgelijk als de CMD kaartenset, maar dan voor het vakgebied Engineering.
- ICT Research Methods pack: soortgelijk als de CMD- en Engineering kaartensets, maar dan voor het vakgebied ICT.
- Eenmaal geregistreerd, kunnen studenten in de PAT een "workspace" aanmaken op basis van templates zoals Design Thinking. In zo'n workspace kunnen studenten en projectaanpak ontwerpen door projectfases, onderzoeksvragen, activiteiten en deelresultaten toe te voegen. Dit kunnen zij ook in samenwerking doen: alle wijzigingen worden dan automatisch gesynchroniseerd met leden van de workspace.

In de onderstaande figuur ziet u een schermafdruk van een workspace in de Project Approach Tool zoals deze was bij aanvang van het afstudeersemester. Verderop in dit document vindt u meer schermafdrukken die de gemaakte wijzigingen laten zien.



Figuur 2: Een Workspace in de Project Approach Tool.

3. Probleemstelling

Het doel van de Project Approach Tool is om studenten een omgeving te bieden waarin zij een projectaanpak kunnen ontwerpen. Daarbij willen we ze laten nadenken over hun projectplan en hoe zij invulling en follow-up gaan geven aan de activiteiten die ze willen uitvoeren. Dit doel wordt in de huidige versie echter in de weg gezeten door een aantal problemen, namelijk: een gebrek aan begeleiding en een paar onhandige interacties in de UI (gebruikersinterface).

3.1. Primair probleem: Virtuele Docent

Studenten die gebruik maken van de Project Approach Tool, hebben mogelijk weinig tot geen ervaring met het DOT Framework en de gerelateerde kaartensets. De Project Approach Tool biedt echter geen begeleiding. Hierdoor ervaren studenten een gebrek aan feedback en maken ze veel voorkomende fouten, bijvoorbeeld dat ze deel-resultaten vergeten of te zwaar inzetten op één onderzoeksstrategie. De wens is om een feedbackmechanisme te ontwikkelen dat onervaren studenten kan begeleiden om meer bewuste keuzes te maken over hun projectaanpak.

3.2. Secondair probleem: UI verbeteringen

Ter aanvulling op de primaire opdracht, waren er drie UI problemen die moesten worden opgelost in de huidige versie van de Project Approach Tool. Omdat dit aanvullende problemen zijn, is de voltooiing ervan is niet vereist om met succes te afstuderen.

Wanneer je in de Project Approach Tool inzoomt, worden de verbindingspijlen tussen projectfases niet goed uitgelijnd met de verbonden projectfases. Het resultaat ziet er raar uit: pijlen wijzen van / naar een leeg gebied of overlappen met de projectfases.

Nadat een verbindingspijl is geplaatst, is deze ook permanent verbonden aan twee projectfases. Het is niet mogelijk om de pijl los te maken en aan een andere fase te verbinden. Het is wel gewenst om dit te kunnen.

Als laatste is het huidige proces om een onderzoeksmethode toe te voegen, niet optimaal. De wens is om dit via een klik-en-sleep interactie te kunnen doen. Vanwege technische beperkingen is dit systeem echter nog niet gerealiseerd.

4. Methodiek

Bij het afstuderen komt behoorlijk wat kijken. Om het project in goede banen te leiden, zijn voor een aantal zaken zoals kwaliteits- en voortgangsbewaking bepaalde methodieken gebruikt. Deze worden in dit hoofdstuk toegelicht.

4.1. Onderzoek

Het afstudeerproject is direct verwant aan het DOT Framework. De logische keuze is om zelf ook een onderzoeksplan te maken op basis hiervan. In Bijlage 2 vindt u een diagram van de gehanteerde projectaanpak. Voor de projectfasering is de Design Thinking methodiek gekozen; dit sluit uitstekend aan bij abstracte problemen zoals de Virtuele Docent (Buchanan, 1992). In de praktijk betekent dit, dat ongeveer de eerste helft van het afstudeersemester is besteed aan voorbereidend onderzoek.

Het doel van dit voorafgaand onderzoek is ten eerste om het probleem te definiëren. “Veel voorkomende feedback geven” is een vaag en abstract probleem. Om hier meer concrete betekenis aan te geven, zijn een aantal docenten geïnterviewd over veel voorkomende feedback en valkuilen. Deze interviews kan men beschouwen als gebruikersinterviews en expert interviews (respectievelijk veld- en biebstrategie).

Het tweede doel van het voorafgaand onderzoek is om een oplossingsrichting te bepalen zodat studenten het systeem goed kunnen gebruiken. Hiervoor is onder andere een UI-ontwerpproces uitgevoerd met enkele usability tests (lab strategie).

Onderzoeksvragen

Hoofdvraag: Hoe kunnen we studenten helpen om meer bewuste keuzes te maken over hun projectaanpak?

1. **Wat is "veel voorkomende feedback" dat door docenten wordt gegeven?**
   Het doel is om "veel voorkomende feedback" te definiëren via een aantal interviews en op basis daarvan een set van requirements te bepalen.
   Deelresultaat: begrip over veel voorkomende feedback & aspecten daarvan
2. **Hoe kan Project Approach Tool studenten begeleiden tijdens het maken van hun projectaanpak?**
   a. **Hoe werken andere, bestaande systemen op het gebied van feedback, begeleiding en visuele vormgeving?**
      Het doel is om ter inspiratie een aantal diverse, bestaande producten te analyseren. Dit zijn niet alleen feedbacksystemen, maar ook systemen die een specifieke syntax / model afdwingen of vergelijkbare functies hebben.
      Deelresultaat: analyse van bestaande systemen & inspiratie voor UI
   b. **Is het mogelijk om "veel voorkomende feedback" te automatiseren?**
      Het doel is om te bepalen of / hoe een automatisch feedbacksysteem technisch haalbaar is. Om deze vraag te beantwoorden is de feedback uit de interviews kritisch geanalyseerd.
      Deelresultaat: lijst met requirements
   c. **Wat voor andere begeleiding kan Project Approach Tool geven om deze valkuilen te vermijden?**
      Het doel is om te bepalen of alternatieve begeleidingsmiddelen haalbaar en zinvol zijn, op basis van de huidige requirements en bestaande producten analyse.
      Deelresultaat: lijst met requirements
   d. **Hoe kan het feedbacksysteem technisch worden geïmplementeerd?**
      Het doel is om een technisch ontwerp te bepalen ter voorbereiding op de ontwikkeling van het feedbacksysteem. Om deze vraag te beantwoorden is een functioneel en technisch ontwerp gemaakt.
      Deelresultaat: functioneel & technisch ontwerp
   e. **Hoe kan de gebruikersinterface worden ontworpen?**
      Het doel is om een iteratief UI / UX ontwerp van het feedbacksysteem te maken dat met een paar testgebruikers is afgestemd. Hierbij is de bestaande productenanalyse gebruikt als inspiratiebron.
      Deelresultaat: functioneel & UI ontwerp

4.2. UI ontwerp

UI ontwerp hoort niet vanzelfsprekend bij de kerncompetenties van HBO-ICT Software Engineering. Veel software engineers hebben weinig kennis over het ontwerpen van een goede gebruikerservaring. In mijn derde - en vierde studiejaar heb ik de kans gekregen om ervaring op te bouwen met UI en UX ontwerp. Daaruit is een interesse voor het vakgebied voortgekomen. Daarom heb ik gekozen om naast de kerncompetenties van Software Engineering ook UI ontwerp mee te nemen op HBO-I niveau 2 (zie hoofdstuk Ontwerp: Gebruikersinterface).

Afbeelding © EEWeb.com (USA) - Fair use voor educatieve doeleinden.

4.3. Iteratieve werkwijze

Bij de uitvoer van het project is de Design Thinking methodiek gebruikt. Hoewel de projectaanpak in Bijlage 2 wellicht anders doet denken, is Design Thinking geen lineair proces zoals het bekende Watervalmodel. In de praktijk is het project in een aantal parallelle iteraties uitgevoerd. De ontwikkeling van het feedbacksysteem is op een Agile achtige manier ontwikkeld in iteraties van 2 weken. (zie volgende paragraaf). Het UX ontwerpproces verliep parallel aan deze sprints in twee iteraties van één week. Meer daarover is te lezen in hoofdstuk 4.2 UI ontwerp.

Figuur 3: Design Thinking methodiek (vereenvoudigde weergave)

4.4. Voortgangsbewaking

Om de voortgang van het project te bewaken, zijn twee keuzes gemaakt.

Ten eerste wordt het project in een Agile-achtige methodiek uitgevoerd. Dat wil zeggen dat in sprints wordt gewerkt om iteraties van het product te ontwikkelen. Teamaspecten van Agile / Scrum zoals daily stand-ups zijn moeilijk door één persoon te doen en zijn daarom achterwege gelaten. Wel wordt bij elke sprint even kort gekeken wat er beter kan, ter vervanging van een conventionele sprint retrospective.

Ten tweede is een omgeving aangemaakt in Jira, een webapplicatie van de firma Atlassian dat uitermate geschikt is voor softwareontwikkeling volgens de Agile methodiek. Hierin worden alle productvereisten ("requirements") bijgehouden en ingepland in sprints. De requirements zijn in vrije vorm geformuleerd omdat dit voor het beoogde doel (ontwikkeling van een feature) duidelijk genoeg was.

User story formats zoals *"als <persoon> wil ik <functie> omdat <reden"* zijn ook overwogen. Gezamenlijk begrip is een van de kernvoordelen van user stories. Dergelijke formats als die hierboven komen het beste tot hun recht in teamprojecten met verschillende rollen, disciplines en stakeholders door een consequent en eenvoudig format te introduceren dat iedereen begrijpt. Gezien het afstudeerproject een éénpersoonsproject betreft, is dit format uiteindelijk niet toegepast, omdat het naar verwachting weinig extra voordeel zou hebben.

4.5. Kwaliteitsbewaking

De afstudeeropdracht betreft de ontwikkeling van oplossing voor een complex, abstract vraagstuk: het maken van een virtuele docent. Om de kwaliteit van de oplossing te bewaken, worden een aantal maatregelen genomen.

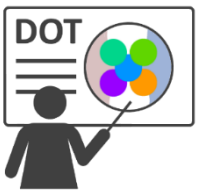
1. Voorafgaand aan het bepalen van de requirements worden docenten geïnterviewd om een beeld te krijgen van wat "veel voorkomende feedback" is.
2. Om te zorgen dat de gebruikersinterface (UI) voor de doelgroep makkelijk in het gebruik is, wordt vooraf een klikbaar UI prototype gemaakt als onderdeel van een UI ontwerpproces en d.m.v. usability tests geëvalueerd met studenten.
3. Studenten die hebben meegewerkt aan usability tests van punt 2 worden regelmatig betrokken bij de ontwikkeling om proactief feedback te krijgen.
4. Tijdens ontwikkeling wordt de virtuele docent voortdurend getest door middel van "*development testing*" praktijken zoals debuggen, smoke- en sanity testing. Geïdentificeerde fouten worden tijdens het ontwikkelproces opgelost.
5. Zodra de virtuele docent is voltooid, zal een systeemtest worden gedaan om de werking van de virtuele docent te controleren op fouten en/of ongemakken.
6. Afsluitend worden betrokken docenten uitgenodigd voor een product reviewsessie waarin zij een laatste kans krijgen om feedback te geven.

5. Empathize & Define

Voordat een oplossing kan worden bepaald, moet men bekend zijn met het probleem. Bij Design Thinking gebeurt dat in de Empathize – en Define fases. Hoewel dit formeel gezien twee afzonderlijke fases zijn, kunnen ze in de praktijk soms in elkaar overvloeien als ware het één fase. Zo ging het ook bij de virtuele docent. Ongeacht de praktische uitvoer van de fases, was het doel om het probleem te definiëren om een oplossingsrichting te bepalen.

Studenten

Studenten ervaren een gebrek aan feedback en kritiek in de Project Approach Tool om de juiste afwegingen te maken. Daarom schakelen ze vaak de hulp in van hun eigen docenten, die niet gelijk tijd hebben.



Docenten

Docenten ervaren dat zij vaak dezelfde feedback geven. Ook kunnen ze niet goed diepgaande vragen stellen omdat de projectaanpak nog te veel oppervlakkige of structurele problemen bevat.

Beoogde oplossing

Om studenten te helpen meer bewuste keuzes te maken, is de wens van de afstudeerorganisatie om een feedbacksysteem te maken dat veel voorkomende feedback van docenten kan automatiseren. Maar hoe doe je dat eigenlijk? Om deze vraag te beantwoorden, is een voorafgaand onderzoek uitgevoerd bestaande uit een aantal deelvragen (zie hoofdstuk 4.1). Een cruciaal onderdeel van dat onderzoek waren een aantal interviews met docenten. Deze docenten hebben ervaring in het lesgeven met het DOT Framework.

5.1. Interviews

“Veel voorkomende valkuilen”… Hoe vraag je dat op zo’n manier dat je waardevolle informatie krijgt? Om de interviews te begeleiden, zijn voorafgaand aan de interviews een aantal voorbeelden van projectaanpakken gemaakt met bepaalde thema’s. Elk thema behandelt een specifiek aspect van het DOT Framework, bijvoorbeeld een zware focus op de afweging Overzicht, of de Biebstrategie.

Tijdens de interviews hebben de docenten de voorbeelden bekeken en er feedback op gegeven. Zo is een overzicht gemaakt van wat elke docent belangrijk vindt in een projectaanpak. De informatie van de interviews is gebruikt om de “requirements” van de virtuele docent te bepalen *(wat moet de virtuele docent kunnen en waar moet het aan voldoen?)*.

De interviewkandidaten zijn strategisch geselecteerd op basis van de hogeschool en opleiding waaraan zij lesgeven. Het doel was om een goede representatie te krijgen van algemene feedback, in plaats van feedback “op de Saxion manier”. Naast twee docenten van Saxion zijn daarom twee docenten van andere hogescholen geïnterviewd.

- Etto Salomons – Saxion HBO-ICT
- Koen van Turnhout – HAN & Hogeschool Utrecht; auteur van DOT Framework
- Floor Weijman – Saxion HBO-ICT
- Sonja Rouwhorst – Hogeschool van Amsterdam

5.1.1. Etto Salomons

Etto Salomons is een docent van Saxion HBO-ICT. Tijdens het interview legde hij uit dat het DOT Framework bij HBO-ICT niet veel wordt gebruikt. Daardoor merkt hij dat begrippen zoals “Triangulatie” (Wikiwijs / Saxion, 2020) voor studenten onbekend zijn en dat zij moeite hebben om invulling en planning te geven aan hun. Netto vindt het belangrijk om voldoende achtergrondinformatie te verzamelen, bijvoorbeeld literatuuronderzoek en analyse van bestaand werk. Ook is hij kritisch over de formulering van onderzoeksvragen, bijvoorbeeld of deze goed passen bij de doelstelling. Etto is sceptisch over het automatiseren van controles op de balans tussen onderzoeksstrategieën en de momenten dat deze worden ingezet, omdat dit naar zijn mening afhankelijk is van het project.

5.1.2. Koen van Turnhout

Als ontwerper van het DOT Framework en docent bij de hogeschool HAN heeft dr. Koen van Turnhout veel ervaring met het geven van feedback op projectaanpakken. Hij geeft direct aan dat er grote verschillen zijn tussen kennis en ervaringsniveau van studenten. Hij pleit daarom voor het geven van informatie over onderzoeksmethodes en strategieën ter aanvulling op het geven van feedback. Hij merkt op dat studenten moeite hebben om een balans te vinden tussen aantal en diversiteit van onderzoeksmethodes. Triangulatie speelt volgens hem een cruciale rol. Net als Etto Salomons benadrukt dr. van Turnhout het belang van achtergrondinformatie. Als laatste vindt hij het belangrijk dat, naast triangulatie, de gebruikte onderzoeks-strategieën in balans zijn zonder zware focus op het een of ander.

5.1.3. Floor Weijmans

Floor Weijmans is net als Etto Salomons een docent van Saxion HBO-ICT. Ze heeft naar eigen zeggen ongeveer 3 jaar ervaring met het gebruik van het DOT Framework bij Saxion. Bij Smart Solutions bijvoorbeeld ziet ze dat studenten moeite hebben om een compromis te sluiten over de te hanteren aanpak vanwege de multidisciplinaire aard van Smart Solutions. Ook wordt probleemanalyse volgens haar niet grondig uitgevoerd, waarbij vragen zoals “wat ga je oplossen” en “waarom wil je dit oplossen” niet volledig worden beantwoord of onderbouwd. Floor benadrukt het belang van literatuuronderzoek en de biebstrategie in het algemeen, wat ze beschrijft als een “must have”. Triangulatie is volgens haar ook belangrijk, waarin ze zelfs wat verder gaat dan Koen van Turnhout door te pleiten voor drie databronnen per fase. Een kort debat over de rol van “requirements” heeft benadrukt dat zij dit ook een cruciaal onderdeel van een projectaanpak vindt.

5.1.4. Sonja Rouwhorst

Sonja Rouwhorst is een docent van de Communication & Multimedia Design aan de Hogeschool van Amsterdam. Van alle interviewkandidaten was zij verreweg het meest sceptisch over het automatiseren van feedback. Al snel bleek dat er bij de CMD opleiding een totaal andere mindset heerst over het maken van een projectaanpak, die deels conflicteert met de feedback van de hiervoor geïnterviewde docenten.

Net als Etto Salomons benadrukte mevr. Rouwhorst dat er geen “goed” of “fout” is en dat keuzes sterk afhankelijk zijn van het project. In tegenstelling tot de vorige docenten, die aangaven dat zij veel letten op triangulatie, balans tussen strategieën en voldoende onderzoeksmethoden wordt hier bij CMD pragmatischer mee omgegaan. Wel vond Mevr. Rouwhorst dat het in het algemeen niet verstandig is om op één onderzoeksstrategie te focussen en dat het belangrijk is om onderzoeksvragen concreet, doelgericht en open te formuleren (geen ja/nee vraag). In het interview deed mevr. Rouwhorst een aantal suggesties voor alternatieve functionaliteiten om het feedbackproces te vereenvoudigen, zoals een “feedbacklegenda” (Peeters, 2019).

Interviews in coronatijd

Sinds maart 2020 woedt een wereldwijde uitbraak van het COVID-19 coronavirus. Ten tijde van de interviews greep deze ziekte nog volop om zich heen. Daarom zijn alle interviews online afgenomen via Microsoft Teams

5.2. Requirements

Uit de interviews bleek dat de docenten vaak ongeveer dezelfde opvattingen hadden over de voorgestelde thema's. Er waren echter wel subtiele verschillen --- de ene docent vindt bijvoorbeeld een goede mix van strategieën belangrijker dan de andere. Aan de virtuele docent zijn een aantal belangrijke hoofdrequirements gesteld.

In de (externe) meegeleverde Excel-werkmap *"comparing feedback.xlsx"* vindt u een overzicht van de uiteindelijke requirements en hoe deze zijn bepaald. De requirements hierin zijn volgens MoSCoW (Must, Should, Could, Would / Won't) geprioriteerd en gecategoriseerd als zijnde Functioneel / Non-functioneel. Bij uitvoer van het project zijn deze requirements in digitale vorm bijgehouden in een (privé) backlog op Atlassian Jira.

1) De Virtuele Docent moet automatisch feedback kunnen leveren om veel voorkomende fouten af te vangen. Dit is belangrijk om studenten meer te kunnen begeleiden bij het maken van een projectaanpak.
    a) Bij feedback dat algemeen van toepassing is, moet de virtuele docent dit kunnen aangeven met uitleg zodat de student het probleem kan oplossen. Dit is belangrijk om veelvoorkomende algemene fouten te voorkomen.
    b) Bij feedback dat sterk afhankelijk is van het project, moet de virtuele docent genoeg informatie kunnen aanleveren zodat studenten en docenten zelf een goed geïnformeerde conclusie kunnen trekken. Dit is belangrijk, want niet alle feedback kan worden geautomatiseerd omdat PAT belangrijke informatie mist over de context van een project.
2) De feedback die de virtuele docent levert, moet instelbaar zijn. Dit is belangrijk om een flexibel systeem in te richten dat aanpasbaar is aan de veranderende mindset over het gebruik van het DOT Framework.
    a) Alleen docenten mogen de globale feedbackregels van de Virtuele Docent beheren. Dit is belangrijk omdat studenten dit niet behoren te kunnen.
    b) Het inrichten van een volledig docentenportaal / backoffice in de Project Approach Tool valt buiten de scope van het afstudeerproject. Dit valt buiten de scope omdat het daarvoor te uitgebreid en complex is t.o.v. de huidige versie van PAT.
3) Studenten kunnen de feedback van de virtuele docent aan/uitzetten. Dit is belangrijk zodat studenten kunnen kiezen wat voor feedback zij zullen krijgen.
    a) Studenten kunnen de virtuele docent in geheel aan/uitzetten. Dit is belangrijk voor studenten die al ervaren genoeg zijn dat de feedback hen niet kan helpen.

b) Studenten kunnen aangeven wat voor feedback zij willen krijgen. De virtuele docent zal vervolgens alleen dat soort feedback geven. Dit is belangrijk om te zorgen dat studenten feedback krijgen die zij op dat moment nodig hebben.

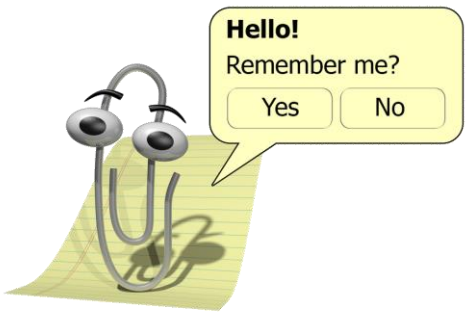
c) Studenten kunnen individuele feedback persistent aan/uitzetten als dit voor hun project niet van toepassing is. Dit is belangrijk omdat bepaalde "algemene" feedback wellicht alsnog conflicteert met bepaalde projecten.

4) De virtuele docent moet makkelijk zijn in het gebruik. Voor studenten is het al lastig genoeg om bewuste keuzes te maken en het DOT Framework te begrijpen. Het is niet de bedoeling dat de Virtuele Docent ze daarbij in de weg zit met een onhandige gebruikersinterface.

6. Ontwerp: Functionaliteit & use cases

Het functioneel ontwerp beschrijft welke functionaliteit de virtuele docent zal ondersteunen en hoe het deze functionaliteit aan de gebruiker beschikbaar stelt. Standaard onderdelen van een functioneel ontwerp zijn: Use cases diagrammen met beschrijvingen en "mockups" of "wireframes" van de gebruikersinterface (UI). De virtuele docent kent een aantal use cases. Alle use cases die door de externe actoren wordt uitgevoerd, beïnvloeden in zekere mate het interne feedbacksysteem. Dit is te zien in Bijlage 4, waarin u een overzicht kan vinden van alle use cases.

Wanneer studenten iets wijzigen aan de projectaanpak, wordt automatisch feedback en statistieken gegenereerd. Studenten en docenten kunnen de virtuele docent op een aantal manieren naar eigen hand zetten, zoals het aan- of uitzetten van specifieke feedbackregels of het kiezen van een "modus" van de virtuele docent. Zo kunnen studenten en docenten zelf kiezen wat voor feedback het systeem zal leveren. Het moet natuurlijk worden voorkomen dat het systeem de student in de weg zit als een soort moderne variant van Clippy, de notoir irritante digitale assistent van Microsoft Office 2000.



Afbeeldingsbron: oginundercover (2015), DeviantArt. Fair use voor educatieve doeleinden.
https://www.deviantart.com/oginundercover/art/Clippit-Office-Assistant-535224892

Een deel van de feedback die uit de interviews naar voren kwam, kan worden geautomatiseerd. Andere feedback daarentegen is te abstract of ambigu om goed in concrete regels om te zetten. Daarom zijn een paar aanvullende hulpmiddelen bedacht die meer inzicht kunnen geven in situaties waar concrete feedback niet haalbaar is.

1. **Triangulatie:** Het concept van Triangulatie (Wikiwijs / Saxion, 2020) is volgens de geïnterviewde docenten een belangrijk aspect van een goede project-aanpak. Maar de gewenste mate van Triangulatie is sterk afhankelijk van het project. Om toch een beeld te geven van de mate van Triangulatie, kan de Virtuele Docent een statistisch overzicht berekenen van de vertegenwoordiging van alle afwegingen van het DOT Framework. Deze statistieken stellen studenten en docenten in staat om daar zelf conclusies over te trekken.
2. **Focus op bepaalde strategieën:** Net als Triangulatie is ook het gebruik van bepaalde strategieën een belangrijk feedbackpunt van de geïnterviewde docenten. Hier geldt echter ook dat er geen sprake is van “goed” of “fout”. Daarom kan de Virtuele Docent net als bij Triangulatie berekenen in hoeverre de gebruiker focust op de strategieën van het DOT Framework.

Als laatste kunnen alle vooraf gedefinieerde feedbackregels per projectaanpak worden aan- of uitgezet. In het geval dat een van de ingestelde feedbackregels niet relevant is voor het betreffende project, kunnen gebruikers deze regel uitzetten. Dit kan op het moment dat de feedback wordt gegeven, maar ook vanuit een instellingenmenu met zoek- en filteropties. In dit menu kan ook de Virtuele Docent geheel worden aan- of uitgezet. De Virtuele Docent staat standaard aan, maar is dus wel persoonlijk in te stellen. Zo kunnen studenten zelf bepalen of zij feedback willen krijgen en zo ja, wat voor soort feedback zij willen krijgen.

6.1. Bestaande productenanalyse

Om inspiratie op te doen voor de gebruikersinterface, is vooraf een analyse uitgevoerd van een aantal bestaande producten. Er is gezocht naar meer specifieke softwareoplossingen die beter aansluiten op de specifieke vraagstelling van de virtuele docent. Echter, vanwege de nieuwigheid van de virtuele docent zijn dergelijke systemen zeer moeilijk te vinden. Daarom is de uiteindelijke selectie meer divers dan aanvankelijk gewenst was. Deze selectie is: Microsoft Word, Microsoft Visual Studio, Unreal Engine 4 en de computergame Factorio. Hoe hebben deze product bijgedragen aan de Virtuele Docent?

Het documenten bewerkingsprogramma Microsoft Word was een grote inspiratiebron voor een aantal functionele zaken zoals het aan/uitzetten van feedbackregels. Bij spelfouten kun je in Word aangeven dat je een fout niet opnieuw wil zien. Ook geeft Word een aantal opties voor correcties. Dit heeft tevens bijgedragen aan het idee van secondaire feedback acties in de Virtuele Docent, zoals het openen van een catalogus-tab of een externe webpagina.

Microsoft Visual Studio is de de-facto standaard software-ontwikkelomgeving voor het Microsoft .NET Framework en .NET Core. Visual Studio heeft met name bijgedragen aan de manier waarop feedback wordt gegenereerd en teruggekoppeld. Dit eerste gebeurt meestal “contextueel”. Dat wil zeggen dat het programma alleen feedback levert op de plek waar je aan het werk bent. Bij een volledige “build” geeft Visual Studio meer feedback over het hele project. Dit ligt aan de basis van het idee van een contextuele analyse in de Virtuele Docent.

Unreal Engine 4 (UE4) is een videogame engine geproduceerd door Epic Games, bekend van games zoals Fortnite en Unreal Tournament. UE4 heeft een innovatief systeem voor visueel scripting genaamd “Blueprints”. De Blueprint editor omgeving gebruikt een aantal visuele trucs om bepaalde informatie te communiceren naar de ontwikkelaar zoals de aanwezigheid van ”breakpoints”, type van variabelen en functies en een paar andere handige hulpmiddelen. UE4 was een grote inspiratiebron voor de grafische vormgeving van feedbackmeldingen en heeft tevens bijgedragen aan het bepalen van aanvullende functies van de Virtuele Docent.

Als laatste is de game Factorio geanalyseerd --- dit lijkt in eerste opzicht een vreemde keuze. Wat heeft een spel over het bouwen en verdedigen van een kolossale fabriek op een alien-planeet in vredesnaam te maken met een feedbacksysteem voor studenten? Factorio bevat een aantal complexe systemen zoals “circuit logica” en het automatiseren van vrachttreinen. Voor dit laatste gebruikt Factorio, net als de Virtuele Docent, een modulair regelsysteem. De manier waarop je als speler eenvoudig regels kan toevoegen, verwijderen en bewerken om treinen te laten wachten bij specifieke stations, was een grote inspiratiebron voor het modulaire regelsysteem van de Virtuele Docent.

Een meer uitgebreid verslag van de analyse van bestaande producten is te vinden in de (externe) bijlage *“bestaande productenanalyse.pdf”* . Deze bijlage is te vinden in de folder UI ontwerp. Als u geïnteresseerd bent in meer informatie, wordt ten sterkste aangeraden om deze bijlage door te lezen.

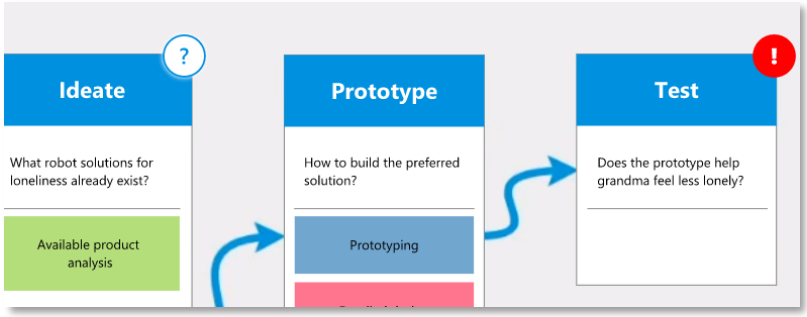
7. Ontwerp: Gebruikersinterface

Het UI ontwerp betreft een klikbaar UI-prototype gemaakt in het programma Lunacy (Icons8, 2021). Dit ontwerp is in een paar iteraties van één week ontwikkeld. Om het gemaakte UX ontwerp te testen en verbeterpunten te identificeren, zijn per iteratie van het ontwerp een paar usability tests uitgevoerd. Hiervoor zijn proefpersonen uitgenodigd uit de doelgroep van studenten. Tijdens de tests hebben zij simpele instructies gekregen zonder hen exact te vertellen waarop te klikken (bijvoorbeeld “zet de virtuele docent in de Editing modus”). Vervolgens is gelet op hun reactie: konden ze de instructies snel en makkelijk uitvoeren, of zaten ze te zoeken naar de betreffende knop? Tussendoor zijn een aantal kritische vragen gesteld (bijv. “wat vind je van XXX?” of “viel het je op dat YYY”). Op die manier is bepaald wat er beter kon aan het UX ontwerp.

In totaal zijn 2 volledige iteraties van het UX proces uitgevoerd. Omdat de feedback op de tweede iteratie al zeer positief was, is de derde iteratie meer ad-hoc geëvalueerd met de proefpersonen tijdens ontwikkeling van de daadwerkelijke gebruikersinterface. Wat de resultaten waren van elke iteratie, leest u in de komende paragrafen.

7.1. UI ontwerp, versie 1

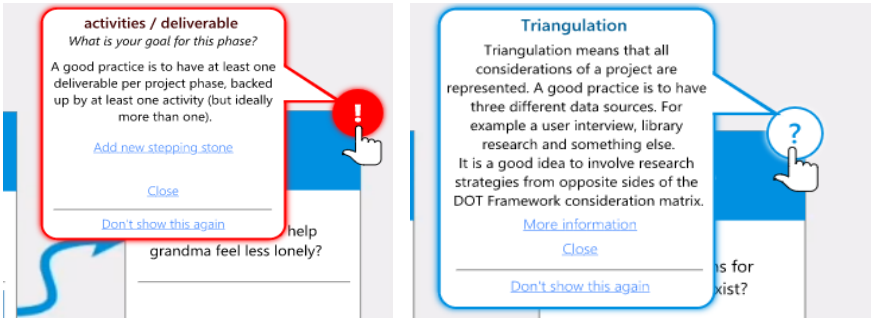
In de eerste versie van het UI-ontwerp wordt feedback aangegeven met ronde “annotaties” boven de plek waar het van toepassing is (zie Figuur 4). Dit is geïnspireerd door Unreal Engine 4, waarin onder andere “breakpoints” in een Blueprint script op dezelfde manier worden weergegeven.



Figuur 4: Het gebruik van "Badges" om feedback te communiceren.

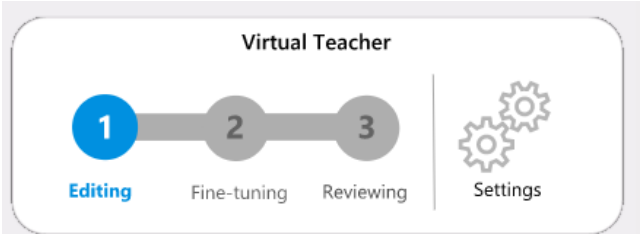
De gebruiker kan op een annotatie klikken om de feedback te bekijken. Op dat moment opent er een “feedback ballon” dat is vormgegeven als een spraakballon, met het “pijltje” dat wijst naar de annotatie om aan te geven waar de feedback bij

hoort. In de ballon staat een korte beschrijving van de feedback, waaronder een paar opties voor vervolgacties te vinden zijn (zie Figuur 5).



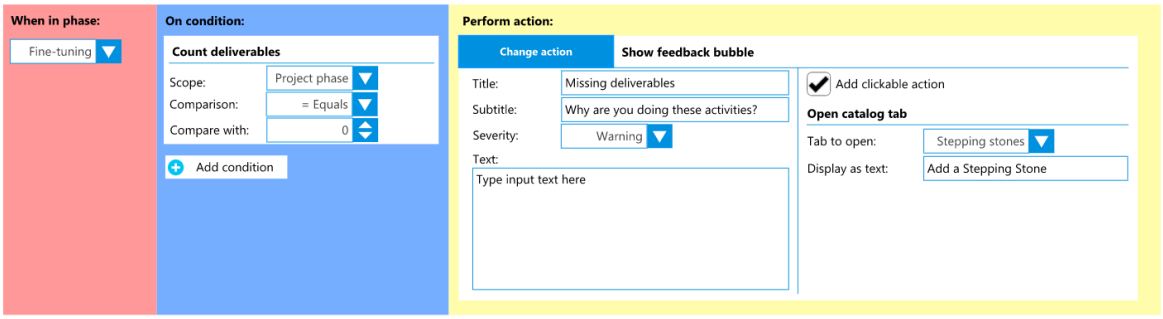
Figuur 5: verschillende "Thema's" voor feedbackballonnen op basis van de ernst van een feedbackpunt.

Om te bepalen wat voor feedback je als gebruiker wil ontvangen, kun je de “Feedback modus” selecteren. Daarvoor is een controlepaneel ontworpen dat rechts onderin het scherm staat. Hierin staat een “modus selector” waarmee de gebruiker kan kiezen tussen drie feedback modi (zie figuur Figuur 6). Ook kan de gebruiker via een knop naar het instellingenvenster om de Virtuele Docent en/of regels aan of uit te zetten.



Figuur 6: Een feedbackmodus selecteren voor de Virtuele Docent

Als laatste is er een pagina waarin docenten feedbackregels kunnen beheren. Een feedbackregel ziet er als volgt uit (zie Figuur 7). Een regel bestaat uit een korte beschrijving en drie hoofdonderdelen: één feedbackmodus, een of meer condities en één actie. Condities kunnen met logische EN / OF operaties worden gecombineerd. Per onderdeel zijn er een aantal instellingen, afhankelijk van het type onderdeel. In het voorbeeld hieronder ziet u een voorbeeld van zo’n feedbackregel.



Figuur 7: Als docent kun je een feedbackregel aanmaken met voorgeprogrammeerde bouwstenen.

7.1.1. Feedback van proefpersonen

Uit de usability tests op de eerste versie van het UI ontwerp is gebleken dat het ontwerp in het algemeen vrij duidelijk was. Wel waren er een aantal feedbackpunten:

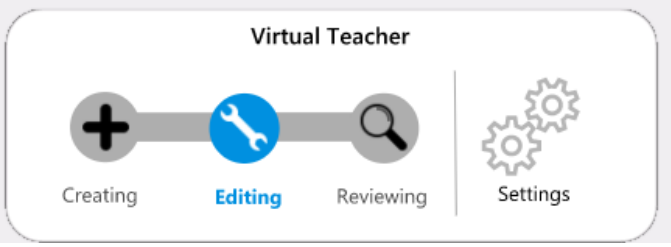
1. De feedbackmodi kunnen beter “Creating”, “Editing”, “Reviewing” heten. Dit is beter te begrijpen dan “Editing”, “Fine Tuning”, “Reviewing”.
2. Bij de tweede en derde optie in de feedbackmodus selector lijkt het alsof alle opties links van de gekozen optie ook actief zijn, maar dit is niet het geval.
3. De feedbackballonnen zijn te klein & slecht leesbaar. De informatie erin is wel duidelijk, alleen niet goed te zien vanwege het kleine lettertype.
4. Bij het instellingenpaneel is niet goed per regel te zien wat voor soort feedback het is. Dit maakt het onnodig moeilijk om specifieke regels te vinden.
5. Bij de optie om de virtuele docent aan/uit te zetten, graag een optie toevoegen of je de virtuele docent permanent - of tot de volgende sessie uit wil zetten.
6. Bij het bewerken van feedbackregels als docent is de dubbele beschrijving (een voor studenten en een voor docenten) onnodig. Liever één beschrijving.
7. Bij het bewerken van feedbackregels als docent, zijn de kleuren voor de drie hoofdonderdelen veel te nadrukkelijk. Liever een subtiele accentkleur in plaats van één groot gekleurd vlak, zodat het niet “van de kleuterschool” lijkt.

7.2. UI ontwerp, iteratie 2

Bij de tweede versie van het UI ontwerp is de feedback van de eerste versie opgelost. De feedbackballonnen zijn opnieuw ontworpen als pop-upvensters. Daardoor is er meer ruimte voor de inhoud, waardoor de leesbaarheid is verbeterd (zie Figuur 8).

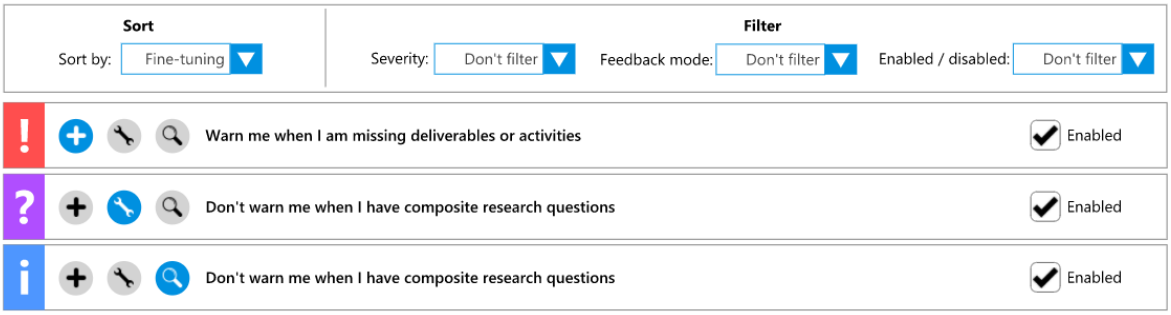
Figuur 8: Nieuwe feedback pop-upvenster

Het element om de feedbackmodus te selecteren, is ook aangepast. De feedbackmodi naamgeving van “Creating”, “Editing”, “Reviewing” is doorgevoerd en elke modus heeft een eigen icoon gekregen in plaats van een nummer. Deze iconen worden consequent op andere plekken gebruikt om de feedbackmodus aan te geven (zie . Als laatste is gezorgd dat bij het selecteren van de 2e of 3e optie, de opties links daarvan niet langer oplichten (zie Figuur 9).



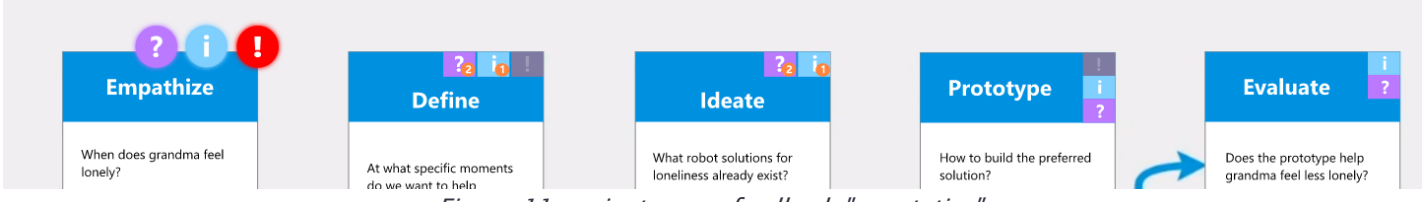
Figuur 9: nieuwe feedbackmodus selectie element

In het instellingenmenu voor feedbackregels wordt per regel aangeven wat voor soort feedback het is en in welke feedbackmodi de regel actief is. Ook zijn er zoek- en filteropties toegevoegd (zie Figuur 10). Al deze elementen ontbraken in de eerste versie, waarin alleen een simpele lijst met regels te zien was.



Figuur 10: Feedbackregel beheer in het instellingenmenu voor studenten.

In de tweede versie van het ontwerp zijn een aantal varianten van de “annotaties” gemaakt die feedback aangeven. In de eerste versie waren dat kleine gekleurde cirkels in de hoek van een projectfase. Het doel van deze varianten was om te bepalen welke visuele stijl de proefpersonen het beste vonden en waarom zij die voorkeur hadden (zie Figuur 11).



Figuur 11: varianten van feedback "annotaties".

7.2.1. Feedback van proefpersonen

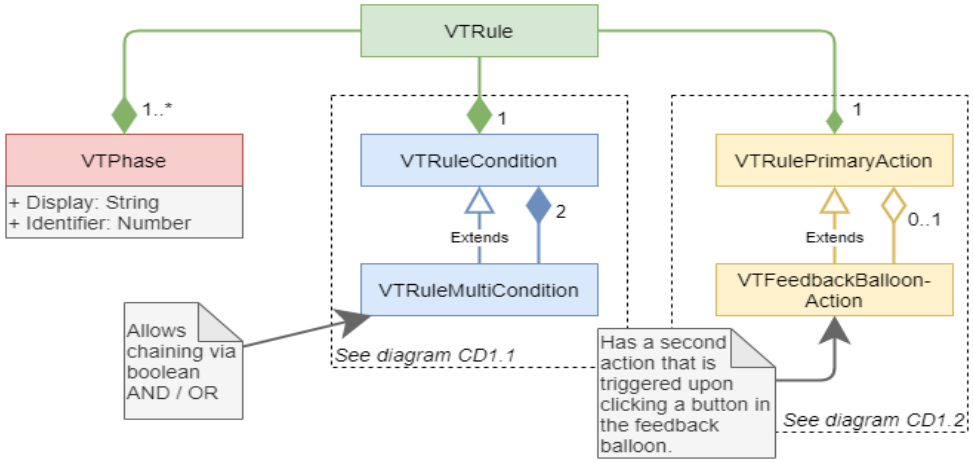
Bij de usability tests van de tweede versie van het UI ontwerp zijn opnieuw een aantal feedbackpunten naar voren gekomen. De proefpersonen waren deze keer veel positiever over het ontwerp.

1. De feedbackmodus selectie element is veel duidelijker. Iconen sluiten goed aan op de verwachte betekenis van elke feedbackmodus.
2. In het instellingenmenu is het nu duidelijk wat elke regel voorstelt. Wel zou het beter zijn om de regels als een tabel te tonen. Dat maakt het zoeken en filteren intuïtiever dan het nu is.
3. Bij de varianten van feedback annotaties, is de voorkeur duidelijk voor de optie "Define", omdat deze een aantal laat zien door middel van een kleine "badge".
4. De nieuwe feedback pop-upvensters zijn veel duidelijker. Het is goed te lezen, maar het is nog niet 100% duidelijk waar de feedback precies bij hoort.

8. Ontwerp: Technische structuur

Op basis van de requirements die zijn bepaald tijdens de interviews met docenten, bleek het mogelijk om veel feedbackpunten in een *"Als conditie X, doe dan actie Y"* vorm te beschrijven. Deze constatering heeft een nieuwe oplossingsrichting geopend: namelijk een modulair, op instelbare regels gebaseerd feedbacksysteem. In dit modulaire systeem kunnen docenten zelf feedbackregels "programmeren" op basis van vooraf gedefinieerde bouwstenen zoals condities en acties. Hiervoor is een systeem bedacht als volgt (zie Figuur 12).

1. Feedbackregels worden gecategoriseerd in drie "**feedbackmodi**". De gebruiker kan op elk moment wisselen tussen feedbackmodi om aan te geven wat voor soort feedback hij of zij wil ontvangen.
2. Elke feedbackregel bestaat uit drie componenten:
    a. **één of meer feedbackmodi** geven aan in welke feedbackmodus de feedbackregel van toepassing is (zie punt 1). Bijvoorbeeld: "*Deze regel geldt in modi Creating en Editing".*
    b. **één conditie** bepaalt welke controle wordt uitgevoerd. Bijvoorbeeld: *"als het aantal deliverables minder is dan 1, geef dan feedback X".*
    c. **één actie** geeft aan op welke manier de feedback aan de gebruiker wordt gemeld. Bijvoorbeeld: "*toon een pop-upvenster met uitleg over het concept Triangulatie"*. Een actie kan ook een secondaire actie bevatten. Dit geeft de gebruiker een manier om op de feedback te reageren, bijvoorbeeld: *"open een externe website met meer informatie"*.



Figuur 12: Opbouw van een feedbackregel (extract uit technisch ontwerp).

Het modulaire regelsysteem is bedoeld om een aantal veel voorkomende feedback te automatiseren. Hiervoor moet deze feedback wel voldoen aan een paar criteria:

1. De feedback is te herkennen aan een specifieke situatie, bijvoorbeeld *"wanneer er minder dan X methodes van strategie Y zijn gebruikt"*.
2. De situatie van punt 1 is te vertalen naar concrete controleregels die beschikbaar zijn in het systeem. Hiervoor zijn een aantal bouwstenen ontwikkeld die zo veel mogelijk situaties afvangen.
3. De feedback is relatief globaal en zal in de meeste gevallen relevant zijn.

8.1. Feedbackregels: bouwsteen "Condities"

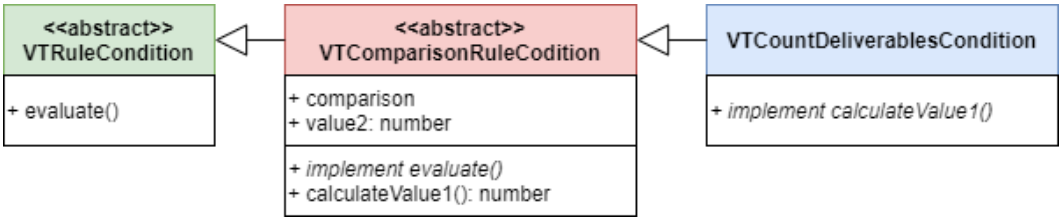
Een groot deel van de complexiteit van een feedbackregel zit in de "Conditie" bouwstenen (zie vorige pagina). Deze condities bepalen wat voor controle er wordt uitgevoerd tijdens het evalueren van een feedbackregel. Een paar voorbeelden:

- Zijn er meer dan drie unieke onderzoeksstrategieën in elke projectfase?
- Heeft elke projectfase tenminste één deliverable (deelresultaat)?
- Wat is de huidige mate van Triangulatie in de projectaanpak?

Om zulke gevarieerde condities mogelijk te maken, is een klassenstructuur ontworpen als volgt:

1. Alle condities erven van de abstracte klasse *VTRuleCondition*. Deze superklasse definieert de functie *evaluate* die de evaluatie van de conditie start.
2. Meerdere gespecialiseerde basistypes implementeren de *evaluate* functie en voeren hierin gespecialiseerde acties uit. De *VTComparisonRuleCondition* bijvoorbeeld berekent twee waardes en vergelijkt deze met een instelbare vergelijkingsoperatie. Deze basistypes zijn ook abstract.

3. Concrete condities zoals *VTCountDeliverablesCondition*, die deliverables telt in een gegeven deel van de projectaanpak, erven van een van de basistypes van punt 2. Deze condities zijn niet abstract en kunnen worden ingezet om de volledige controle uit te voeren.



Figuur 13: feedbackregel condities klassendiagram (sterk versimpeld)

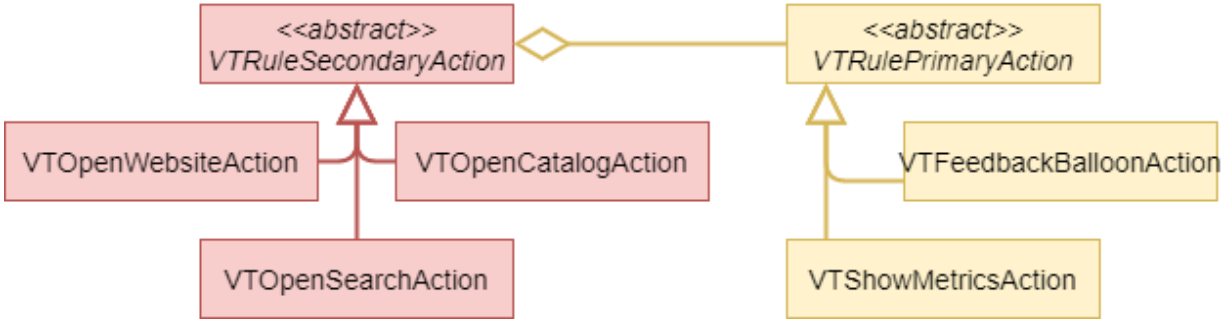
Doordat overlappende functionaliteit van meerdere soorten condities is gedefinieerd in basistypes zoals *VTComparisonRuleCondition* en het eenvoudig is om nieuwe basistypes te introduceren, is het systeem van feedbackregel condities eenvoudig uit te breiden met andere soorten condities. Als in de toekomst bijvoorbeeld op tekst gebaseerde condities gewenst zijn die onderzoeksvragen kunnen analyseren, kan men in ±30 minuten een werkende uitbreiding maken voor het regelsysteem. In Bijlage 4 is een volledig klassendiagram van alle condities en basistypes te zien.

8.2. Feedbackregels: bouwsteen “Acties”

Zodra tijdens evaluatie van een feedbackregel de geprogrammeerde conditie waar is, zal de virtuele docent een actie uitvoeren. Denk bijvoorbeeld aan het tonen van een notificatie zodat de gebruiker een pop-upvenster met feedback kan openen.

8.2.1. Primaire- en secondaire acties

Er wordt onderscheid gemaakt tussen primaire- en secondaire acties. Primaire acties zijn bedoeld om feedback / statistieken aan de gebruiker te melden, bijvoorbeeld via een pop-upvenster. Secondaire acties dienen voor de gebruiker om op de feedback / statistieken te reageren, bijvoorbeeld om de feedback op te lossen. Voor dit systeem is een ontwerp gemaakt (zie Figuur 14).



Figuur 14: Vereenvoudigde klassenstructuur van feedbackregel "actie" bouwstenen.

Zoals eerder uitgelegd, bestaat elke feedbackregel uit feedbackmodi, een conditie en een actie. Deze actie is altijd van het type primaire actie. De primaire actie is verantwoordelijk voor het tonen van feedback. Bijvoorbeeld:

- Toon een notificatie bij het gedeelte van de projectaanpak waarop feedback is. Wanneer de gebruiker op de notificatie klikt, opent een pop-upvenster met uitleg over de feedback.
- Toon statistische informatie over de projectaanpak zodat studenten en docenten daar zelf conclusies op kunnen trekken, bijv. over Triangulatie.

Elke primaire actie kan optioneel een secondaire actie hebben. Hoe deze wordt weergegeven, is afhankelijk van het type primaire actie. In het algemeen geldt dat secondaire acties de student in staat stellen om te reageren op feedback. Een paar voorbeelden:

- Open een specifieke tab in de catalogus met onderzoeksmethoden, bijvoorbeeld wanneer er een deliverable ontbreekt.
- Open een tab van de catalogus om alle onderzoeksmethodes te doorzoeken en stel een specifieke zoekterm in, bijvoorbeeld als de virtuele docent adviseert om een bieb- of werkplaatsmethode toe te voegen.
- Open een externe webpagina met uitleg over een theoretisch onderwerp van het DOT Framework, bijvoorbeeld om uit te leggen wat Triangulatie is.

Zowel primaire - als secondaire acties zijn op zo'n manier ontworpen dat deze nooit de workflow van de gebruiker onderbreken. Bijvoorbeeld een primaire actie die een pop-upvenster zal tonen, zal het pop-upvenster pas tonen zodra de gebruiker dat expliciet aangeeft. En statistieken over de projectaanpak zullen pas worden weergegeven als de gebruiker daar voor kiest. Zo kan worden voorkomen dat de virtuele docent de gebruiker in de weg zit.

9. Prototyping: bestaand platform

De virtuele docent bouwt voort op de bestaande, huidige versie van de Project Approach Tool. In dit hoofdstuk wordt beschreven wat de beginsituatie is, welke wijzigingen zijn doorgevoerd en wat het eindresultaat is. Bij aanvang van de afstudeerperiode was Project Approach Tool een volledig functioneel systeem om een projectaanpak te ontwerpen. Als student kon je een account registreren, waarmee je via Google Firebase kon authentiseren. Hiervoor was geen wachtwoord nodig. Omdat dit strikt gezien geen multifactor-authenticatie is (de enige factor is het bezit van het

opgegeven emailadres), schat ik zelf de veiligheid van dit systeem niet optimaal in. Verbeteren van de veiligheid valt echter buiten de afbakening van mijn project.

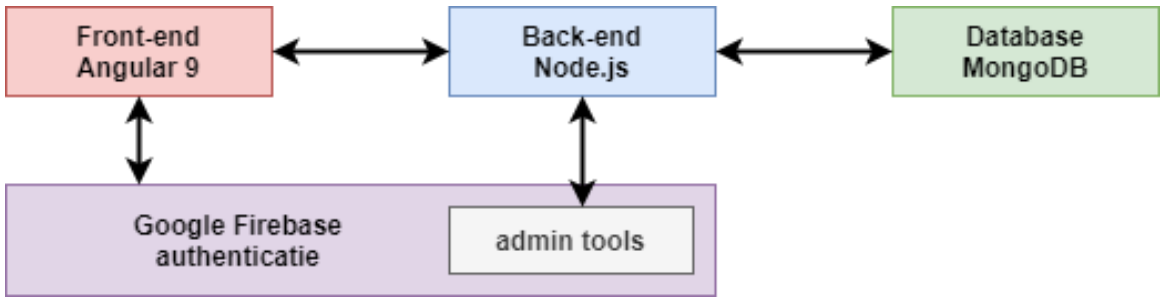
Na inloggen kon je als student een zogeheten “workspace” aanmaken. Dit is een omgeving waarin je één projectaanpak kan maken. Elke projectaanpak heeft een eigen workspace. Een workspace kan worden aangemaakt op basis van een standaardtemplate, bijvoorbeeld Design Thinking. De projectfasering wordt dan aangemaakt volgens de methodiek in het template, maar is voor de rest leeg.

Eenmaal in een workspace, kun je als student projectfases toevoegen / verwijderen, een hoofdvraag schrijven en per fase onderzoeksvragen, -methodes en deliverables toevoegen.

Als laatste kun je een workspace ook delen met anderen. Door een email-adres van de ontvanger in te vullen kun je andere studenten uitnodigen naar jouw workspace. Via een websocket-verbinding o.b.v. de Socket.io bibliotheek (Socket.io, n.d.) worden alle wijzigingen gesynchroniseerd tussen alle leden van een workspace.

9.1. Technische architectuur

Project Approach Tool bestaat uit vier componenten, namelijk een front-end app, back-end app, een extern authenticatieplatform en een database. De architectuur van het platform ziet er als volgt uit (zie figuur hieronder).



De front-end is een Angular 9 applicatie geschreven in Typescript. De back-end app is op Node.js gebaseerd en is geschreven in Javascript. Beide applicaties gebruiken het platform NPM als bron van softwarebibliotheken en -afhankelijkheden. De front-end gebruikt daarnaast ook enkele “CDN” providers --- dit zijn externe leveranciers waarvan softwarebibliotheken op runtime kunnen worden ingeladen.

De back-end gebruikt voor persistentie MongoDB, een zogeheten *"document driven No-SQL database"*. MongoDB gebruikt voor documenten een structuur die veel lijkt op het JSON dataformaat. Daarnaast ondersteunt het optioneel dataschema's om een bepaalde documentstructuur af te dwingen. Project Approach Tool gebruikt dergelijke schema's om de structuur van Workspaces vast te leggen.

Als laatste maken de front-end en back-end gebruik van Google Firebase om gebruikers te authentiseren. In PAT dient Firebase puur als authenticatiemiddel. Autorisatie wordt door de back-end intern afgehandeld.

10. Prototyping: overzicht van wijzigingen

Om de Virtuele Docent te realiseren zijn een aantal wijzigingen doorgevoerd. Deze zijn grofweg in te delen in de volgende categorieën: integratie met het bestaande systeem, genereren en tonen van feedback, instellen van feedbackregels, en algemene wijzigingen. In Bijlage 5 vindt u een overzicht van alle gemaakte wijzigingen.

Samenvatting van wijzigingen backend

In de backend applicatie zijn een aantal wijzigingen doorgevoerd om de Virtuele Docent mogelijk te maken. Ten eerste is een koppeling gemaakt met het bestaande, websocket gebaseerde synchronisatiemechanisme (zie hoofdstuk 11.1). Deze koppeling verbindt de websocket interface met een "Virtual Teacher Service" die dient als een soort "facade" voor alle Virtuele Docent functies. Daarna is een feedback generatorsysteem gemaakt op basis van modulaire regels. Om deze regels voor docenten instelbaar te maken, is een rol-gebaseerd autorisatiesysteem gemaakt om een aantal nieuwe, administratieve API-endpoints te beveiligen. Deze endpoints zijn uiteraard ook nieuw ontwikkeld om regels op te slaan en uit te lezen.

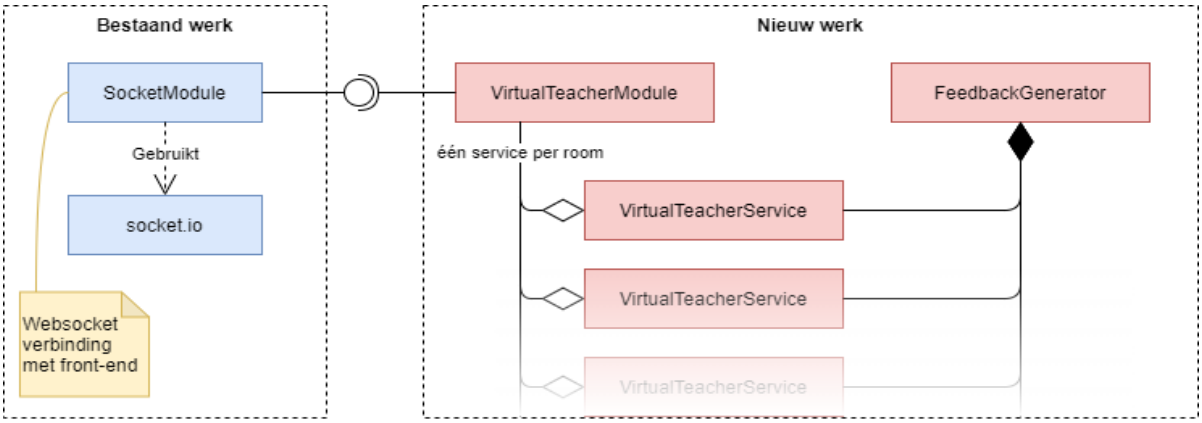
Samenvatting van wijzigingen front-end

De front-end heeft enkele wijzigingen ondergaan om feedback te tonen. Onder andere een Virtuele Docent dashboard, feedback pop-upvenster, "drawer menu" en een instellingenpaneel zijn hiervoor ontwikkeld. Ook enkele bestaande Angular componenten zijn aangepast. Voor het kunnen instellen van feedbackregels is ook in de front-end een autorisatiesysteem gerealiseerd, wat door middel van "route guards" een nieuwe, administratieve pagina voor docenten afschermt. Op deze pagina kunnen docenten feedbackregels beheren, en kunnen administratoren gebruikersrollen promoveren of demoveren. Ook zijn meer dan honderd security zwaktes opgelost in afhankelijkheden, waaronder een aantal met kritische ernst.

11. Prototyping: Analyse-engine

De wijzigingen aan de Project Approach Tool zijn grofweg genomen in drie “fases” uitgevoerd. In de eerste fase is de “analysis engine” ontwikkeld. In de tweede fase is gezorgd dat de feedback van de Virtuele Docent goed wordt weergegeven. In de derde fase zijn de feedbackregels die door de Virtuele Docent worden uitgevoerd, instelbaar gemaakt.

De analyse-engine is het brein van de Virtuele Docent. Het is verantwoordelijk voor drie dingen: luisteren naar wijzigingen in een projectaanpak, verzamelen van benodigde en uiteindelijk feedback genereren. Hiervoor gebruikt de analyse-engine een aantal verschillende klassen (zie Figuur 15).



Figuur 15: vereenvoudigde structuur van de Virtuele Docent analyse-engine

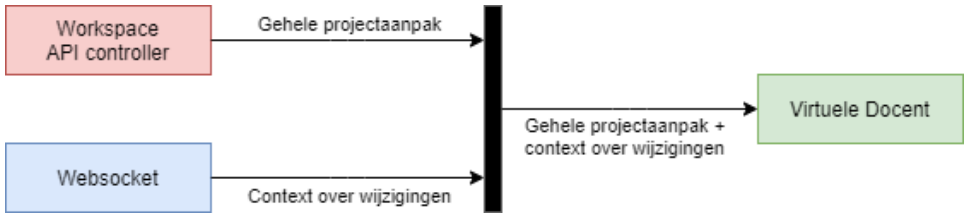
11.1. Aansluiting op het bestaande platform

Voor iedere workspace met tenminste één actieve gebruiker wordt een instantie van een “Virtual Teacher Service” gemaakt Elke instantie stelt één virtuele docent voor en geeft alleen feedback op de geassocieerde workspace. Om in de eerste plaats te weten wanneer feedback moet worden gegenereerd, moest de Virtuele Docent worden aangesloten op het bestaande platform van de Project Approach Tool. De eerste optie was de “workspace” controller in het API gedeelte van de back-end. De tweede optie was de websocket interface die wordt gebruikt voor datasynchronisatie tussen meerdere gebruikers die samen een projectaanpak bewerken.

11.1.1. Versie 1: Hybride oplossing met synchronisatiemechanisme

De virtuele docent aansluiten op de websocket verbinding zou als voordeel hebben dat het snel is omdat er sprake is van een real-time verbinding. Ook zou dan context kunnen worden verzameld over wijzigingen. De workspace API controller daarentegen ontvangt de gehele projectaanpak, maar kan deze context niet bieden.

Om een balans te vinden tussen de voor- en nadelen van beide opties, is aanvankelijk gekozen voor een hybride oplossing. Door de virtuele docent te koppelen met zowel de workspace API controller als de websocket verbinding, kon alle vereiste informatie worden verzameld. Wel was een synchronisatiemechanisme nodig dat pas feedback zou genereren zodra beide databronnen hun informatie hadden afgeleverd. Tijdens ontwikkeling aan de front-end werd de Project Approach Tool regelmatig herladen en werden wijzigingen in vrij hoog tempo doorgevoerd. Op dit punt begon de informatie asynchroon te lopen, waardoor feedback al gedateerd was zodra het werd gegenereerd. Dit probleem kon helaas niet goed worden opgelost.

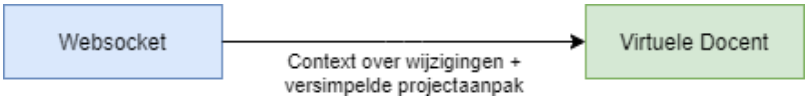


Figuur 16: Deze versie werkte slecht vanwege synchronisatieproblemen tussen beide databronnen.

11.1.2. Versie 2: Websocket oplossing met totaalanalyse

Bij de tweede versie van de koppeling met het bestaande platform is de workspace API controller volledig buiten beeld gelaten. Door de virtuele docent alleen met de websocket te koppelen, was een synchronisatiemechanisme niet langer nodig. Het probleem was echter wel dat informatie over de gehele projectaanpak ontbrak. Daarom is de front-end aangepast, zodat deze bij elke wijziging ook een versimpelde versie van de projectaanpak meestuurt over de websocket. Deze projectaanpak bevat precies genoeg informatie voor de virtuele docent om feedback te genereren.

De resulterende oplossing is zeer snel --- zo snel dat een contextuele analyse niet langer nodig is voor performance. Theoretisch kan de context over wijzigingen daardoor worden genegeerd, maar om de optie voor een contextuele analyse open te houden is gekozen om deze informatie wel beschikbaar te laten.



Figuur 17: De uiteindelijke koppeling tussen het bestaande platform en de virtuele docent.

In Bijlage 6 ziet u enkele codevoorbeelden van de uiteindelijke oplossing. De code in deze voorbeelden is niet 100% volledig: alleen code die relevant is voor de koppeling tussen de virtuele docent en het websocket synchronisatiemechanisme, zijn weergegeven in de voorbeelden.

11.1.3. Conclusie

Achteraf gezien was het synchronisatiemechanisme van versie 1 een naïeve, complexe oplossing. Als de virtuele docent met slechts één verbonden gebruiker al zo onvoorspelbaar werkte, zou het met meerdere gebruikers samen in één projectaanpak nog veel slechter werken. De uiteindelijke implementatie daarentegen is de eenvoud zelve en werkt bijzonder snel dankzij de real-time websocket verbinding met de front-end.

11.2. Feedback genereren

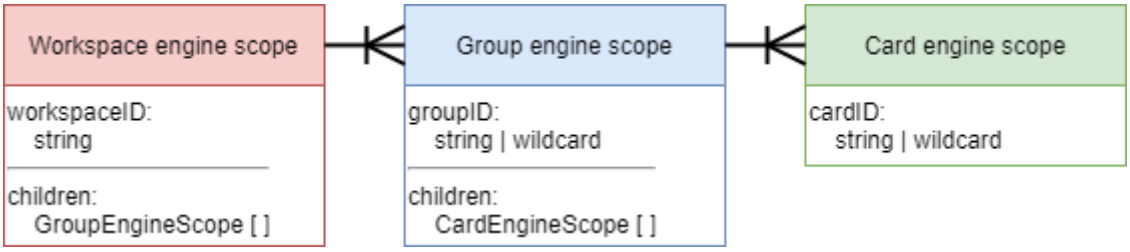
De feedbackgenerator wordt door de Virtuele Docent gestart bij elke wijziging aan een projectaanpak. Daarvoor zijn drie types informatie nodig: de projectaanpak om te analyseren, een set met feedbackregels om uit te voeren en een zogeheten “engine scope”. Dit laatste dient om het analysegebied af te bakenen, zodat bijvoorbeeld alleen projectfase X tot en met Y wordt geanalyseerd. In andere woorden: Engine Scopes stellen de virtuele docent in staat om contextuele analyses uit te voeren op de plek waar de gebruiker aan het werk is. Dit is een veelgebruikte manier om performance te verbeteren, wat men onder andere terug kan zien in software IDE’s zoals Microsoft Visual Studio en JetBrains IntelliJ.

11.2.1. Engine Scopes

De engine scopes dienen om het analysegebied af te bakenen. Hoe werkt dat precies? Een projectaanpak bestaat uit drie niveaus: een workspace, daarin een of meer “groepen”, en daarin een of meer “kaarten”. Dit zijn technische benamingen voor onderdelen van een projectaanpak.

Passend bij deze drie niveaus zijn er drie soorten Engine Scopes gedefinieerd (zie de lijst & figuur hieronder).

- Workspace: een projectaanpak.
- Groepen: fases in een projectaanpak (bijv. Empathize of Define)
- Kaarten: Onderzoeksvragen, activiteiten en deliverables.



Figuur 18: De structuur van een hiërarchie van engine scopes.

In de figuur hierboven zijn een aantal ontwerpkeuzes te zien. Ten eerste het onderscheid tussen workspace-, groep- en kaart scopes. Zoals eerder vermeld zijn dit de drie hoofdonderdelen van een projectaanpak. De logische keuze is om voor elk type een "scope" klasse te definiëren. Helaas is een meer generieke oplossing niet goed mogelijk vanwege subtiele verschillen tussen workspaces, groepen en kaarten.

U vraagt zich wellicht af: hoe weet de virtuele docent dat een engine scope van elk willekeurig niveau, geldt voor een element van dat niveau? Dit werkt als volgt. Elke engine scope accepteert een "identifier". Dit is een unieke sleutel die hoort bij een specifiek onderdeel van de projectaanpak. De sleutel mag ook een wildcard zijn. In dat geval geldt de scope voor alle elementen van dat niveau. Scopes hebben een functie die zegt of een gegeven deel van de projectaanpak binnen de scope valt.

11.2.1.1. Conclusie

Na testen kan worden geconcludeerd dat de engine scopes goed werken. Toch worden ze op het moment van schrijven niet maximaal benut, omdat de huidige set van condities mogelijk afhankelijk zijn van informatie die buiten de scope zou vallen. Maar de kans is groot dat in de toekomst het afbakeningssysteem van de engine scopes meer tot zijn recht kan komen, wanneer er meer specifieke feedbackregels worden toegevoegd. De huidige implementatie kan men dus beschouwen als voorbereiding op de toekomst.

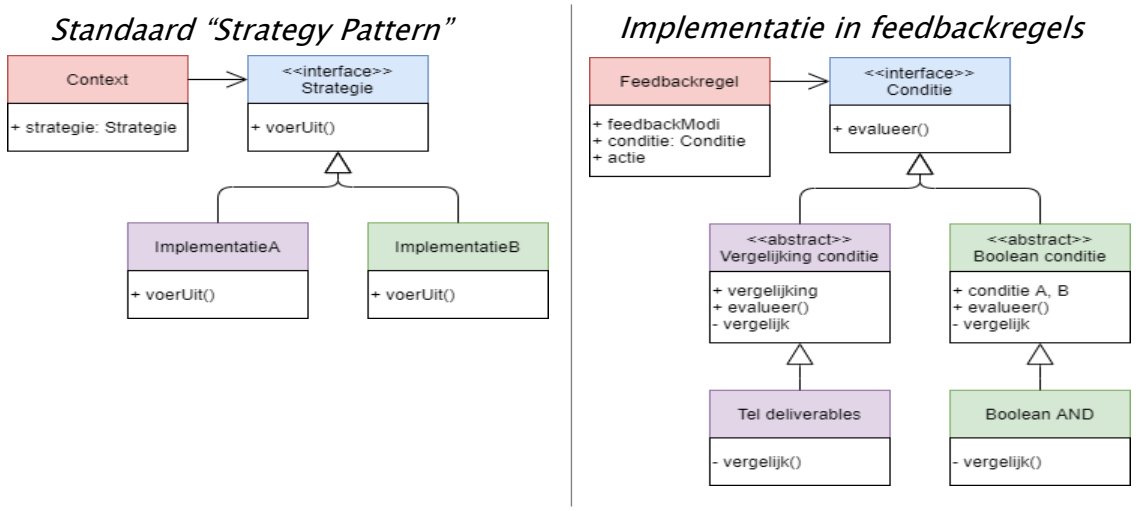
11.2.2. Feedback genereren door middel van modulaire regels

Om feedbackregels instelbaar te maken, is een modulair regelsysteem bedacht. In hoofdstuk 8 (pagina 24) is daarover meer te lezen. In de uiteindelijke implementatie van dit regelsysteem vertegenwoordigen feedbackregels concrete, uitvoerbare controles. Afhankelijk van de opgegeven soort conditie wordt een ander resultaat gegenereerd. Dit lijkt qua werking en structuur veel op het zogeheten "Strategy Pattern", een van de 23 software-ontwerppatronen die door de zogenoemde "gang of four" is geïntroduceerd in het boek *"Design Patterns: Elements of Reusable Object-Oriented Software"* (Gamma, Helm, Johnson, & Vlissides, 1994).

In het Strategy Pattern wordt een uitwisselbaar scenario voorgesteld waarvan varianten zijn geïmplementeerd in zogenoemde "strategieën". Het voordeel hiervan is dat de code die de scenario's uitvoert, geen kennis vereist over de specifieke variant die wordt aangeroepen. Daardoor krijg je sterke encapsulatie van scenario's en blijft de code die de scenario's aanroept, zeer simpel.

In de virtuele docent wordt een iets ingewikkeldere versie met meerdere lagen gebruikt. Dit is omdat een aantal “condities”, die onderdeel uitmaken van een regel, gedeelde functionaliteit hebben zoals het vergelijken van twee delen van een projectaanpak. De gedeelde functionaliteit is opgenomen in abstracte klassen die specifieke implementatie-aspecten uitbesteden aan concrete subklassen.

In Bijlage 8 ziet u hoe een feedbackregel zichzelf dankzij het Strategy Pattern kan evalueren met geen tot weinig kennis van het exacte type conditie of actie dat is ingesteld.



Figuur 19: strategy pattern t.o.v. implementatie in de virtuele docent (versimpeld).

11.2.2.1. Conclusie

Dankzij het strategie patroon is de implementatie van modulaire feedbackregels relatief simpel gebleven zonder de functionaliteit te beperken. Dit komt de onderhoudbaarheid en uitbreidbaarheid van het systeem ten goede. Uit testen kan worden geconcludeerd dat dit in combinatie met de websocket-verbinding en de engine scopes als afbakening samen een goed werkend geheel vormt. Het systeem is snel, modulair uit te breiden en kan eenvoudig worden aangepast bij toekomstige vervolgprojecten.

12. Prototyping: feedback tonen en verwerken

Zodra de virtuele docent feedback heeft gegenereerd, wordt dit via de websocket verbinding naar alle gebruikers gestuurd die de bijbehorende projectaanpak aan het bewerken zijn. Hoe toont de front-end app de feedback? En wat kan de gebruiker vervolgens doen om de feedback te verwerken? Dat was het doel van de tweede ontwikkelfase. Zodra dit doel eenmaal was voltooid, zou de functionele kern van de

functionele docent zo goed als af zijn. Welke wijzigingen daarvoor nodig waren, wordt in de komende paragrafen toegelicht.

12.1. Angular 9 in een notendop

De front-end app maakt gebruik van Angular (Angular / Google, n.d.). Dit bekende framework van Google is meer dan een UI framework. Het is een alles-in-een pakket voor single-page webapplicaties, programmalogica, UI architectuur en communicatie met externe API's, om een paar dingen te noemen. Project Approach Tool maakt gebruik van Angular versie 9 van februari 2020 (Angular / Google, n.d.).

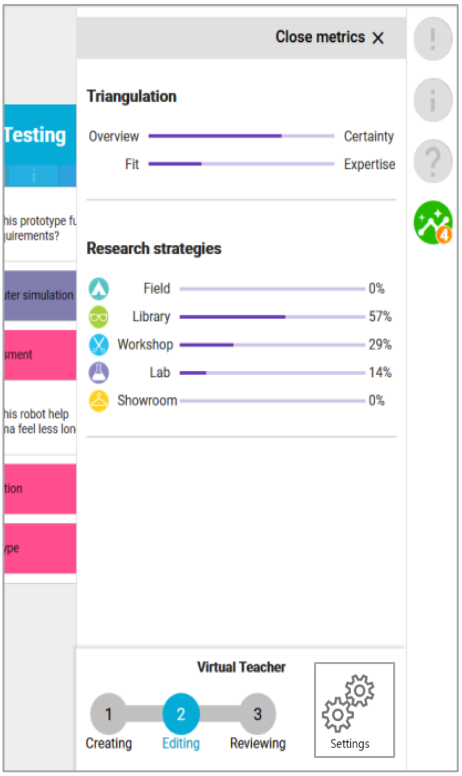
In de komende paragrafen worden een aantal Angular-specifieke begrippen gebruikt:

1. **Components**: dit zijn afgebakende delen van de gebruikersinterface met eigen code, vormgeving en opmaak. Elk component bestaat uit een Typescript klasse, HTML document en een eigen CSS stylesheet.
2. **Services**: dit zijn herbruikbare "diensten" met daarin applicatielogica. Denk bijvoorbeeld aan API communicatie of eventafhandeling.
3. **Dependency Injection**: dit is een systeem waarin code-afhankelijkheden automatisch van buitenaf worden "geïnjecteerd" in klassen die die afhankelijkheden nodig hebben. Angular heeft een ingebouwde dependency injector waarvoor geen handmatige registratie of installatie nodig is.

12.2. Virtuele docent dashboard

Een Angular component met de belangrijkste functionaliteit van de virtuele docent.

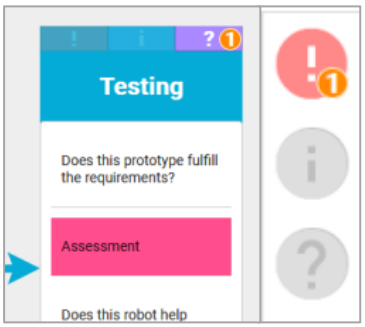
Als hulpmiddel voor begeleiding en feedback moet de virtuele docent eenvoudig zijn te gebruiken. Daarvoor is een "dashboard" component gemaakt waarop de belangrijkste functionaliteit van de virtuele docent beschikbaar wordt gesteld. Het dashboard is altijd zichtbaar op de bewerkingsomgeving van een projectaanpak en neemt standaard weinig ruimte in. In een paneel rechts onderin het scherm kan de gebruiker de huidige "feedbackmodus" selecteren. Langs de rechterrand van het scherm staat een menubalk waarin knoppen voor algemene feedback en statistieken worden getoond.



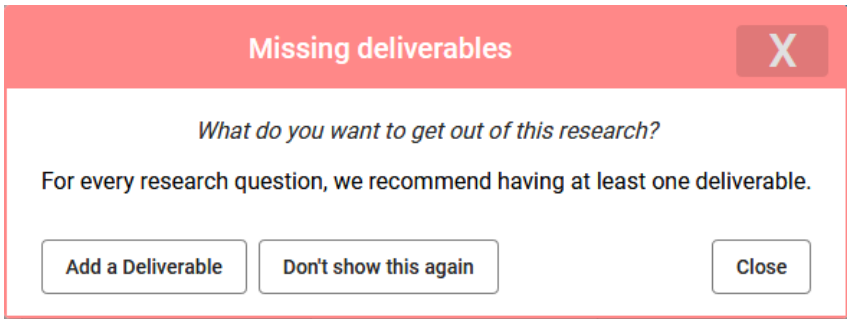
12.3. Feedback pop-upvenster

Een Angular component en – service die samen feedback in een pop-upvenster tonen wanneer de gebruiker op een daarvoor bestemde knop klikt.

Wanneer de virtuele docent feedback geeft op de projectaanpak, zijn er twee mogelijke "scopes" waar de feedback bij hoort: een specifieke projectfase, en de gehele projectaanpak. In de header van elke projectfase worden drie "badges" getoond die horen bij de drie mogelijke feedbackniveaus: waarschuwing, vraag en informatie. In de verticale menubalk van het Virtuele Docent Dashboardcomponent staan eveneens drie knoppen die bij deze feedbackniveaus horen (zie Figuur 21). Wanneer feedback van een bepaald niveau beschikbaar is, licht de knop / badge van dat niveau op bij het bijbehorende deel van de projectaanpak. Wanneer de gebruiker daarop klikt, opent een pop-upvenster met meer informatie over de feedback.



Figuur 21: Badges voor feedback



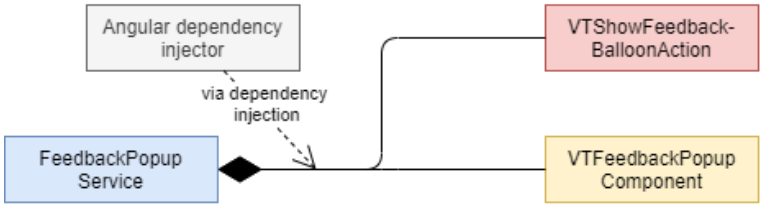
Figuur 21: Het pop-upvenster met een beschrijving van de feedback.

De gebruiker kan op drie manieren reageren op de feedback: sluiten, negeren en een voorgestelde actie uitvoeren. In Figuur 21 zijn deze opties respectievelijk te zien als "*Close*", "*Don't show this again*" en "*Add a deliverable*". Als de gebruiker kiest om de feedback te negeren, wordt de pop-up gesloten en zal deze feedback niet vaker worden getoond (de feedbackregel wordt technisch "uitgeschakeld").

Als de gebruiker kiest om de voorgestelde actie uit te voeren, zal afhankelijk van de voorgestelde actie iets gebeuren. In Figuur 21 zal de catalogus uitklappen op de categorie "Stepping Stones" zodat de gebruiker een deliverable kan kiezen en toevoegen. Andere mogelijkheden zijn: onderzoeksmethodes zoeken in een zoekvenster, of een externe webpagina openen met meer uitleg.

```
export class VTFeedbackBubbleService {
  //deze code zorgt voor een observeerbaar event waarop UI componenten kunnen reageren.
  @Output() showFeedback: EventEmitter<VTActionFeedbackBubble> = new EventEmitter<VTActionFeedbackBubble>();
}
```

Om de feedback pop-up mogelijk te maken, is de volgende technische oplossing gerealiseerd (zie figuur hieronder).



Figuur 22: klassendiagram van de feedback pop-up oplossing

De oplossing werkt als volgt: een Angular service (FeedbackPopupService) wordt in het pop-up component en een Angular model voor de bijbehorende feedback actie geïnjecteerd en dient als bemiddelaar tussen de twee. Dit lijkt enigszins op het Mediator design pattern (Gamma, Helm, Johnson, & Vlissides, 1994), maar is niet 100% hetzelfde. Wel heeft het ongeveer dezelfde voordelen omdat het een vorm van "*loose coupling*" faciliteert tussen componenten. Dit betekent dat componenten van een systeem zo min mogelijk kennis over elkaar hebben, zodat individuele componenten makkelijk uitwisselbaar zijn. In dit geval is het tonen van een feedback pop-upvenster nu onafhankelijk van de structuur van de gebruikersinterface.

```
export class VTActionFeedbackBubble extends VTPrimaryAction {
    //deze code wordt aangeroepen vanuit UI componenten om feedback te tonen.
    perform() { this._feedbackService.showFeedback.emit(this); }
}

export class VTFeedbackBubbleComponent implements OnInit {
    //dit is het feedback popup component.
    ngOnInit(): void {
        this.vtFeedbackService.showFeedback.subscribe(x => this.showFeedback(x))
    }
    showFeedback(data: VTActionFeedbackBubble) {
        const dialogRef = this.dialog.open(VTFeedbackBubbleDialogComponent, { … });
        dialogRef.afterClosed().subscribe(result => { /*verwerk resultaat*/ });
    }
```

12.4. Conclusie

Met behulp van waardevolle feedback van een tweetal proefpersonen (zie hoofdstuk 7) wordt feedback op een niet storende manier weergegeven. Het gebruik van twee bekende software ontwerppatronen (Strategy & Mediator patronen) resulteert in een nette code-oplossing die voor toekomstige ontwikkelaars relatief eenvoudig is te onderhouden. Niet alle beoogde functionaliteit is in dit stadium voltooid, zoals het instellen van feedbackregels. Dit volgde in een later stadium.

Iteratieve werkwijze

Veel oplossingen zijn ontstaan als volgt. Initieel is een strikt functionele oplossing ontwikkeld dat voldeed aan de gestelde vereisten. De focus ligt daarbij op functionaliteit – wat is nodig om een werkende oplossing te realiseren? Daardoor was zo'n eerste oplossing niet per sé de meest elegante of best geschreven oplossing. De oplossing wordt vervolgens kritisch geanalyseerd om verbeterpunten te identificeren en aan te passen.

13. Prototyping: instelbare feedbackregels

In dit stadium werkte de feedbackgenerator en kon feedback worden weergegeven in de front-end. De functionele kern van de virtuele docent was voltooid. Als laatste was het vereist dat de feedbackregels waarop de virtuele docent feedback baseert, kunnen worden ingesteld door docenten. Dit was het doel van de derde ontwikkelfase.

13.1. Autorisatie

Studenten hebben niet het recht om de globale feedbackregels in te stellen. Om te zorgen dat alleen docenten dit mogen, moest een autorisatiesysteem worden toegevoegd aan zowel de front-end als de backend. Dit was nog niet aanwezig, in tegenstelling tot authenticatie, wat al sinds de eerste opzet van de Project Approach Tool in het voorjaar van 2020 in het systeem aanwezig is.

Voor authenticatie gebruikt Project Approach Tool het Firebase-platform van Google. Wanneer een gebruiker inlogt, krijgt hij of zij een JWT bearer token waarmee toekomstige webrequests kunnen worden geauthentiseerd. Firebase ondersteunt autorisatie via *"custom claims"*. Echter, wijzigingen aan de custom claims van een Firebase gebruiker worden niet gelijk doorgevoerd vanwege de aard van bearer tokens (eenmaal uitgegeven kun je een bearer token niet zomaar ongeldig verklaren). Daarom is besloten om een intern, server-side autorisatiesysteem te ontwikkelen dat leidend is boven Firebase en het geleverde JWT token. Daarvoor moesten een aantal wijzigingen worden doorgevoerd.

Ten eerste moest in de server app het authenticatiemechanisme worden uitgebreid, zodat een Firebase gebruiker wordt voorzien van een rol. De geaccepteerde rollen zijn Student, Docent en Admin. Het onderscheid tussen docenten en admins is dat alleen admins gebruikers mogen promoveren of demoveren. Zodra een gebruiker via Firebase inlogt, wordt de claim "rol" van het Firebase token waar nodig bijgewerkt met de up-to-date rol in de lokale database. De volgende keer dat de gebruiker inlogt of zijn/haar token vernieuwt, wordt het Firebase token bijgewerkt met de nieuwe rol.

Authenticatie via Firebase kan via een zogeheten "*fluent interface*" (Fowler, 2005) dat gebruik maakt van Javascript Promises om bepaalde aspecten van een gebruikersidentiteit te controleren. In Rechts ziet u hoe men de geldigheid van een token kan controleren.

```
const decodedToken = await admin
    .auth()
    .verifyIdToken(req.headers.authorization)
    .catch(() => rejectAuth(res));
  if (!decodedToken) {
    rejectAuth(res);
    return;
  };
```

Om te zorgen dat er altijd een administrator account is, is er één inlogaccount dat altijd administrator rechten zal hebben. Dit account is instelbaar als environment variabele. Hoewel dit niet de meest elegante oplossing is, is het voor nu goed genoeg.

```
const dbUser = await User
    .findOne({uid: decodedToken.uid})
    .catch(() => console.warn(`User ${decodedToken.uid} does not exist in the local DB.`)
);

if (dbUser && dbUser.email == teacherEmail && dbUser.role !== 'admin') {
  //user did not have teacher role, so it must be re-enabled.
  dbUser.role = 'admin';
  await User.findOneAndUpdate({uid: dbUser.uid}, dbUser);
}
```

Zodra een gebruiker inlogt, wordt als laatste in het Firebase token de meest recente gebruikersrol ingevuld. Deze wijziging zal pas worden doorgevoerd zodra de gebruiker opnieuw inlogt; dit kan echter (nog) niet in de backend worden geforceerd.

```
const authUser = await admin.auth().getUser(decodedToken.uid);
  const userRole = authUser.customClaims['role'];
  if (userRole !== dbUser.role) {
    await admin
      .auth()
      .setCustomUserClaims(decodedToken.uid, { role: dbUser.role });
      dbUser.save();
  }
```

Ten tweede moest in de front-end en backend zogeheten “Route guards” worden toegevoegd die afdwingen dat gebruikers alleen met de juiste rol een bepaalde webpagina of endpoint mogen raadplegen. Route guards zijn per pagina individueel instelbaar en per route guard kan men opgeven welke rechten de gebruiker moet hebben. In de Angular router die routering binnen de front-end app afhandelt, wordt het Firebase JWT token gecontroleerd.

```
const teacherOnly = () => pipe(customClaims, map(claims => {
  console.log(claims);
  return claims.role === 'teacher' || claims.role === 'admin'
}));
...
const routes: Routes = [{
    path: "teacher/settings",
    component: TeacherSettingsPageComponent,
    canActivate: [AngularFireAuthGuard],
    data: { authGuardPipe: teacherOnly },
  }, ...
]
```

13.2. Instellingenpagina voor docenten

Op de instellingenpagina kunnen docenten de globale feedbackregels van de Virtuele Docent beheren. Admins kunnen hier ook gebruikers promoveren of demoveren naar een hogere of lagere rol. Studenten kunnen en mogen deze pagina niet bezoeken en alle bijbehorende API endpoints in de server app zijn eveneens afgeschermd via de hiervoor besproken route guards.

Om feedbackregels in te stellen, is een interessant mechanisme van Angular gebruikt genaamd “Component Factories”. Het probleem was dat, afhankelijk van het type modulaire conditie dat in een feedbackregels, een ander soort Angular component moest worden ingeladen in de UI. Component Factories zijn daar geschikt voor.

Om een component dynamisch in te laden, is eerst een “anchor” nodig. Voor de duidelijkheid: dit is geen HTML anchor tag *(<a href=“…”>)*, maar kan elk willekeurig element zijn zolang deze via een *“@ViewChild selector”* kan worden geselecteerd.

```
//Anchor component where the dynamic condition component will be injected.
@ViewChild(VtRuleConditionDirective, {static: true})
conditionDirective: VtRuleConditionDirective;
```

Omdat het dynamisch inladen van conditie componenten op meerdere plekken moet gebeuren, is de component factory inbegrepen in een abstract basiscomponent. Deze biedt een functie `loadComponent` aan die op basis van het type conditie het ene of andere component inlaadt. Zodra een ander soort modulaire conditie wordt geselecteerd, wordt ook de `loadComponent` functie opnieuw aangeroepen om het dynamisch geladen component te vernieuwen of vervangen. Hetzelfde dynamische laadsysteem wordt ook gebruikt voor modulaire feedbackacties. Dit systeem is echter iets simpeler omdat acties niet kunnen worden geketend via een AND / OR operatie, wat condities wel kunnen. Maar voor de rest is dit systeem in essentie hetzelfde.

```
loadComponent(conditionSelector: Function, viewRef: ViewContainerRef) {
    viewRef.clear();
    const condition = conditionSelector();
    if (!condition || !this.configurableConditions) return;

    const componentFactory = this.componentFactory.resolveComponentFactory(
        this.mapConditionToType(condition.identifier));
    const component = viewRef.createComponent<VtRuleConditionComponent>(componentFactory);

    component.instance.condition = condition;
    component.instance.onDelete.subscribe(x => {
      this.deleteCondition(x)
      this.reloadComponent();
    });
}
```

14. Test: evaluatie van werkwijze

Uit de vorige hoofdstukken is duidelijk gebleken dat tijdens het project diverse werkzaamheden zijn uitgevoerd. De kwaliteit van de uitvoer en resultaten hiervan zijn op verschillende manieren bewaakt.

Zoals eerder vernoemd, is bij de uitvoer van het project een Agile / Scrum achtige iteratieve werkwijze gehanteerd. Een cruciaal onderdeel van Agile / Scrum zijn *retrospectives*: momenten waarop wordt teruggekeken op de afgelopen sprint om sterke en zwakke punten van de werkwijze te identificeren. Vanwege de relatief korte duur van de ontwikkelfase zijn er maar een beperkt aantal retrospectives uitgevoerd. Toch waren deze waardevol voor het project. Hieronder volgt een korte samenvatting van de feedback die daaruit naar voren is gekomen.

14.1. Eerste retrospective

Zie Bijlage 8 voor een retrospective whiteboard met meer informatie.

Deze retrospective is volgens de KALM methodiek (FunRetrospectives, n.d.) gedaan tussen de bedrijfsbegeleider Danny Plass – Oude Bos en mijzelf. In het algemeen waren we allebei zeer positief over de voortgang en werkwijze. Er werd ruim genoeg tijd genomen om deze scriptie te schrijven, er werd grondig gewerkt aan onderzoek & ontwikkeling en de voortgangsgesprekken 2x per week werkten uitstekend om obstakels weg te nemen en de voorgang te bewaken. Als actiepunten zijn geformuleerd: zorgvuldiger omgaan met het JIRA backlog en sprints, tweewekelijkse sprint retrospectives houden, proberen om dagelijks een "*git commit*" te maken en op 26 mei de conceptscriptie opsturen naar de bedrijfsbegeleider.

14.2. Tweede retrospective

Zie Bijlage 11 voor een retrospective whiteboard met meer informatie.

Voor de tweede retrospective is de "4 L's" methodiek gebruikt: *liked, learned, lacked, longed for* (FunRetrospectives, n.d.). De positieve toon van de vorige retrospective was ook hier aanwezig. Er was veel vooruitgang gemaakt met het instellen van feedbackregels, er zijn security problemen opgelost en er is goede feedback verkregen over de gerealiseerde functionaliteiten.

De kritiekpunten gingen bij deze retrospective meer over aparte ontwerpkeuzes van vorige ontwikkelaars, waardoor bepaalde oplossingen van de virtuele docent noodgedwongen enigszins "hacky" moesten worden geïmplementeerd. Door drukte met deze scriptie was er ook geen tijd voor een tussentijdse presentatie voor stakeholders; deze is in overleg opgeschoven naar vlak na de deadline van de definitieve scriptie. De actiepunten waren: doordeweeks iets meer tijd nemen voor de scriptie, focus verleggen naar de conceptscriptie, aparte ontwerpkeuzes en "oud of scope" zaken noteren als advies voor vervolgprojecten en een stakeholder meeting inplannen in de week na de deadline van de definitieve scriptie.

14.3. Voortgangsgesprekken met bedrijfs- en afstudeerbegeleider

Gedurende het afstudeerproject zijn regelmatig voortgangsgesprekken gehouden met de bedrijfsbegeleider Danny Plass – Oude Bos en docent afstudeerbegeleider Tristan Pothoven. De feedback die hieruit naar voren kwam, was vrijwel uitsluitend positief: beide begeleiders complimenteren mijn grondige werkwijze t.o.v. onderzoeken en ontwikkelen en vinden dat ze goed diepgaande inhoudelijke discussies met mij kunnen voeren.

14.4. Conclusie

Gezien de productieve retrospectives en de positieve feedback van de voortgangs-gesprekken kan worden geconcludeerd dat de gehanteerde werkwijze goed heeft uitgepakt voor het afstudeerproject. Er is grondig, en consequent vooruitgang geboekt met het onderzoek en de ontwikkeling van de virtuele docent. Daardoor heb ik goed om kunnen gaan met een aantal onverwachte vertragingen, zoals de afwezigheid van een autorisatiesysteem. Er is voldoende tijd ingepland voor het werken aan de afstudeerscriptie, waardoor meer tijd beschikbaar was voor verbeteringen en verwerken van feedback.

15. Test: evaluatie van UI ontwerp

In voorgaande hoofdstukken is te lezen dat een UX ontwerpproces is uitgevoerd om de gebruikersinterface te ontwerpen. Evaluatie en feedback speelt daarbij een cruciale rol. Voorafgaand aan het ontwerpproces is inspiratie opgedaan van een aantal bestaande producten (zie hoofdstuk Bestaande productenanalyse). Op basis hiervan is een initieel ontwerp gemaakt. Vervolgens zijn twee HBO studenten van Saxion uit de beoogde doelgroep betrokken voor usability tests en feedback. Achteraf zijn zij nog regelmatig gevraagd om feedback te geven over specifieke ontwerpen. Dit alles is natuurlijk geen absoluut bewijs dat de UI “goed” is, maar geeft wel een duidelijke indicatie dat het “goed genoeg” is.

16. Test: evaluatie van software

Tijdens ontwikkeling is de virtuele docent voortdurend getest op fouten, ongemakken en ongewenst gedrag. Zo zijn een paar problemen geïdentificeerd en opgelost. Zie bijvoorbeeld hoofdstuk 11.1 voor een voorbeeld. Hierin resulteerde een ongelukkige ontwerpkeuze in ongewenst gedrag toen feedback van de virtuele docent al gedateerd was toen het werd gegenereerd, omdat niet de meest recente data beschikbaar was. Dit soort methodieken om fouten op te sporen, vallen onder de categorie Debugging en vormen een cruciaal onderdeel van softwareontwikkeling.

Naast debugging zijn ook enkele testmethodieken toegepast zoals black-box smoke- en sanity testing om nieuwe features oppervlakkig te testen, usability testing van de UI (zie hoofdstuk 15 hierboven) en enkele development testing methodieken en strategieën zoals debugging en sanity checks. Daarnaast is een meer algemene systeemtest gedaan om het systeem als geheel te testen.

16.1. Systeemtest

Om het voltooide systeem van de Virtuele Docent te testen, is een systeemtest gepland. Tijdens deze test zullen een aantal use cases en scenario's worden doorlopen om te controleren of deze met succes kunnen worden voltooid en of er eventuele onverwachte resultaten op zullen treden. Ter voorbereiding wordt een testplan gemaakt waarin de scenario's en verwachte uitkomst staan beschreven. Tijdens de systeemtest worden deze resultaten opgenomen in een testrapport. Het testplan en -rapport zullen kort worden toegelicht als onderdeel van de afstudeer-verdediging.

De systeemtest is gepland in de twee weken tussen de deadline van de definitieve afstudeerscriptie en de afstudeerverdediging. Het testplan en -rapport zijn dus niet inbegrepen bij deze scriptie zelf. Deze worden bij de eindoplevering van het afstudeerproject beschikbaar gesteld. Zoals genoemd, worden de resultaten van de systeemtest ook kort besproken tijdens de afstudeerpresentatie, samen met feedback van de stakeholder evaluatie die is gepland in dezelfde periode voorafgaand aan de afstudeerverdediging.

16.2. Unit testing

Idealiter zou unit testing ook een belangrijke rol hebben gespeeld. Het voordeel van unit testing is dat dit een goed hulpmiddel is voor automatische regressietests om onbedoelde fouten op te sporen. Echter was hier een dilemma om op te lossen. De voorgangers hebben namelijk nooit een unit testing oplossing gerealiseerd in de front-end of backend. Gezien de prioriteit van de Virtuele Docent is besloten om meer in te zetten op goede debugging- en development testpraktijken, zodat kwaliteit van de oplossing kon worden gewaarborgd in afwezigheid van automatische regressietesten. Denk bijvoorbeeld aan maatregelen om code meer robuust te maken, zoals sanity checks. Ter vervanging van automatische tests, zijn door middel van testpraktijken zoals smoke & sanity testing en een dosis gezond verstand veel denkbare scenario's getest.

16.3. Conclusie over evaluatie

Door regelmatige feedback te zoeken bij eindgebruikers kan worden geconcludeerd dat de UI van genoeg kwaliteit is voor de eerste versie van de virtuele docent. Het gebrek aan regressietesten is goed opgevangen door meer te investeren in debugging, smoke / sanity - en development testing. De resulterende werkwijze voor testing is wellicht wat onorthodox t.o.v. moderne softwareontwikkeling, maar werkt prima voor een eenpersoons afstudeerproject met beperkte tijd en resources. Wel

wordt geadviseerd om alsnog een degelijke unit / regressie testoplossing te ontwikkelen, zodat de kwaliteit van de code kan worden onderbouwd met code coverage – en teststatistieken.

17. Conclusie

"Begeleiding geven aan studenten en feedback automatiseren". Een complex vraagstuk dat in ±20 weken tijd is onderzocht en gedefinieerd, waarvoor productvereisten zijn geformuleerd en een volledig functioneel eerste prototype ontwikkeld. De grote vraag is: is het doel van het project voltooid? De signalen die zijn ontvangen gedurende het project, zijn in ieder geval hoopvol. De feedback die de virtuele docent geeft, is volledig aanpasbaar en wordt op een duidelijke, niet onderbrekende manier aan de gebruikers gecommuniceerd. Na de zomervakantie van 2021 zal de virtuele docent voor het eerst worden ingezet. Dan zal definitief blijken hoe de virtuele docent presteert in het begeleiden van studenten en het vereenvoudigen van het feedbackproces. Er zijn hoe dan ook mogelijkheden om het project nog verder te ontwikkelen; zo is de aard van een eerste prototype. Wat dat is, leest u in het hoofdstuk Aanbevelingen hieronder.

18. Aanbevelingen

Tijdens de ontwikkeling van de virtuele docent zijn een paar zaken vanwege tijdsredenen buiten de scope van het project gelaten. Ook zijn een aantal technische verbeterpunten geïdentificeerd waar toekomstige ontwikkelaars aandacht aan zouden moeten besteden.

18.1. Virtuele docent

De virtuele docent heeft een aantal uitbreidingsmogelijkheden. Ten eerste zou het interessant zijn om meer diverse soorten feedback te verkennen. Bijvoorbeeld op tekst gebaseerde controles die onderzoeksvragen analyseren, of controles op het gebruik van bepaalde onderzoeksmethoden in bepaalde fases en onderzoeksvragen.

Concrete voorbeelden van meer diverse controles

- Is een onderzoeksvraag samengesteld? (twee vragen in één)
- Is een onderzoeksvraag SMART geformuleerd?
- Sluit een onderzoeksvraag aan bij het projectdoel?
- Is een onderzoeksmethode "logisch" om te gebruiken binnen de context van de onderzoeksvraag en projectfase?
    - Wellicht kan hier advies op worden gegeven, bijvoorbeeld "andere groepen gebruiken in dit soort situaties vaak methodes XXX en YYY".

Overige uitbreidingsmogelijkheden

Een andere uitbreidingsoptie voor de virtuele docent is om feedbackregels op basis van opleidings- of leermodule templates in te kunnen stellen. Feedbackregels worden nu globaal ingesteld en gelden voor alle projecten. Echter is bijvoorbeeld een Smart Solutions project heel anders dan een introductie-les over het DOT Framework. En bij bijvoorbeeld de CMD opleiding (Communication & Multimedia Design) heerst een andere beroepscultuur dan bij HBO-ICT, met dito verschil in feedback op projectaanpakken.

Als laatste wordt geadviseerd om de huidige virtuele docent kritisch te analyseren in de praktijk zodat eventuele verbeterpunten kunnen worden geïdentificeerd. Hier was tijdens de ontwikkeling van de virtuele docent weinig gelegenheid voor, met name op het gebied van aansluiting op de beroepspraktijk binnen Saxion.

18.2. Project Approach Tool algemeen

In de Project Approach Tool zijn een aantal punten die aandacht nodig hebben van een HBO-ICT software engineer. Ten eerste zijn bij de ontwikkeling van de front-end Angular app door de voorgangers rare ontwerpkeuzes gemaakt, waardoor de structuur van de gebruikersinterface het een en ander te wensen overlaat. Het advies is om een afstudeerder met ervaring op het gebied van Angular en Node.js server apps de huidige gebruikersinterface en code kritisch te laten analyseren en waar nodig verbeteren.

Het tweede advies is om een unit / regressietest oplossing in zowel de front-end als backend in te richten zodat tijdens ontwikkeling snel en betrouwbaar fouten kunnen worden opgespoord en opgelost. Dit heeft tevens als voordeel dat codekwaliteit kan worden onderbouwd met code coverage - en teststatistieken. Dit kan mogelijk worden gecombineerd met het vorige punt.

Ten derde wordt geadviseerd om de security van het huidige platform kritisch te analyseren. Het is belangrijk dat bekende security flaws die zijn gerapporteerd bij NPM, op worden gelost. Het commando "*npm audit*" kan hierbij helpen. Ook de algemene security van het project moet worden geanalyseerd op bekende security loopholes en exploits om de kans op hacks en datalekken te minimaliseren. Daarnaast is het verstandig om de huidige authenticatie- en autorisatie oplossing te analyseren en verbeteren met bijvoorbeeld multifactor authenticatie. Als het creëren van een docentenportaal gewenst is, is het een optie om de huidige autorisatie-oplossing uit te breiden tot bijvoorbeeld een rechten-gebaseerd systeem.

Als laatste wordt geadviseerd om de bekende problemen met de verbindingspijlen tussen projectfases en de wens voor een drag & drop interactie om onderzoeksmethodes toe te voegen, op te lossen. Deze problemen maken de huidige gebruikersinterface onhandig in gebruik. De aard van deze problemen ligt in het feit dat de huidige ontwerpomgeving een HTML structuur is, waarover de verbindingspijlen worden gerenderd.

De oplossing die voor kwaliteit het beste zou zijn, maar wel meer werk kost, is om de gehele ontwerpomgeving opnieuw te bouwen als een SVG canvas (*"vector graphics"*). Moderne internetbrowsers hebben hier prima ondersteuning voor en dergelijke oplossingen worden ook door populaire javascript bibliotheken zoals Chart.js gebruikt. Een conversie naar een canvas-oplossing opent mogelijkheden om de ontwerpomgeving intuïtiever te maken in het gebruik, vergelijkbaar met bijvoorbeeld het programma Draw.io.

Om alles samen te vatten:

1. Virtuele docent feedbackregel systeem uitbreiden met meer diverse controles.
2. Virtuele Docent templating systeem voor feedbackregels voor verschillende profielen, opleidingen en hogescholen.
3. Kritische analyse van Virtuele Docent in een praktijkomgeving, bijvoorbeeld als pilot tijdens lesmodules.
4. Wellicht een docentenportaal voor meer alternatieve feedbackmogelijkheden
5. Refactor van de front-end Angular UI structuur en - oplossingen
6. Backend codekwaliteit analyseren en waar nodig refactoren
7. Unit & regressietest oplossing introduceren in front- en backend.
8. Security analyseren van het gehele platform
9. True multifactor authenticatie integreren
10. Autorisatiesysteem uitbreiden met rechten (in combinatie met punt 4)
11. Verbindingspijlen en overige secondaire problemen oplossen
12. Wellicht de ontwerpomgeving ombouwen naar een SVG canvas.

19. Samenvatting

Met de Project Approach Tool van het Saxion Ambient Intelligence lectoraat kunnen studenten een projectaanpak ontwerpen op basis van het DOT Framework. Dit is een raamwerk om stapsgewijs, goed onderbouwd en grondig antwoord te geven op onderzoeksvragen. Studenten hebben echter vaak weinig ervaring met het DOT Framework en hebben daardoor moeite om bewuste keuzes te maken over hun projectaanpak. Met het afstudeerproject “Virtuele Docent” is een systeem gerealiseerd dat studenten hierbij moet helpen, door een aantal veel voorkomende feedbackpunten te automatiseren.

Na een uitgebreid onderzoek bestaande uit enkele interviews, een UX ontwerpproces en een technisch & functioneel ontwerp is een eerste prototype ontwikkeld van de “virtuele docent”. Dit feedbackmechanisme is in staat om de projectaanpak van een gebruiker van de Project Approach Tool te analyseren, met als doel om een aantal veel voorkomende soorten feedback te genereren. Studenten kunnen deze feedback lezen en waar mogelijk, snel verwerken. Ook kunnen docenten instellen wat voor feedback wordt gegeven.

In de nieuwe versie van Project Approach Tool is een eerste prototype van de Virtuele Docent geïntroduceerd. Dit eerste prototype is bewust simpel gehouden uit tijd- en complexiteitsoverwegingen. Met de mogelijkheid tot het automatiseren van veel voorkomende feedback zijn docenten in staan om instelbare feedbackregels aan te maken, waardoor studenten meer begeleiding krijgen bij het maken van een projectaanpak. Hierdoor krijgen zij meer informatie over de betekenis van hun ontwerpkeuzes en kunnen ze beter bewuste keuzes maken.

Naast de introductie van de Virtuele Docent zijn tijdens ontwikkeling een aantal kritiekpunten over het bestaande platform van de Project Approach Tool geïdentificeerd. In hoofdstuk 18 Aanbevelingen is daarover meer informatie te vinden. Als laatste wordt aangeraden om de secondaire problemen die zijn beschreven in hoofdstuk 3.2, te onderzoeken en op te lossen. Ter behoeve van het voltooien van de Virtuele Docent zijn deze problemen om tijdsredenen niet opgepakt.

20. Reflectie

Voor mijn afstudeerproject heb ik een opdracht uitgevoerd bij het lectoraat Ambient Intelligence van Saxion. Vanwege de COVID-19 pandemie heb ik net als de meeste medewerkers van het lectoraat volledig op afstand gewerkt vanuit huis. Voor mijn afstudeerproject heb ik een "virtuele docent" ontwikkeld die veelvoorkomende feedback automatisch kan geven. Daarnaast kan de virtuele docent een aantal statistische kenmerken van een projectaanpak berekenen, zoals de mate van Triangulatie en de balans tussen gebruikte onderzoeksstrategieën.

Anders dan bij de meeste studieprojecten en de beroepspraktijk, is het afstudeertraject iets dat studenten alleen moeten uitvoeren. Daarbij laten ze zien dat ze individuele beroepsbekwaamheid hebben om te kunnen functioneren op HBO niveau. Voor mezelf heb ik daarom een aantal maatregelen genomen zoals het aanmaken van een backlog op het platform Jira, proactief feedback zoeken over kwaliteit van geleverd werk en documentatie en rekening houden met bestaande software ontwerppatronen en principes zoals *"loose coupling"* en het *"strategy pattern"*. Daarbij heb ik een aantal lastige keuzes moeten maken die haaks staan op wat ik normaliter bij softwareprojecten doe, omdat deze buiten de afbakening van het project vielen of omdat er taken met hogere prioriteit waren. Een voorbeeld hiervan is de keuze om geen unit tests te schrijven. Dit zou ik normaliter wel doen, ware het niet dat er belangrijkere taken waren. Om het gebrek aan unit testing op te vangen heb ik meer gefocust op alternatieve manieren om de oplossing te evalueren.

Desalniettemin ben ik zeer te spreken over het resultaat. In 20 weken tijd heb ik een zeer complex en abstract vraagstuk onderzocht, gedefinieerd, een oplossing ontworpen en met succes gerealiseerd. Natuurlijk moeten daarbij soms lastige keuzes worden gemaakt; dat is in de beroepspraktijk niet anders. Toch heb ik dankzij het grondige voorafgaande onderzoek niet echt te maken gehad met grote tegenslagen. Het was een verstandige keuze om hier meer tijd aan te besteden; zo heb ik een aantal moeilijke ontwerp- en realisatiekeuzes kunnen maken die anders veel langer hadden geduurd.

Als ik hetzelfde project nogmaals zou doen, zou ik veel eerder beginnen met het organiseren van retrospectives en code reviews. Vanwege tijdbeperkingen en het feit dat het een eenpersoonsproject betreft, heb ik daar nu niet veel tijd aan kunnen besteden en zijn er relatief weinig retrospectives gehouden. Ondanks dit kleine gebrek vind ik zelf dat ik uitstekend heb gefunctioneerd. De feedback van mijn begeleiders beaamt dit.

21. Bronvermelding

Angular / Google. (sd). *Angular versioning and releases*. Opgehaald van Angular.io: https://angular.io/guide/releases

Angular / Google. (sd). *Angular: the modern web developer's platform*. Opgehaald van https://angular.io/: https://angular.io/

Buchanan, R. (1992). Wicked Problems in Design Thinking. *Design Issues, Vol. 8, No. 2*, Pagina 5-21. doi:10.2307/1511637

Fowler, M. (2005, december 20). *Fluent Interface*. Opgehaald van https://martinfowler.com/bliki/FluentInterface.html

FunRetrospectives. (sd). *4 Ls: Liked - Learned - Lacked - Longed for*. Opgehaald van https://www.funretrospectives.com/the-4-ls-liked-learned-lacked-longed-for/

FunRetrospectives. (sd). *KALM - Keep, Add, Less, More*. Opgehaald van Fun Retrospectives: https://www.funretrospectives.com/kalm-keep-add-more-less/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* VS: Addison-Wesley.

Icons8. (2021, mei 21). *About Lunacy*. Opgehaald van Icons8 docs: https://docs.icons8.com/about/#what-is-lunacy

Peeters, W. (2019, april 4). *Effectief nakijken met de feedbacklegenda*. Opgehaald van Vernieuwenderwijs: https://www.vernieuwenderwijs.nl/efficient-nakijken-met-de-feedbacklegenda/

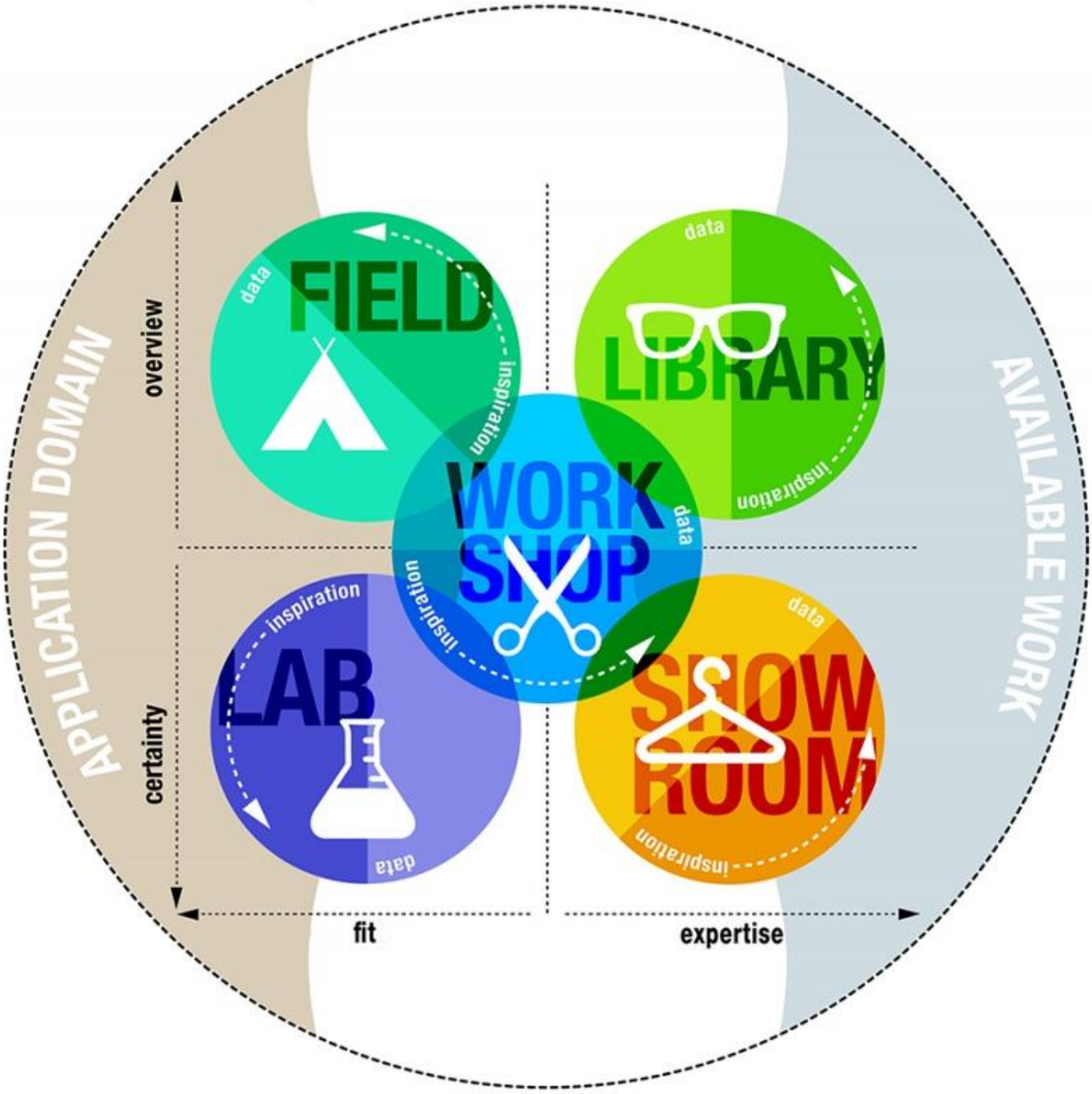
Saxion. (sd). *Ambient Intelligence*. Opgehaald van https://www.saxion.nl/onderzoek/smart-industry/ambient-intelligence

Socket.io. (sd). *Socket.IO 4.0 is here*. Opgehaald van Socket.io: https://socket.io/

Wikiwijs / Saxion. (2020, september 14). *1.1.2: Aanpak bepalen met het DOT Framework*. Opgehaald van Wikiwijs: https://maken.wikiwijs.nl/156085/1_1_2_Aanpak_bepalen_met_het_DOT_Framework#!page-5768841

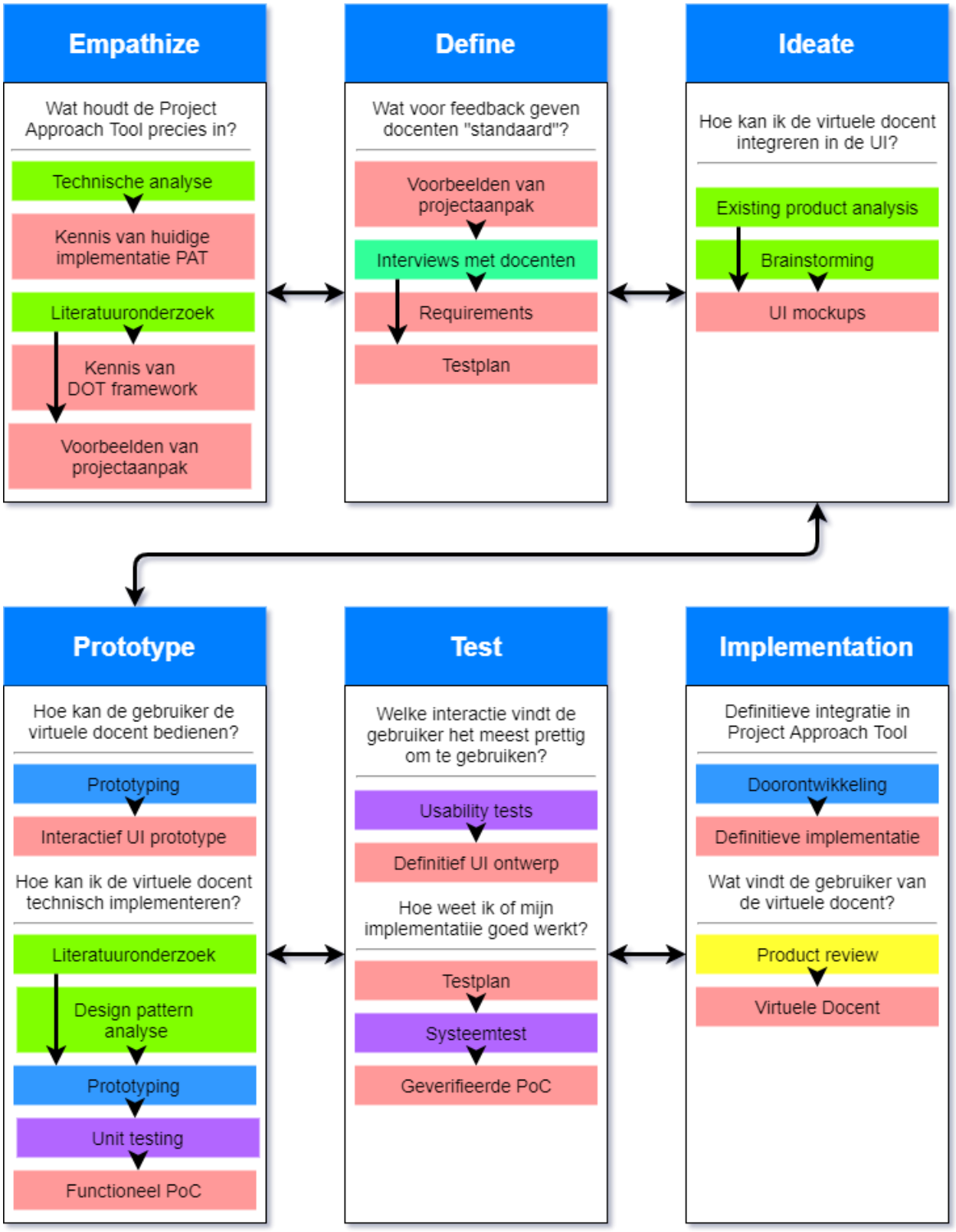
Wikiwijs / Saxion. (2020, september 14). *Waarom: 3 afwegingen*. Opgehaald van 1.1.2: Aanpak bepalen met het DOT Framework: https://maken.wikiwijs.nl/156085/1_1_2_Aanpak_bepalen_met_het_DOT_Framework#!page-6249757

Bijlage 1. DOT Framework drielagenmodel



Afbeeldingsbron:
https://maken.wikiwijs.nl/156085/1_1_2_Aanpak_bepalen_met_het_DOT_Framework#!page-6249757

Bijlage 2. Projectaanpak “Virtuele Docent”



Bijlage 3. Analyse van projectmanagementsoftware

Om de requirements te kunnen beheren, zijn een aantal projectmanagement-programma's overwogen. Dit zijn Trello, Azure DevOps, Atlassian Jira, Gitlab en Github Projects. Daarbij is gelet op een aantal criteria:

1. Ondersteuning voor git-flow
2. Koppeling met externe git repository
3. Taken met hierarchie "epics-issues-taken"
4. Agile / scrum ondersteuning

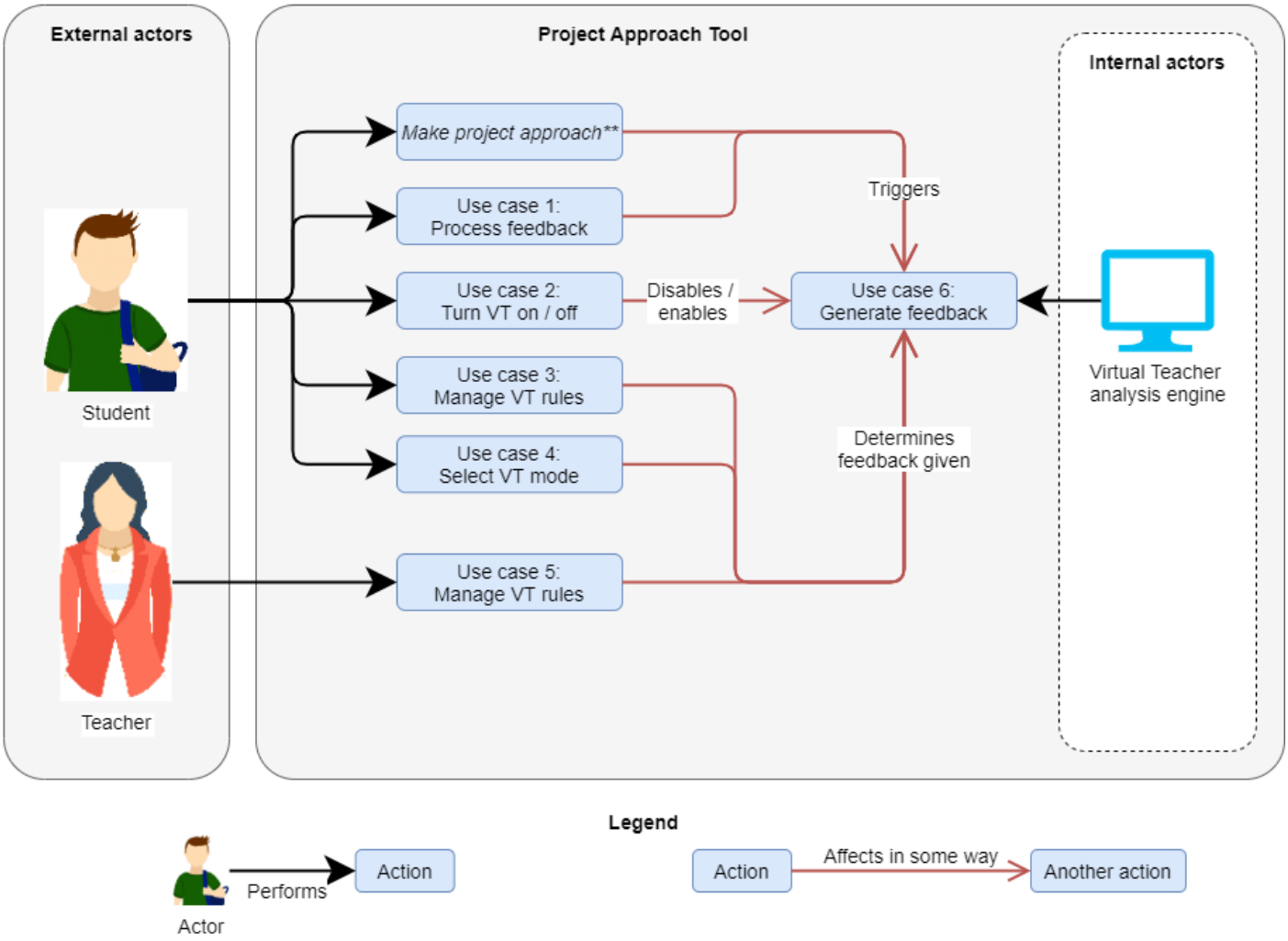
	Github projects	MS Azure DevOps	Trello	Atlassian Jira	Gitlab
Git-flow ondersteuning	✓	✓	X	✓	✓
Externe git repository	X	✓	X	✓	?
Epics-issues-taken	X**	✓	X	✓	✓
Agile / scrum ondersteuning	X**	✓	X	✓	✓

***Github projects kan wel integreren met externe projectmanagement platformen.*

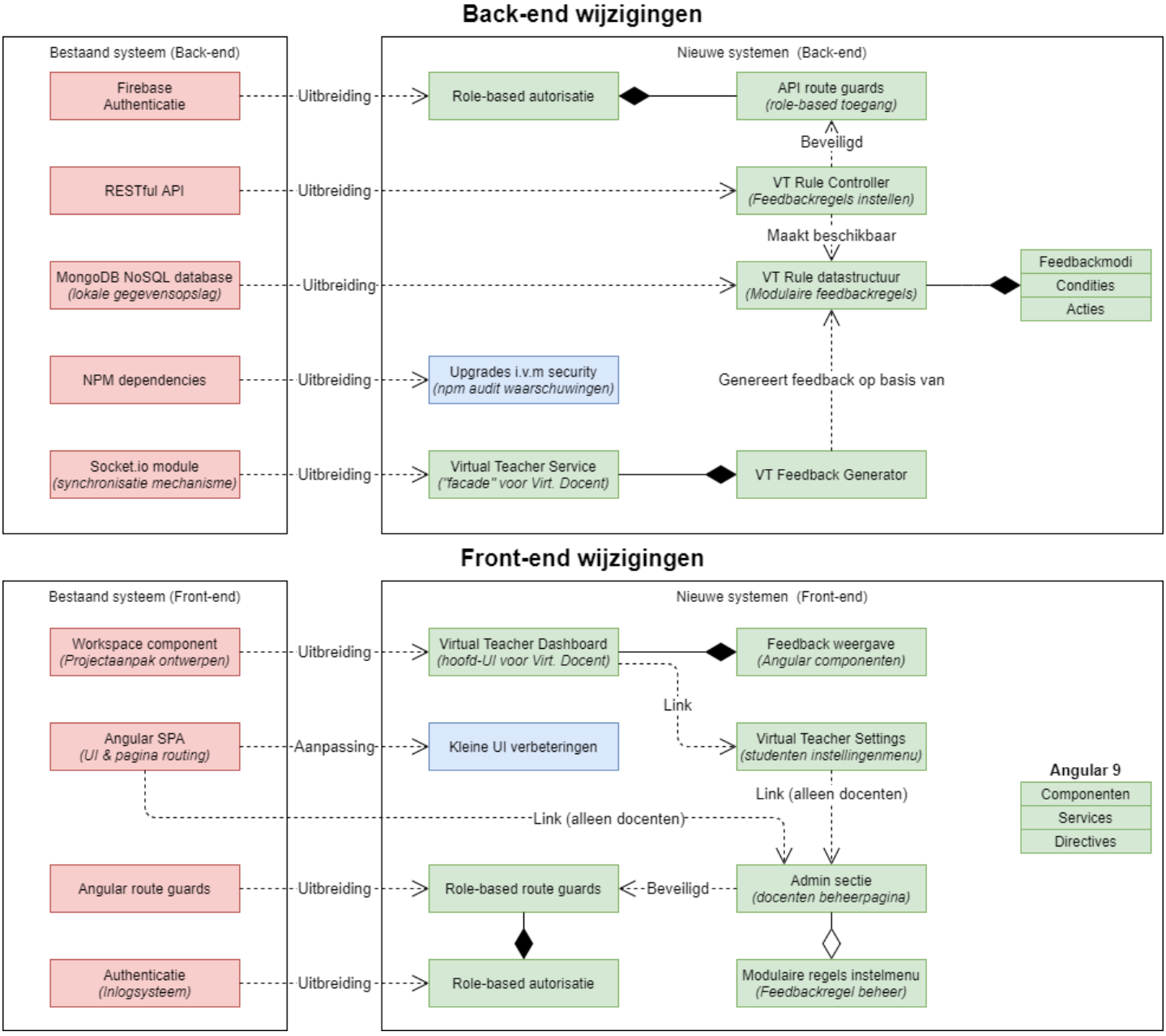
In de matrix hierboven is te zien dat Trello, hoewel het uit eigen ervaring een prima pakket is, niet goed aansluit op de gestelde criteria. Voor Github Projects geldt hetzelfde, wat niet verbazend is gezien Github in de eerste plaats geen projectmanagement platform is. Voor Gitlab geldt dat het een uitgebreide ontwikkelsuite heeft inclusief projectmanagement (Agile Delivery), maar niet met zekerheid kan integreren met Github.

Zowel Azure DevOps als Atlassian Jira vervullen alle criteria. Omdat ik persoonlijk al ervaring heb met Azure DevOps, heb ik meer interesse om Jira beter te leren kennen. Daarnaast heb ik over Jira zeer positieve dingen gehoord. Daarom heb ik voor Jira gekozen als projectmanagementomgeving.

Bijlage 4. Overzicht van use cases



Bijlage 5. Overzicht van wijzigingen



Bijlage 6. Codevoorbeeld koppeling websocket

Onderstaande voorbeelden zijn incompleet: alleen relevante code is weergegeven.

SocketModule.js *(uitbreiding van bestaande code)*

```
let connectEventEmitter = new events.EventEmitter();
//Voorbeeld van een connectie event:
socket.on("connectRoom", async (event) => {
    if (!rooms.find((room) => room.id === event.room)) {
        let workspaceEventEmitter = new events.EventEmitter();
        rooms.push({... , emitter: workspaceEventEmitter });
    }
    //uitbreiding: signaleer nieuwe room in connectEventEmitter
    connectEventEmitter.emit('joinWorkspace', rooms[index]);
}
//Voorbeeld van een workspace-wijziging socket event:
socket.on("moveGroup", (event) => {
    //uitbreiding: signaleer workspace wijziging event in workspaceEventEmitter.
    emit(socket, 'moveGroup', event.data, event.workspace);
});
```

server.js

```
//socketModule is de connectEventEmitter (zie vorige voorbeeld)
const socketModule = require("./socketModule")(io);
socketModule.on('joinWorkspace', (workspace) => {vt.tryCreateVtInstance(workspace)})
socketModule.on('leaveWorkspace', (workspace) => {vt.tryRemoveVtInstance(workspace)})
```

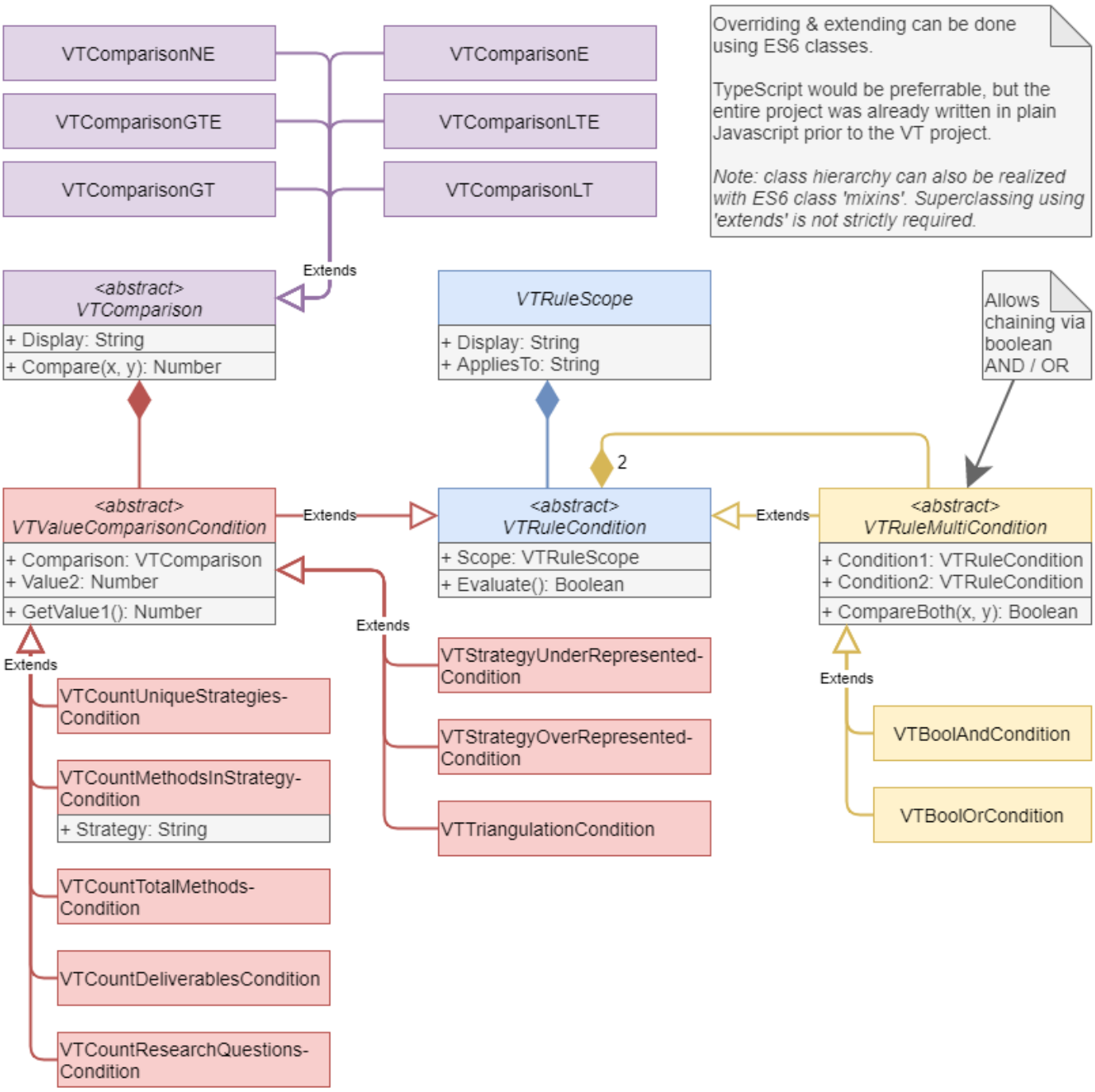
virtualTeacherModule.js

```
//In de praktijk is deze code wat complexer i.v.m deduplicatie
//en opruimen van (oude) virtualTeacherService instanties.
tryCreateVtInstance(workspace, workspaceEvents) {
    let vtIndex = vtInstances.findIndex(x => x.workspace.id == workspace.id);
    if (vtIndex >= 0) { /*Ruim oude VT instantie op als nodig*/ }
    const vt = new VirtualTeacherService(workspace, workspaceEvents);
    vtInstances.push(vt);
}
tryRemoveVtInstance(workspace) {
    let vtIndex = vtInstances.findIndex(x => x.workspace.id == workspace.id);
    if (vtIndex >= 0) { /*Ruim oude VT instantie op als nodig*/ }
}
```

virtualTeacherService.js

```
constructor(workspace) {
    this._workspace.emitter.on('enableVirtualTeacher', this.start);
    this._workspace.emitter.on('disableVirtualTeacher',this.stop);
    this._workspace.emit('virtualTeacherReady');
}
start() { /*start met luisteren naar workspace-wijziging socket events*/ }
stop() { /*stop met luisteren naar workspace-wijziging socket events*/ }
```

Bijlage 7. Volledig klassendiagram “regel condities”



Bijlage 8. Codevoorbeeld “VTRule” (backend)

De onderstaande code is incompleet: alleen relevante code is weergegeven.

vtRule.js

```
class VTRule {
    constructor(phases, condition, action) { /*sanity checks & fill in local fields*/    }

    evaluate(currentFeedbackMode, data) {
        //sanity checks
        if (!this.phases.find(x => x.identifier == currentFeedbackMode)) return;
        if (!this.condition.scope.evaluate(data)) return undefined;
        const evaluationResult = this.condition.evaluate(data)
        if (!evaluationResult) return undefined;

        //Condition passed; create a feedback item.
        let dataType = getDataType(data);
        const result = { type: dataType, id: data.id, action: this.action }
        if (result.action instanceof VTRuleActionShowMetrics) {
            //In case of calculated metrics, we want the metrics to be sent to the client.
            result.action.metrics = evaluationResult;
        }
        return result;
    }
}
```

vtRuleCondition.js

```
class VTRuleCondition {
    get scope() { return this._scope; }
    evaluate(data) {
       //Unfortunately, there’s no such thing as “abstract classes” in plain javascript.
       throw Error("function VTRuleCondition.Evaluate must be overridden in all derived types.");
    }
}
```

vtTriangulationCondition.js

```
class VTTriangulationCondition extends VTRuleCondition {
    evaluate(data) {
        if (VTConditionChecks.isWorkspace(data)) {
            return this.calculateTriangulationMetricsInWorkspace(data);
        }
        else if (VTConditionChecks.isGroup(data)) {
            return this.calculateTriangulationMetricsInGroup(data);
        }
        else return undefined;
    }
    this.calculateTriangulationMetricsInWorkspace(data) { /*calculate triangulation metrics*/ }
    this. calculateTriangulationMetricsInGroup(data) { /*calculate triangulation metrics*/ }
}
```

Bijlage 9. Codeerstandaarden

Bij de ontwikkeling van de Virtuele Docent is een codeerstandaard gehanteerd die voortkomt uit enkele jaren van persoonlijke ontwikkeling met meerdere programmeertalen. Bij deze standaard zijn enkele concepten zoals duidelijkheid, leesbaarheid en eenmalige typering gebruikt om de code overzichtelijk te houden.

Scoping & vertakkende code (bijv. if statements)

- Vermijd geneste code blokken waar mogelijk.
- Gebruik `return`, `break` en `continue` statements om vroegtijdig codeblokken of methode-aanroepen af te sluiten.
- Beperk lengte van scopes & codeblokken. Vaak is het mogelijk om code anders te structureren zonder de functionaliteit aan te passen.
- Vermijd grote try-catch blokken tenzij het echt niet anders kan.

Naamgeving

- Vermijd afkortingen in variabele-, methode- en klasse namen (uitzondering voor prefixes m.b.t. groepering van klassen), bijvoorbeeld "`condition`" in plaats van "`con`"
- Geef variabelen, methodes en klassen een duidelijke, concrete naam die het doel beschrijft, bijvoorbeeld "`newCondition`" in plaats van "`condition`".
- Bij tijdelijke variabelen zoals in lambdas / arrow functies of for-loops is het toegestaan om één letter variabele namen zoals "x" of "i" te gebruiken.

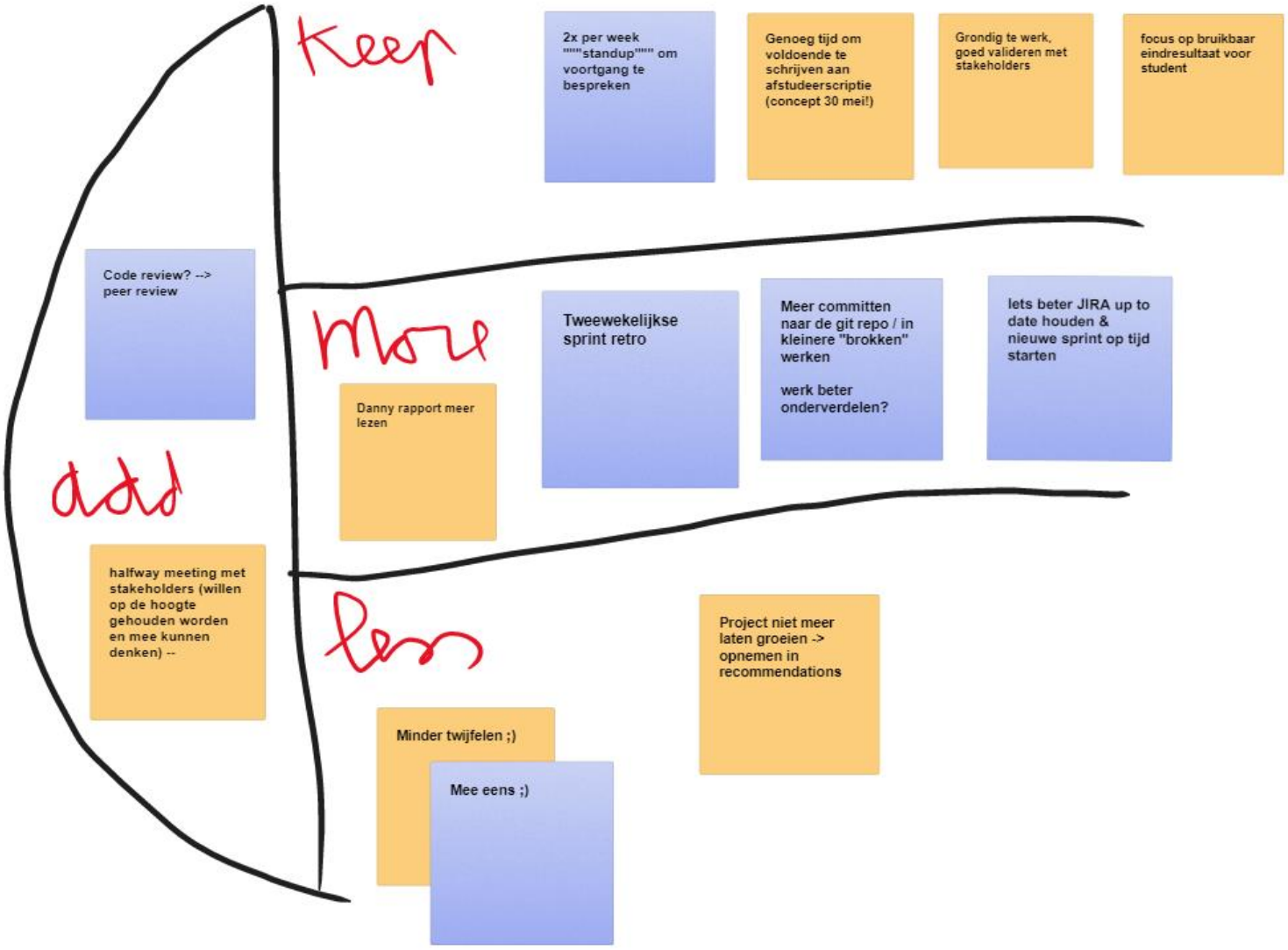
Typering

- Vermijd dubbele declaratie van het type van een variabele. Impliciete typering of heldere naamgeving zijn vaak al duidelijk genoeg.
- Bij generic types, gebruik type inference tenzij dit compilerfouten geeft. Bijvoorbeeld "`var x = Foo(bar)`" in plaats van "`var x = Foo<Bar>(bar)`".
- Vermijd onnodige overerving. Compositie en Aggregatie bestaan ook.

Overig

- Wanneer dezelfde code op drie plekken hetzelfde wordt geschreven, overweeg om dit af te splitsen naar een aparte methode of klasse.
- Vermijd overmatig gebruik van commentaar. Als veel commentaar nodig is, is je oplossing wellicht te complex.
- Voorzie klassen en openbare (public) functies waar nodig van jsdoc / summary commentaar, maar ga ook uit van goede naamgeving.
- Gebruik waar nodig "sanity checks" om parameterwaardes te controleren.

Bijlage 10. 1e sprint retrospective



- Zorgvuldiger omgaan met JIRA sprints / backlog en code commits.
  - Maandagochtend van een nieuwe sprint een sprint in Jira openen.
  - Op dat moment ook naar Jira kijken.
- Tweewekelijkse sprint retro's inplannen.
- Dagelijks werkende commit maken
- Voor code reviews Hugo / Rudo vragen
- Datumprikker maken voor halfway meeting
  - Gebaseerd op de agenda's van de docenten waarvan ik wel bij de agenda's kan.
  - Ook Miriam Losse en Kasper Kamperman vragen.
- Woensdag 26 mei conceptversie voor feedback naar Danny sturen.

Bijlage 11. 2e sprint retrospective

*VT = Virtual Teacher

concrete follow-up actions (or inactions):

- doordeweeks meer tijd inplannen voor verslag
- focus nu op conceptverslag
- out of scope stuff noteren als recommendations voor vervolg
- stakeholder meeting in wk 4.7-8