Plan van Aanpak

Team 10
Kevin Damen 1660811
Remco Nijkamp 1657833
Jordan Ramirez 1652760
Jeroen Kok 1666479
concept nr. 0.4

1. Inleiding

De was, het wordt al voor decennia gedaan. Voor vele jaren werd het gedaan met gewoon de hand. Maar begin 20ste eeuw is de uitvinding gemaakt die dat allemaal een stuk makkelijker maakte: “De wasmachine”. Nu is het doel om deze alledaagse activiteit nog gemakkelijker te maken door middel van een “Webapplicatie”.

In opdracht van de Hogeschool Utrecht moet in blok 2 van het tweede jaar in een team een opdracht, die door de klant geven is uit gevoerd worden. In dit project moet een webapplicatie worden gemaakt die een wasmachine moet kunnen aansturen. Dit wordt in een aantal stappen gedaan.

Ten eerste wordt er een interview gehouden met de klant. Hierdoor wordt informatie verkregen over wat de klant verwacht van het product dat aan het einde van het project moet worden opgeleverd. Met de informatie verkregen uit dit gesprek wordt dan de requirements architectuur gemaakt.

Als dit allemaal klopt dan gaat de aandacht naar de ondersteunende modellen voor het eindproduct. Zo zal een use case model worden gemaakt, waarin een duidelijke beeld wordt geschept wat de gebruiker met het product kan doen, activity diagram die duidelijk maakt welke verbanden er tussen de activiteiten liggen en contraints model is een tabel die een voorwaarden aangeeft hoe goed het systeem kan functioneren.

Ten derde wordt er een solution architectuur gemaakt. Hieronder vallen een klasse diagram met alle klassen en hun functies van programmeer code, een concurrency model die laat zien hoe interactie tussen de taken binnen het systeem plaatsvinden, en een dynamic model, waarin de manier waarop objecten intern functioneren en hoe deze objecten reageren op events, wordt vastgelegd.

Wanneer deze architecturen klaar zijn, wordt begonnen met het programmeren en testen van de code. Als tijdens de voorgaande stappen de architectuur goed is gemaakt, zou dit relatief snel en makkelijk moeten gebeuren. Mocht echter de architectuur niet helemaal kloppen, zal dit blijken tijdens de vertaling naar code. Dan zal niet alleen de code maar ook de architectuur flink op de schop moeten worden genomen. Zodra de code is gemaakt moet deze nog getest worden in vele situaties. Blijkt dit allemaal te werken dan moet nog met de klant worden gekeken of het systeem ook daadwerkelijk aan de functionele eisen voldoet en kan het eindproduct worden ingeleverd.

Het uiteindelijke doel van de opdracht is niet alleen het opleveren van een goed product, maar met name ook het leren werken binnen dit soort projecten. Zo is het een belangrijk leerdoel om met een klant op tafel te gaan zitten en zijn wensen om te kunnen zetten in functionele eisen. Deze eisen moeten vervolgens verwerkt kunnen worden in een architectuur van waaruit het eindelijke product zal ontstaan. Ook de manier van samenwerken en de manier van ontwikkelen zijn belangrijke leerdoelen.

2. Onderzoek

Voor dit project zijn er verschillende onderwerpen waarnaar onderzoek gedaan moet worden.
Bij de aanvang van het project was bekend dat er een webapplicatie en besturingssysteem moest worden gemaakt voor een wasmachine.

Het is van belang om een redelijk inzicht te krijgen in de werking van een wasmachine, alvorens met de opdrachtgever in zee te gaan. Door thuis in gesprek te gaan met gebruikers van een wasmachine en op het internet informatie op te zoeken over eventuele wasprogramma's kan een redelijk beeld worden gevormd. Dit is voldoende voor dit project.

Vervolgens zal informatie moeten worden verzameld over de functionele eisen van de webapplicatie voor de wasmachine. Om deze informatie te verkrijgen zal een interview worden afgenomen bij de opdrachtgever (in dit geval een docent). Na afloop van het interview zal een terugkoppeling plaatsvinden met de opdrachtgever om te checken of alle zaken goed zijn begrepen door het team.

Na de terugkoppeling met de opdrachtgever zal worden gewerkt aan twee architecturen die helpen de productie van het eindproduct in goede banen te leiden. Deze architecturen zijn de requirements architectuur, waarin vooral in kaart wordt gebracht wat het systeem allemaal kunnen en hoe goed het systeem moet functioneren, en de solution architectuur. Hierin wordt vooral gekeken hoe de eisen die uit de requirements architectuur naar voren zijn gekomen, moeten worden verwerkt in het systeem. Dit is als het ware de "oplossing".

Verderop in het project zullen experimenten en testen moeten worden uitgevoerd om erachter te komen of bepaalde delen van het systeem naar behoren functioneren. Zo zal met name de socket server verbinding uitvoerig getest moeten worden om er zeker van te zijn dat de verbinding tussen de wasmachine en de webserver goed loopt.

Het is ook belangrijk een aantal noodsituaties te simuleren om te testen of het systeem hiermee op redelijke wijze om kan gaan. Zo kan bijvoorbeeld getest worden hoe het systeem reageert als de stroom wegvalt halverwege een wasprogramma.

3. Producten en modellen

Te samen met het eindproduct zelf zullen een aantal andere producten worden opgeleverd. Hieronder vallen met name modellen die nodig zijn om het project in goede banen te leiden. Het eerste belangrijke product dat zal worden opgeleverd is de MoSCoW. Dit is een document met daarin omschreven aan welke eisen het product zal moeten voldoen. Tevens staan hierin dingen die graag in het product verwerkt worden, maar enkel worden meegenomen als er tijd genoeg is om dit uit te voeren.
De MoSCoW wordt opgesteld naar aanleiding van het interview dat met de opdrachtgever gehouden zal worden. Het is dan ook belangrijk dat er een terugkoppeling plaats zal vinden richting de opdrachtgever om te checken of de eisen goed begrepen en verwoord zijn.

Naar aanleiding van dit document en het interview zullen een aantal modellen worden opgesteld die samen de Requirement Architectuur vormen.
De modellen die hieronder vallen zijn een Use Case Diagram, een Activity Diagram en een Contraints Model.

Het Use Case Diagram is een diagram met daarin een aantal use cases vastgelegd. Dit diagram zorgt voor structuur van de verschillende acties die moeten worden uitgevoerd door het systeem. Een enkele use case wordt geschreven als een samenhangende reeks van acties die worden uitgevoerd door het systeem om een doel van een actor te bereiken. Dit betekent dat het verband tussen de actoren en de acties ook goed zichtbaar is. Per use case wordt verder een korte omschrijving gegeven met daarin vermeld de naam van de use case, het doel, de pre-condities, de postcondities en de uitzonderingen.

Een Activity Diagram is een diagram dat het verband tussen verschillende activiteiten binnen een systeem of proces weergeeft. Een activity diagram begint altijd bij een initial node en eindigt al dan niet met een final node.

Het constraints model is een tabel met daarin een aantal constraints uitgezet. Een constraint is als het ware een niet-functionele eis die voorwaarden stelt aan hoe goed een systeem bepaalde functies uit moet voeren. Er zijn veel verschillende soorten constraints, onder andere die te maken hebben met performance, accuracy, resource use, availability en reliability. Zo kunnen bij resource use eisen worden gesteld aan de hoeveelheid geheugen die een systeem mag gebruiken.

Zodra de requirement architectuur klaar is, zal gekeken worden hoe dit alles zo goed mogelijk verwerkt kan worden in het eindproduct. Hiervoor zal een solution architecture worden opgesteld. Deze bevat een Class Diagram, een Concurrency Model en een Dynamic Model.

In een Class Diagram (klassendiagram) wordt de statische structuur tussen de objecten binnen een systeem vastgelegd. Het beschrijft welke typen objecten er zijn, de zogenaamde klassen. En welke attributen en operaties deze klassen bevatten. Tevens zijn de relaties

tussen de klassen duidelijk zichtbaar. De structuur wordt statisch genoemd, omdat deze tijdens de werking van het systeem niet meer kan veranderen.

Het Concurrency model beschrijft voor taken binnen het systeem hoe de interactie tussen deze taken plaats moet vinden. Zo zijn er soms regels nodig voor resource gebruik als twee of meer taken hun resources delen. Dit diagram geeft dus een goed beeld van de aanwezige taken en de manier waarop ze met elkaar communiceren.

In een Dynamic model wordt de manier waarop objecten intern functioneren en hoe deze objecten reageren op events, vastgelegd. Er wordt dus in beeld gebracht welke acties een event, afhankelijk van de toestand waarin een object zich verkeert, tot gevolg heeft. De beschrijving van de events in bepaalde toestanden en welke acties hieruit voortkomen, wordt vastgelegd in een State Transition Diagram (STD). In een STD wordt dus het gedrag van een object vastgelegd.

4. Methode van kwaliteitsbewaking

Aan de hand van het interview dat in de tweede week van het project gehouden zal worden, zullen veel eisen naar voren komen waaraan het eindresultaat zal moeten voldoen. Natuurlijk zullen er hiernaast, mede door de teamleden zelf, nog eisen worden opgesteld, maar de eisen van het interview zullen leidend zijn.
Bij de testen die later in het project uitgevoerd gaan worden, zullen deze eisen als maatstaf gebruikt worden.

4.1 Documenten

De gemaakte documenten worden altijd geheel doorgelezen door alle teamleden. Hier kan dan tijdens de eerst volgende vergadering commentaar over geleverd worden zodat de punten van kritiek besproken kunnen worden. Dit zorgt ervoor dat alles meerde keren is doorgelezen en de kans op fouten minimaal wordt. Tijdens het doorlezen dient er zowel gekeken te worden naar spelling en grammatica.

4.2 Code

Tijdens het maken van de code voor dit project moet de documentatie bijgehouden worden door de persoon die het heeft geschreven. Dit moet gedaan worden volgens het doxygen formaat zodat aan het einde van het project de documentatie van de code netjes gegenereerd kan worden. Daarnaast is het van belang dat iedereen alle code kent en weet wat het precies doet. Door elkaars code te reviewen kunnen ook sneller fouten worden ontdekt en voorkomen.

4.3 Testprocedure

Voor de testen die in de projectweek zullen worden uitgevoerd, om te kijken of aan de eisen is voldaan, maar ook om losse stukken code te testen, zal verslag worden uitgebracht. Om te zorgen dat de documentatie hiervan op een goede manier verloopt, en dat de testen zo worden uitgevoerd en gedocumenteerd, dat ze heruitvoerbaar zijn, is het belangrijk dat alle teamleden bij alle tests aanwezig zijn. Zo zullen minder snel zaken over het hoofd worden gezien en kan beter een discussie plaatsvinden over de uitkomst van eventuele tests. Terugkoppeling met elkaar staat hierin dus voorop. Voor de testprocedure’s wordt een documentatie template gemaakt zodat alle test volgens dezelfde methode worden uitgevoerd.

5. Projectinrichting

5.1 Organisatie projectteam

Om het project in goede banen te leiden, worden aan het begin van het project duidelijke afspraken gemaakt over ieders verantwoordelijkheden en over de samenwerking.
Binnen het team is een projectleider aangesteld, dit is Kevin geworden.
De projectleider is verantwoordelijk voor het algemene verloop van het project. Hij zal hierin voornamelijk een communicerende rol fungeren, zowel binnen het team, als naar buiten.

Van alle teamleden wordt verwacht dat ze zich aan hun afspraken houden en hun taken zullen uitvoeren. Mocht dit niet gebeuren dat zal dit worden besproken en zullen er maatregelen genomen worden. Deze staan beschreven in het teamcontract.

Alle leden zijn verantwoordelijk voor alle documenten en andere zaken die worden opgeleverd. Het is de bedoeling dat iedereen op de hoogte blijft van alles en dat ook veel controle zal plaatsvinden. Dit om de kwaliteit van de documenten te waarborgen en te zorgen dat niemand achter raakt.

5.2 Vergaderingen

Tijdens het houden van vergaderen zijn twee taken die verdeeld moeten worden, de voorzitter en de notulist. Er is voor gekozen deze rollen te rouleren per vergadering zodat iedereen beide rollen een aantal keer zal vervullen. Dit is met name gekozen met het oog op leervaardigheid, maar ook om de flow binnen het team fris te houden.

De vergaderagenda voor de eerst volgende agenda dient opgesteld te worden door de voorzitter. Deze moet op tijd worden gemaakt en beschikbaar worden gesteld voor alle teamleden.

Elke week wordt tijdens de vergadering besproken wat er allemaal voor de komende week op de planning staat. Als hier een duidelijk beeld van is geschept, zullen de opdrachten in kleinere taken worden opgedeeld en verdeeld over de teamleden. Elk teamlid werkt in principe aan zijn eigen deel en natuurlijk de delen die als gezamenlijke taken zijn aangewezen. Als de taken af zijn, wordt met het gehele team gekeken naar alle aparte opgeleverde documenten en wordt gesproken wat een ieder hiervan vindt. Natuurlijk kan tussentijds om hulp worden gevraagd bij andere teamleden als een taak te moeilijk of onduidelijk mocht blijken. Het kan ook voorkomen dat onderling wordt geruild van taken als dit beter uit mocht komen. Wat dat betreft is er dus voor gekozen zo flexibel mogelijk te werk te gaan.

5.3 Documentatie

Alle documentatie die gemaakt gaat worden tijdens dit project komt op de Github repositorie te staan. Hierdoor staan al onze documenten op dezelfde plek en kunnen we ook gebruik maken van versie beheer. Op de Github is een mappenstructuur aangemaakt waarin alle documenten verdeeld kunnen worden. Hierdoor is er meer overzicht in alle documenten. De document structuur die aangehouden wordt is:

- Documenten
- Modellen
- Software
- Teamcontract
- Urenverantwoording
- Vergaderingen

Dit zijn de hoofdmappen op de Github. Het is de bedoeling dat bestanden onder de juiste mappen worden neergezet. Als er in één van de hoofdmappen bestanden kommen te staan die samen onder één categorie kunnen staan wordt hiervoor een sub-map gemaakt waarin deze documenten samen komen te staan.

De code van dit project staat ook op de Github. Bij de code van dit project moet ook de documentatie hiervoor worden bijgehouden. Dit wordt gedaan door comments in de code. Dit moet gedaan worden in het doxygen formaat zodat de documentatie van de classes en functies in de code gegeneerd kan worden met het doxygen programma.

6. Planning

6.1 Globale planning

Om dit project goed uit te voeren, is het van belang op tijd bepaalde onderdelen af te hebben en op te leveren. Hiervoor moet dan ook een globale planning worden opgezet zodat alle teamleden goed weten waar de aan toe zijn. In overleg met alle teamleden en met oog op de eisen van de opdrachtgever is hier het volgende uitgekomen.

week	wat
1	globale planning + teamcontract
2	interview + moscow
3	X
4	plan van aanpak
5	requirements architectuur
6	solution architectuur
project1	testen
project2	eindproduct + verslag

In de tabel hierboven is per week te zien wat moet worden opgeleverd. In week 1 is dit de globale planning (deze planning) en het teamcontract. Het teamcontract bevat alle belangrijke informatie om de samenwerking tussen de teamleden in goede banen te laten lopen. Zo staan er de contactgegevens en de tijden waarop een ieder beschikbaar is. Ook staan er afspraken over de bijeenkomsten en vergaderingen, en afspraken hoe er wordt opgetreden als iemand zich niet aan de gemaakte afspraken kan houden.

In de tweede week zal een interview worden gehouden met de opdrachtgever. Aansluitend zal een MoSCoW worden opgesteld met daarin alle functionele eisen waaraan het eindproduct moet voldoen en zaken die eventueel worden meegenomen in het product.

De derde week hoeft niet iets te worden opgeleverd, deze week zal worden gebruikt om vast in voren te werken aan het plan van aanpak (dit document) en eventueel vast aan de requirement architectuur.

In week vier moet het plan van aanpak worden ingeleverd. Alvorens dit zal worden ingeleverd zal eerst feedback worden gevraagd aan een docent om te kijken of er niets mist.

Week vijf en zes worden gebruikt om de requirement en de solution architectuur af te maken en in te leveren. In week vijf zal feedback worden gevraagd aan een docent over de requirement architectuur. Deze zal worden verwerkt en de uiteindelijke architectuur zal als leidende draad functioneren voor de solution architectuur.

In de eerste projectweek zal aan de slag worden gegaan met de code en het product zelf. Hiervoor zullen ook testen moeten worden uitgevoerd. Het is de bedoeling zoveel mogelijk van deze testen in de eerste week te houden.

De tweede week van de projectweek is de laatste week van het project. Hierin zullen de laatste testen plaatsvinden en het uiteindelijke product, samen met het eindverslag worden opgeleverd.

6.2 Urenverdeling

Om iets meer inzicht te geven in de planning per week en om de gemaakte uren bij te houden, wordt een urenverantwoording opgesteld. Deze urenverantwoording is te vinden als bijlage bij dit document. In deze verantwoording is per persoon te zien hoeveel tijd er besteed is aan alle onderdelen op de planning.

7. Risico’s

7.1. Hardware

Een van de eerste risico’s die er is, zijn problemen met de hardware. Zo zijn er nog geen raspberry pies om op te werken beschikbaar gesteld, missen de bordjes die tussen de verbinding van/naar de pi en de microcontrollers en missen 2 trilmotors die de wastrommel emuleren. Wanneer die verkregen kunnen worden en van wie is nog niet duidelijk.

Andere problemen die zich met de hardware kunnen voordoen is dat de onderdelen slijten en/of kapot gaan. Zo is er bekent gemaakt dat vorig jaar microcontrollers beschadigt zijn geraakt door foutieve verbindingen waarbij ze te veel stroom aangeboden kregen. Hoewel dit probleem nu verholpen lijkt te zijn, blijkt dat er niet meer genoeg stroom beschikbaar is voor de trilmotors.

De bedoeling is dat de docenten benaderd worden warneer deze onderdelen wel beschikbaar zijn in het geval van ontbrekende onderdelen. Ook bij de gebrekige delen is de bedoeling daar de docenten te benaderen.

Een ander probleem dat zich kan voordoen is dat er onderdelen kapot gaan tijdens het werken. Met vier personen in het team zijn er 4 sets microcontroller en onderdelen beschikbaar die kapotte onderdelen kan vervangen. Mocht het onderdeel niet door de extra sets te vervangen zijn, dan kunnen deze - afhankelijk van het onderdeel - bijgekocht worden.

7.2. Software

Een andere groep risico’s bestaat uit het samenwerken en de ondersteunde mogelijkheden daartoe. Hoewel GitHub bedoelt is om met grote groepen samen te werken, is dit meer gericht op het offline samenwerken. Als twee personen hetzelfde bestand aanpassen, dan komen de veranderingen pas samen als het bestand wordt geupload. Google Drive (GDrive) daarintegen is gericht op het online samenwerken. Twee of meerdere personen kunnen tegelijkertijd een bestand aanpassen. Tegen de prijs dat bijde gebruikers online moeten zijn om het bestand te kunnen aanpassen. Software Ideas Modeler (SIM) kent een combinatie van beide problemen; de gebruiker moet offline zijn - hij kan er niet online bij - en er kan maar één persoon aangewerkt hebben. Als twee personen er aan gewerkt hebben, raken de veranderingen van één persoon verloren.

Voor GitHub bestaat een geheel protecol voor het uploaden (pushen) van veranderde/nieuwe content door middel van slim samenvoegen (mergen). Bij GDrive is de gebruiker afhankelijk van het internet, wil hij toch offline werken, dan treden de problemen bekent bij SIM in werking. De oplossing voor het SIM probleem is omslachtig maar nodig; één persoon wordt aangewezen om aan een SIM project te werken en bij de volgende bijeenkomst wordt er met één of meerdere teamleden naar het gemaakte project gekeken.

7.3. Overige

Er hoeft verder geen rekening gehouden te worden met licensies. Alle software die nodig is, is beschikbaar.

Mocht de klant niet meer beschikbaar zijn dan moet er naar een plaatsvervangende klant gezocht worden.
Mocht de klant onduidelijk zijn over bepaalde onderwerpen dan is het genootzaakt dat er met de klant nogmaals besproken wordt wat er nou precies bedoelt wordt.

8. Bronvermelding

Boeken:
Jansen, C., Mulder, J., van der Pool, E., Steehouder, M., Zeijl, W. (2012). Leren Communiceren. zesde druk. [Groningen]: Noordhof Uitgevers.

Sites:
http://www.greenem.nl/wasprogrammas-voor-een-wasmachine/#1440591206102-470bc24e-e50f
https://nl.wikipedia.org/wiki/Wasmachine

Overig:
Modulebeschrijven Themaopdracht 6 (TCTI-V2THO6)
Reader Realtime System Programming (TCTI-V2RTSP1)

Planning en registratie van de uren

	Themaopdracht 6	Team:	# 10								
		Kevin Damen [teamleider]		Jeroen Kok		Remco Nijkamp		Jordan Ramírez			
Weeknr	Omschrijving activiteit	geplande uren	bestede uren	geplande uren	bestede uren	geplande uren	bestede uren	geplande uren	bestede uren	Totaal geplande uren	Totaal bestede uren
1	Vergadering + Agenda + Notulen	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
1	Teamcontract opstellen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
1	Modulebeschrijving bestuderen	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	6.0	6.0
1	GitHub opzetten	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	2.0	2.0
1	Hoorcollege - Kick-off	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
1	Start Plan van Aanpak	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
2	opstellen vragen voor interview	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
2	pre-ontwerp GUI opstellen	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	6.0	6.0
2	Hoorcollege - interview	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
2	interview + voorbereiden	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
2	Vergadering + Agenda + Notulen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
2	interview nabespreken	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	2.0	2.0
2	MoSCoW opstellen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
3	Hoorcollege	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
3	MosCow aanpassen	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	2.0	4.0
3	Overleg	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
3	Requirements architectuur	2.0	1.5	2.0	1.5	2.0	1.5	2.0	1.5	8.0	6.0
3	Plan van aanpak	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	8.0
3	Vergadering + Agenda + Notulen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
4	Plan van aanpak afmaken	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	13.0	16.0
4	Vergadering + Agenda + Notulen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	4.0	4.0
4	Plan van aanpak feedback verwerken	2.0		2.0		2.0		2.0		8.0	0.0
4	Inlezen RTSP voor komende week + aanpak opzetten	1.0		1.0		1.0		1.0		4.0	0.0
5	Vergadering + Agenda + Notulen	1.0		1.0		1.0		1.0		4.0	0.0
5	Requirements architectuur	3.0		3.0		3.0		3.0		12.0	0.0
5	RA feedback verwerken	2.0		2.0		2.0		2.0		8.0	0.0
5	Solution Architectuur	3.0		3.0		3.0		3.0		12.0	0.0
6	Vergadering + Agenda + Notulen	1.0		1.0		1.0		1.0		4.0	0.0
6	Solution Architectuur	3.0		3.0		3.0		3.0		12.0	0.0
6	SA feedback verwerken	2.0		2.0		2.0		2.0		8.0	0.0
6	Planning maken projectweek	1.0		1.0		1.0		1.0		4.0	0.0
6	Wasmachine bouwen	1.5		1.5		1.5		1.5		6.0	0.0
P1	coderen	5.0		5.0		5.0		5.0		20.0	0.0
P1	testen	3.0		3.0		3.0		3.0		12.0	0.0
P1	testuitslagen verwerken in code	3.0		3.0		3.0		3.0		12.0	0.0
P2	testen	3.0		3.0		3.0		3.0		12.0	0.0
P2	laatste codering	3.0		3.0		3.0		3.0		12.0	0.0
P2	terugkoppeling	2.0		2.0		2.0		2.0		8.0	0.0
P2	eindverslag	4.0		4.0		4.0		4.0		16.0	0.0
P2	presentatie	1.0		1.0		1.0		1.0		4.0	0.0
P2	eindverslag feedback verwerken	2.0		2.0		2.0		2.0		8.0	0.0
	Totaal	77.0	30.5	77.0	30.5	77.0	30.5	77.0	30.5	308.0	122.0