



Scrum

Webleren

School je gratis bij via het internet. Waar en wanneer je wilt.

www.vdab.be/webleren

© COPYRIGHT 2013 VDAB

Niets uit deze syllabus mag worden verveelvoudigd, bewerkt, opgeslagen in een database en/of openbaar gemaakt door middel van druk, fotokopie, microfilm, geluidsband, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van VDAB.

Hoewel deze syllabus met zeer veel zorg is samengesteld, aanvaardt VDAB geen enkele aansprakelijkheid voor schade ontstaan door eventuele fouten en/of onvolkomenheden in deze syllabus en of bijhorende bestanden.

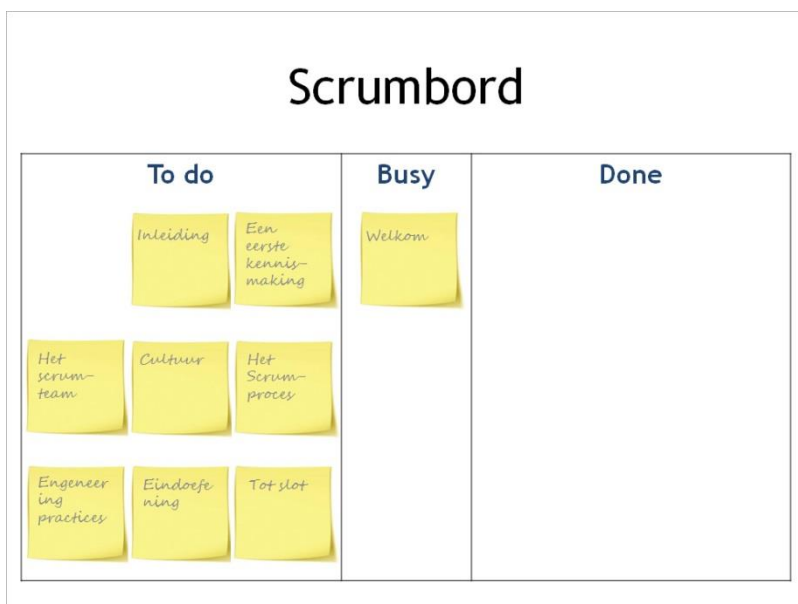
Inhoud

Hoofdstuk 1.	Inleiding	1
1.1.	Over deze cursus?	1
1.2.	Cegeka	2
Hoofdstuk 2.	Inleiding	3
2.1.	Het softwareontwikkelp proces	3
2.2.	De watervalmethode.....	4
2.2.1.	Inleiding	4
2.2.2.	Voordelen	4
2.2.3.	Nadelen	5
2.2.4.	Communicatieprobleem.....	5
2.3.	Wat is Agile.....	6
2.3.1.	Een andere benadering	6
2.3.2.	Ontstaan	6
2.3.3.	Agile, Scrum, eXtreme programming	7
2.4.	Agile vs waterval.....	8
2.5.	Agile manifesto.....	9
2.5.1.	Inleiding	9
2.5.2.	Wat?	9
2.5.3.	12 principes	9
2.6.	Opgaven.....	10
Hoofdstuk 3.	Een eerste kennismaking.....	11
3.1.	Inleiding	11
3.1.1.	De term Scrum.....	11
3.1.2.	De essentie van Scrum	12
3.2.	Het Scrumproces	12
3.2.1.	Schema	12
3.2.2.	Opgaven.....	13
Hoofdstuk 4.	Het scrumteam	14
4.1.	Inleiding	14
4.1.1.	Overzicht Teamrollen	14
4.1.2.	De andere belanghebbenden of Stakeholders.....	15
4.2.	Basisrollen	16

4.2.1.	Inleiding	16
4.2.2.	Ontwikkelaars	16
4.2.3.	Scrum master	17
4.2.4.	Product owner	18
4.3.	Bijkomende rollen	18
4.3.1.	Inleiding	18
4.3.2.	Customer proxy	19
4.3.3.	Coach	19
4.3.4.	Klant	20
4.3.5.	Operations	20
4.4.	Opgaven	21
4.4.1.	Opgave 1	21
4.4.2.	Opgave 2	21
Hoofdstuk 5.	Cultuur	22
5.1.	Intro	22
5.2.	11 Agile Values	23
5.2.1.	Commitment	23
5.2.2.	Focus	24
5.2.3.	Openness	24
5.2.4.	Respect	25
5.2.5.	Courage	26
5.2.6.	Communication	26
5.2.7.	Simplicity	27
5.2.8.	Feedback	27
5.2.9.	Collaboration	28
5.2.10.	Flexibility	29
5.2.11.	Continuous learning	29
5.2.12.	Tips	30
5.3.	Opgaven	30
Hoofdstuk 6.	Het scrumproces	31
6.1.	Inleiding	31
6.2.	Voor de eerste sprint	31
6.2.1.	Product backlog	31
6.2.2.	Enkele tips: Drie eenvoudige waarheden	33

6.2.3.	Scrum room	33
6.3.	Start van de sprint	34
6.3.1.	Inleiding	34
6.3.2.	Sprint planning meeting	34
6.3.3.	Planning poker	35
6.3.4.	Sprint backlog	35
6.3.5.	Hoeveel story's in één sprint?	36
6.4.	Tijdens de sprint	36
6.4.1.	Inleiding	36
6.4.2.	Het Scrumbord	37
6.4.3.	De Daily Scrum of Standup Meeting	38
6.4.4.	Burndown chart	38
6.4.5.	Inflexibiliteit	38
6.4.6.	DoD - Definition of Done	39
6.5.	Einde van de sprint	39
6.5.1.	Inleiding	39
6.5.2.	Sprint retrospective	39
6.5.3.	De Demo of Sprint Review meeting	40
6.6.	Opgaven	40
6.6.1.	Opgave 1	40
6.6.2.	Opgave 2: Scrum in aan andere context	40
Hoofdstuk 7.	Engineering practices	42
7.1.	Beproefde technieken	42
7.1.1.	Wat?	42
7.1.2.	Pair programming	42
7.1.3.	Test driven development	43
7.1.4.	Refactoring	43
7.1.5.	Collective ownership	44
Hoofdstuk 8.	Eindoefening	45
Hoofdstuk 9.	Einde cursus	46
9.1.	Tot slot	46
9.2.	Bronnen	47

Hoofdstuk 1. Inleiding



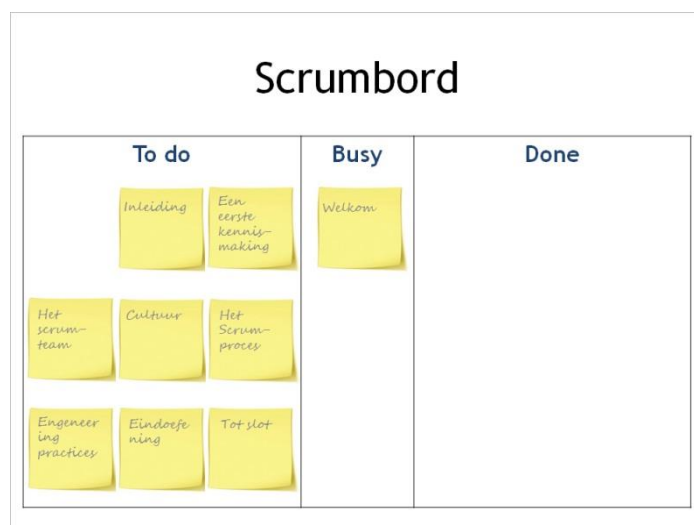
1.1. Over deze cursus?

In de cursus **Scrum** leggen we uit hoe de samenwerking in een Scrumomgeving verloopt. We bespreken waar de accenten liggen bij deze samenwerkingsmethode, hoe de onderlinge **communicatie** wordt georganiseerd en welke de verschillende **rollen** zijn die binnen een Scrumteam worden opgenomen.

Van een ontwikkelaar die deel uitmaakt van een Scrumteam, wordt ook een bepaalde houding verwacht. In het onderdeel **Cultuur** gaan we hier wat verder op in.

Als methode staat Scrum niet alleen en wordt vaak gecombineerd met andere **ontwikkelingstechnieken** zoals XP-practices. Ook deze bespreken we kort.

Doorheen de cursus zullen we een **Scrumbord** als leidraad gebruiken. Zo houd je een goed overzicht over de ganse cursus.



1.2. Cegeka

Deze cursus kwam tot stand dankzij een intensieve samenwerking tussen VDAB en de firma **Cegeka**.

Cegeka is een **full-service ICT-bedrijf**: je kan bij hen terecht voor advies, detachering van informatici, ontwikkeling en onderhoud van applicaties, bouw van websites, on-site en remote beheer van ICT-infrastructuur en outsourcing.

Met hun datacenters zijn ze klaar voor **cloud computing** en je digitale toekomst. 1700 experts voelen zich thuis in de complexe IT-wereld en zorgen samen met jou voor continue verbetering en innovatie.

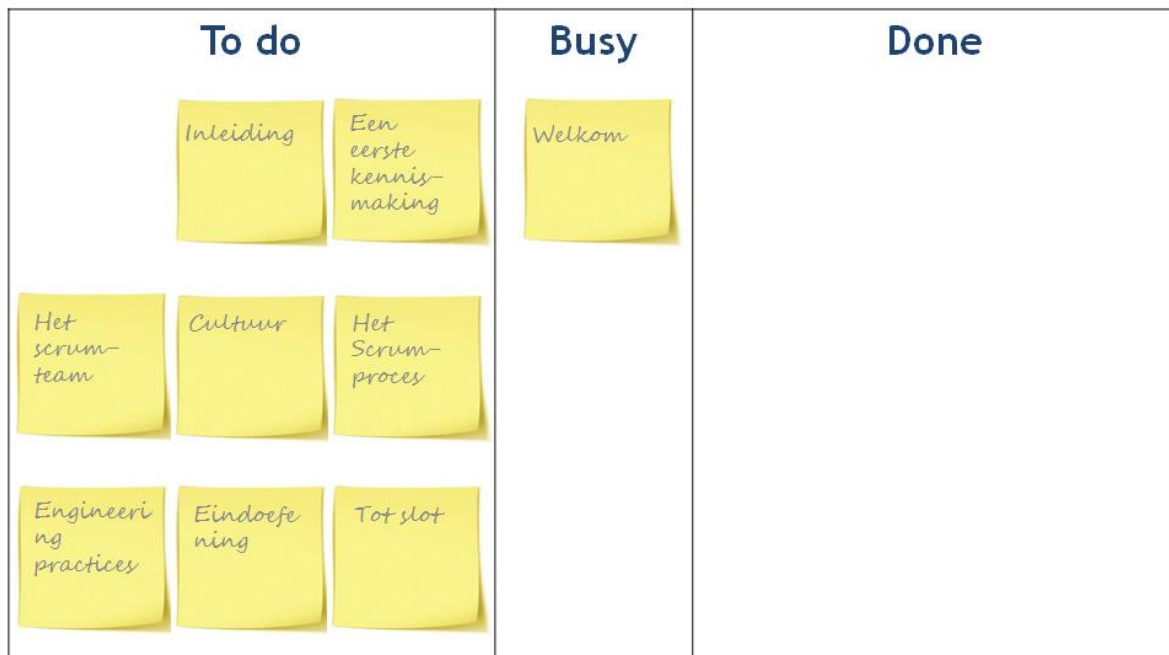
Meer informatie vind je op www.cegeka.be

De rechten op de foto's en afbeeldingen die opgenomen werden in de cursus zijn eigendom van Cegeka.



Hoofdstuk 2. Inleiding

Scrumbord



In dit hoofdstuk bekijken we

- wat de verschillende fasen in een softwareontwikkelingsproces zijn;
- hoe de watervalmethode verloopt;
- wat de voor- en nadelen van het watervalmodel zijn;
- wat Agile werken is;
- hoe Scrum binnen Agile past.

2.1. Het softwareontwikkelproces

Bij de ontwikkeling van nieuwe software programma's en applicaties komt heel wat kijken. Het is een **complex proces** waarbij meerdere medewerkers betrokken zijn. Het doorloopt verschillende fasen en vraagt om een projectmatige aanpak.

De fasen die in elk softwareontwikkelingsproces voorkomen zijn:

- **Analyse**
In samenspraak met de klant worden de vereisten van de applicatie gespecificeerd, nl. wat moet de applicatie doen en waarom. Dit wordt gedetailleerd op papier gezet.
- **Development**
De softwareapplicatie wordt ontwikkeld en getest.

- **Validatie**
In deze fase wordt nagegaan of de software voldoet aan de eisen en verwachtingen van de klant.
- **Onderhoud**
Na ingebruikname wordt het systeem verder opgevolgd en onderhouden. Indien nodig worden bepaalde functionaliteiten toegevoegd of aangepast.

2.2. De watervalmethode

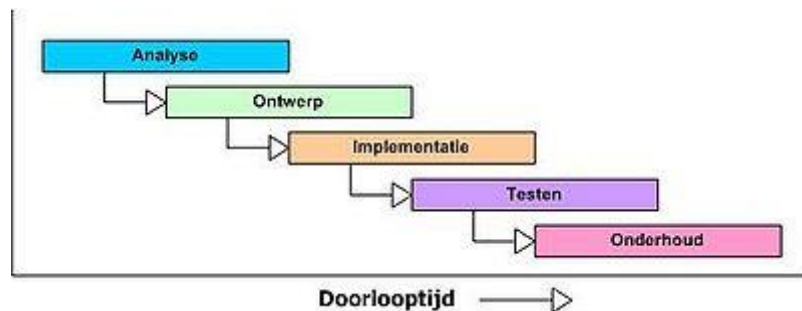
2.2.1. Inleiding

De traditionele manier om software te ontwikkelen volgt het **watervalmodel**.

Deze methode is gebaseerd op een zeer **planmatige** aanpak waarbij de ganse ontwikkelingsperiode in een aantal duidelijk **afgebakende fasen** wordt opgedeeld.

Deze fasen worden in **chronologische** volgorde doorlopen en afgewerkt. Men begint met fase 1 en werkt deze volledig af. Fase 2 kan pas starten als fase 1 afgesloten is.

In elke fase is het werk in handen van een team van **specialisten**. Deze teams werken los van elkaar hun deel van het project af.



2.2.2. Voordelen

De watervalmethode is nog steeds populair. Veel informaticabedrijven geven de voorkeur aan deze gestructureerde en planmatige aanpak.

De belangrijkste voordelen van deze manier van werken zijn:

- **Duidelijkheid**
De watervalmethode is een rechttoe-rechtaan methode. De manier van werken zorgt ervoor dat er zich concrete fasen vormen.
Op elk moment in het project is het voor iedereen duidelijk in welke fase men zit.
- **Mijlpalen**
Voorafgaand aan het project worden er bepaalde mijlpalen voorzien. Dit zijn belangrijke momenten of deadlines in het project die gebruikt worden om de voortgang van het werk in te schatten.
- **Bekendheid**
De watervalmethode is erg bekend. Veel mensen hebben er ervaring mee en kunnen er mee werken.

- **Documentatie**

Bij het watervalmodel gaat veel aandacht naar documentatie. Het werk dat in elke fase moet geleverd worden, wordt gedetailleerd beschreven. Ook worden de projectdoelstellingen en producteisen bij aanvang geanalyseerd en uitgebreid gedocumenteerd. Zo zijn alle vereisten van in het begin voor alle betrokkenen duidelijk.

2.2.3. Nadelen

In de loop van de jaren '90 blijken de methoden op basis van het watervalmodel **ontoereikend**. Ervaring leert dat er een aantal onoverkomelijke **nadelen** aan verbonden zijn.

In de eerste plaats blijkt het moeilijk om bij dergelijke langdurige projecten vooraf de benodigde **kosten en tijd** in te schatten.

Een tweede probleem is dat elke fase in het project wordt afgewerkt door een team van specialisten bv. de analisten, de ontwerpers, de bouwers,... Deze teams werken **los van elkaar**. Zij kunnen een andere kijk op het project hebben. Wanneer men in één van de fasen een fout ontdekt, die gemaakt werd in een eerdere fase, moet men helemaal terug om die te corrigeren.

Ook wordt er veel tijd verspild: specialisten die op elkaar moeten wachten, opstellen van documentatie die zelden wordt geraadpleegd,...

Vooraf het **onbuigzame karakter** van het watervalmodel is aanleiding van heel wat problemen. De strikt geplande volgorde waarin de verschillende fasen worden afgewerkt laat **weinig ruimte voor flexibiliteit**.

In de praktijk wijzigen de vereisten in de loop van een ontwikkelingsproject meer dan eens, bv. omdat de klant van gedacht is veranderd en toch iets anders wil.

Door de beperkingen van het watervalmodel hebben de ontwikkelaars weinig mogelijkheden om hierop in te spelen en hun werk te allen tijde bij te sturen.

In de jaren '90 dringt zich een nieuwe manier van werken op. Hier en der ontstaan er allerlei alternatieve methoden, de zogenaamde **lightweight methods of lichtgewicht methoden**, die uitgaan van een meer flexibele en organische aanpak. De Scrummethode is er hier één van.

2.2.4. Communicatieprobleem

Eén van de belangrijkste nadelen van de traditionele manier van werken met de watervalmethode is het **gebrek aan communicatie** tussen de betrokken teams.

Doordat de teams los van elkaar werken verloopt de onderlinge communicatie zelden rechtstreeks. Bepaalde teams hebben gedurende het ganse project zelfs helemaal geen contact met elkaar.

Vooraf het contact met de klant is onvoldoende. Daardoor heeft iedereen een ander idee van wat de klant precies gevraagd heeft. De klant krijgt enkel aan het begin van het project de kans om zijn verwachtingen en wensen in verband met de applicatie te formuleren. Enkel de analisten krijgen deze te horen. Deze requirements worden uitgebreid op papier gezet. Achteraf is het niet meer mogelijk om extra toelichting te vragen aan de klant. Ook kan de klant zijn requirements niet meer herzien of aanpassen. Hierdoor voldoet de applicatie vaak slechts gedeeltelijk of helemaal niet aan de verwachtingen van de klant.

In de praktijk is dit de belangrijkste reden waarom ontwikkelingsprojecten mislukken.



2.3. Wat is Agile

2.3.1. Een andere benadering

In het filmpje in de webcursus wordt het softwareontwikkelp proces op een heel andere manier bekeken.

2.3.2. Ontstaan

Het ontstaan van Agile als term en als concept is terug te voeren tot het **Agile Manifesto**. Dit manifest werd in februari 2001, tijdens een informele bijeenkomst van ontwikkelaars, opgesteld. Agile betekent letterlijk 'lenig' of 'soepel'.

In de praktijk bestaan er meerdere toepassingen van het Agile concept. Ze vertonen duidelijke verschillen, maar hebben een aantal kenmerken gemeenschappelijk:

Samenwerking en communicatie

Bij Agile development zijn **teamwerk** en **samenwerking** cruciaal.

Agile teams zijn in de regel klein. De teamleden zijn **multifunctioneel**. Ze werken in dezelfde ruimte en staan voortdurend in contact met elkaar.

Het team is **zelfsturend**. Het wordt niet gestuurd door een projectmanager. Ook is er geen overkoepelende teamleader die bepaalt wie welke taken zal uitvoeren en hoe problemen opgelost kunnen worden.

Binnen een Agile team is er **geen hiërarchie**. Elk teamlid is even belangrijk en heeft evenveel inspraak. De verantwoordelijkheid voor het eindresultaat is een gedeelde verantwoordelijkheid: ieder teamlid draagt bij tot en is even verantwoordelijk voor het resultaat van het geheel.

Niet alleen de onderlinge **communicatie**, maar ook regelmatige contacten in levende lijve met andere, externe betrokkenen zoals de klant, eindgebruikers en mensen van de business, enz. zijn ingebouwd.

Regelmatische releases van werkende software

Bij Agile is **werkende software** de eerste maatstaf van vooruitgang.

De verschillende fasen van het project worden niet langer chronologisch afgewerkt naar het einde toe, maar men geeft de voorkeur aan een iteratieve aanpak. Het ganse project wordt opgedeeld in steeds **terugkerende, korte ontwikkelperiodes** met een gemiddelde duur van 2 tot 4 weken.

Binnen deze periodes of sprints wordt er één essentieel, bruikbaar deel van de software geanalyseerd, ontworpen, ontwikkeld, getest én geïmplementeerd bij de klant. Stapsgewijs worden kleine onderdelen of functionaliteiten afgewerkt en afgeleverd bij de klant. Door de snelle en **continue levering** van bruikbare software wordt de tevredenheid en het vertrouwen van de klant gestimuleerd.

Flexibiliteit

De Agile methoden verwelkomen **verandering**.

Software ontwikkelen is een complex gegeven waarbij veranderingen zich voortdurend opdringen. Niet alleen kunnen de vereisten en prioriteiten van de klant veranderen, ook de omstandigheden en zelfs de inzichten van de ontwikkelaars zelf kunnen in de loop van het project wijzigen. Een flexibele en open houding vangt dit op.

Verandering en aanpassing zijn noodzakelijke voorwaarden voor verbetering.

Klantenbetrokkenheid

Agile ontwikkelaars dragen de **klantentevredenheid** hoog in het vaandel. Ze werken op maat van de klant en de eindgebruikers. De software wordt geheel volgens zijn behoeften en wensen ontworpen en ontwikkeld.

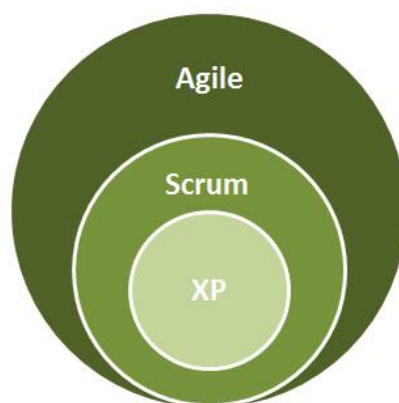
Concreet betekent dit dat de klant en eindgebruikers zoveel mogelijk bij het ontwikkelproces **betrokken** worden. Hun **feedback** is een noodzakelijke voorwaarde voor het slagen van het project. Bijgevolg worden bij aanvang duidelijke afspraken gemaakt omtrent de fysieke aanwezigheid van de klant op regelmatige en steeds terugkerende momenten in elke ontwikkelperiode.

2.3.3. Agile, Scrum, eXtreme programming

Binnen Agile bestaan er verschillende ontwikkelmethoden, zoals Scrum en eXtreme Programming.

Deze cursus zal zich vooral toespitsen op Scrum. Deze methode richt zich op de eigenlijke organisatie van het softwareontwikkelingsproces.

Een tweede methode die we kort zullen bekijken is eXtreme Programming (XP). XP richt zich vooral op hoe het programmeren zelf aangepakt zal worden.



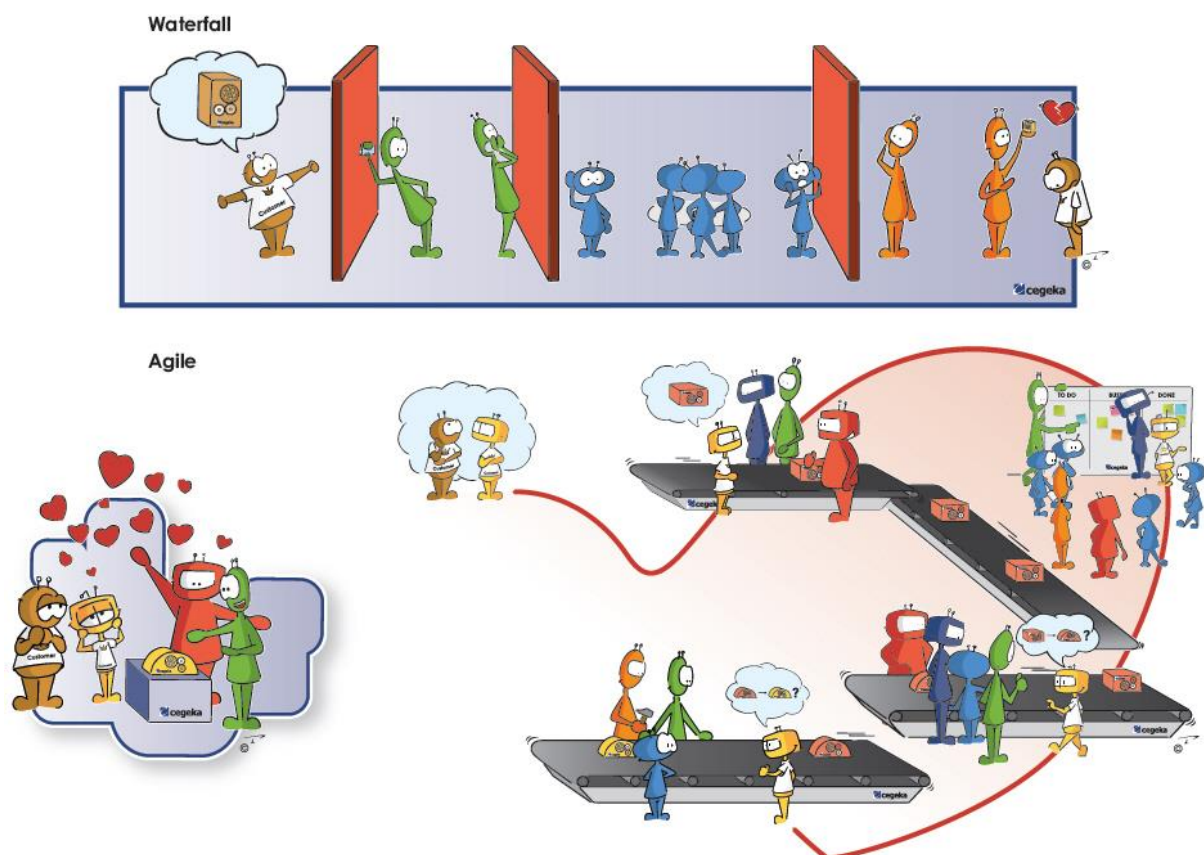
2.4. Agile vs waterval

In de animatie in de webcursus wordt Agile vergeleken met de watervalmethode.

Belangrijk om weten: de kleuren van de mannetjes komen overeen met hun rol in het team: klant, ontwikkelaar,

Het is ook niet zo dat de ene methode beter is dan de andere. Voor het ene project zal de traditionele manier van werken beter zijn, voor het andere project kan je beter Agile werken.

Een Agile project kan ook maar slagen als iedereen er achter staat.



In de webcursus vind je nog twee vergelijkingen tussen beide methoden.

2.5. Agile manifesto

2.5.1. Inleiding

Zoals gezegd organiseerden **17 software experts** in februari 2001 een informele bijeenkomst in Snowbird Utah. Deze vertegenwoordigers van lichtgewichtmethoden als eXtreme Programming, Scrum, Adaptive Software Development, Crystal, Feature-Driven Development en Pragmatic Programming kwamen bijeen om een degelijk en onderbouwd alternatief voor de gangbare zwaargewichtmethodologieën te bespreken.

Het resultaat van deze ontmoeting is het **Agile Manifesto**, een handvest dat de voornaamste principes van moderne software ontwikkeling bevat.

2.5.2. Wat?

Het manifest stelt dat bij goede software ontwikkeling...



De linkerkant is belangrijker dan de rechterkant. Dat wil niet zeggen dat de rechterkant helemaal niet meer nodig is.

Bv. Er wordt vooraf een plan gemaakt maar dit plan is flexibel aan te passen.

2.5.3. 12 principes

Agile volgt deze 12 principes:

1. Klanttevredenheid, door snelle, continue levering van bruikbare software is het doel.
2. Zelfs late veranderingen in de requirements zijn welkom.
3. Werkende software wordt regelmatig en op korte termijn geleverd (weken eerder dan maanden).
4. De ontwikkelaars werken nauw en dagelijks samen met de mensen die de business kennen.
5. Projecten steunen op gemotiveerde en betrouwbare personen.
6. Een gesprek in levende lijve is de beste manier van communicatie, wat betekent dat men zich best op dezelfde plek bevindt.

7. Werkende software is de eerste maatstaf van vooruitgang.
8. De ontwikkeling kan te allen tijde worden voortgezet.
9. Er is voortdurende aandacht voor technische uitmuntendheid en goed ontwerp.
10. Eenvoud is belangrijk: hoe meer er niet gedaan wordt, hoe beter.
11. De teams organiseren zichzelf.
12. Men past zich aan aan de omstandigheden.

Om deze ideeën te kunnen toepassen op de werkvloer, moeten ze worden omgezet in concrete afspraken en regels. Een concrete vertaling van de abstracte waarden van de Agile Cultuur is terug te vinden bij samenwerkingsmethoden als Scrum, Lean en Kanban.


Het originele manifesto kan je nalezen op <http://agilemanifesto.org/principles.html>. De volledige Nederlandstalige versie vind je op <http://agilemanifesto.org/iso/nl/principles.html>.


2.6. Opgaven


Sleep elk item naar de juiste plaats in het schema.


Begrijpbare documentatie	Reageren op verandering	Een plan volgen
Werkende software	Individuen en interacties	Contract onderhandeling
Procedures en tools		Samenwerking met de klant


is belangrijker dan














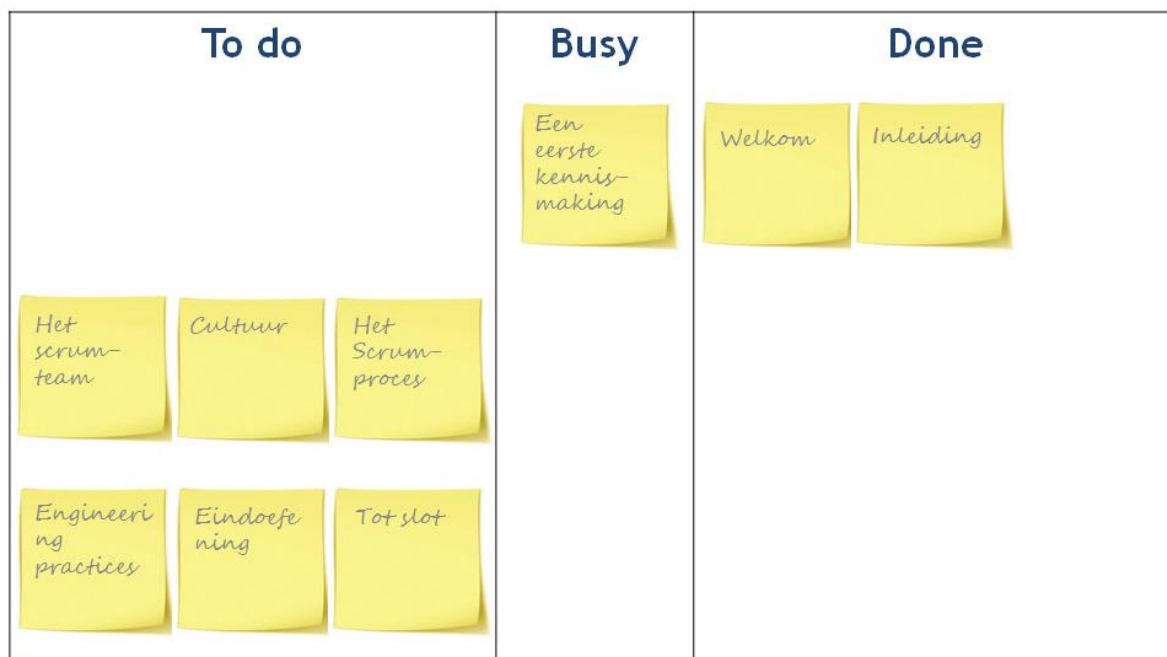






Hoofdstuk 3. Een eerste kennismaking

Scrumbord



3.1. Inleiding

3.1.1. De term Scrum

Scrum is een vernieuwende samenwerkingsmethode voor de ontwikkeling van softwareapplicaties. Het is een concrete toepassing van de agile methodologie.

De term 'Scrum' komt uit de **rugby** sport. Het is de spelfase waarbij de leden van een ploeg schouder aan schouder 1 front vormen tegen de tegenpartij. De individuele teamleden vormen één kluwen en zijn zo niet meer afzonderlijk herkenbaar. Als één groep werken ze aan hetzelfde doel, nl. de tegenpartij wegduwen om de bal opnieuw in eigen kamp te krijgen.



3.1.2. De essentie van Scrum

De kern van Scrum is een **multidisciplinair en zelfsturend team**. Samen pakt het team het project op. Iedereen is betrokken bij het plannen van het werk, het verdelen van de taken en het signaleren van de blokkades. Daarbij gaat Scrum ervan uit dat de benodigde kennis in het team zelf aanwezig is.

Bij Scrum wordt gewerkt in korte iteraties of **sprints** van twee tot maximaal vier weken. Samen met de klant wordt er bepaald waaraan er tijdens de sprint gewerkt zal worden. Het resultaat van iedere sprint is een werkend onderdeel of klein stukje functionaliteit van de softwareapplicatie die ontwikkeld wordt.

Aan het begin van iedere werkdag wordt er een **Daily Scrum** of **Daily Standup** gehouden. Zo weet iedereen wie wat aan het doen is en wat de eventuele problemen zijn.

Elke sprint wordt door het team zowel met elkaar als met de klant **geëvalueerd**. Dit nodigt uit tot het geven van feedback en daarmee tot openheid binnen het team.

Scrum is geen methode die je zonder meer kan toepassen. Het is eerder een raamwerk dat je aan je eigen noden kan/moet aanpassen.

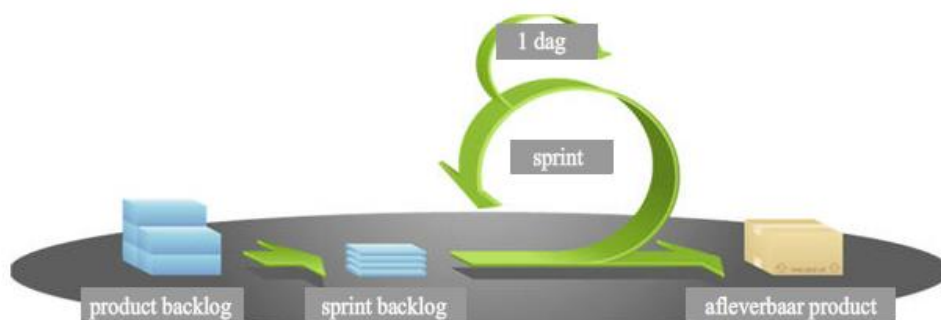
3.2. Het Scrumproces

3.2.1. Schema

Het wordt tijd om een eerste keer het volledige scrum-proces te bekijken. Belangrijk om weten: we willen voorlopig enkel in grote lijnen tonen hoe de sprint verloopt. Verder in de cursus gaan we dieper in op de verschillende fasen van de sprint.

Zoals gezegd wordt er binnen Scrum gewerkt met sprints, korte ontwikkelperiodes die volgens een vast patroon verlopen.

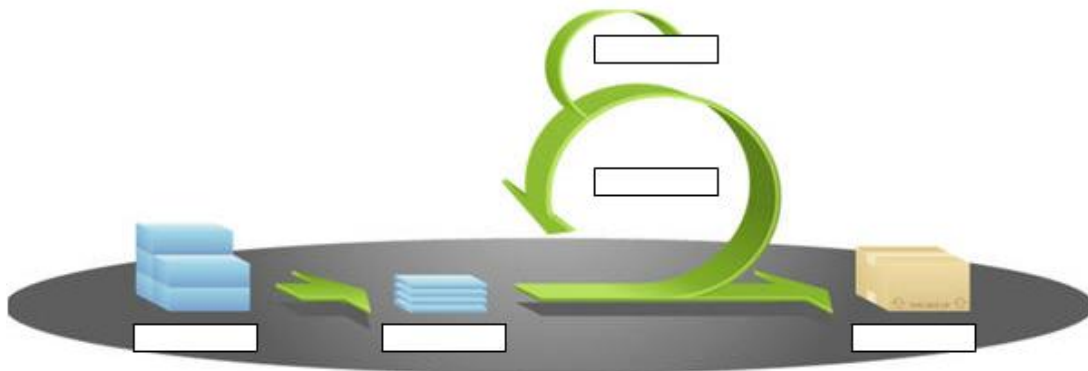
In de animatie in de webcursus krijg je meer uitleg over elk van de stappen door met je muis over de vakjes te bewegen.



3.2.2. Opgaven

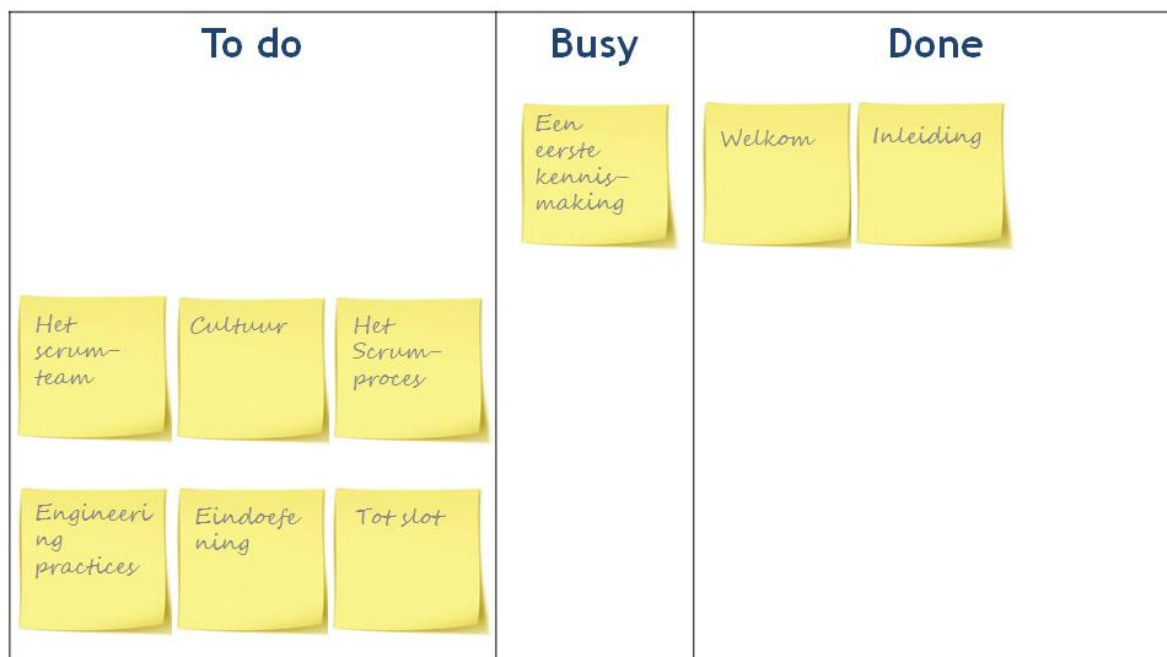
Sleep elk item naar de juiste plaats op het schema

Afleverbaar product	Sprint backlog	1 dag
Sprint	Product backlog	



Hoofdstuk 4. Het scrumteam

Scrumbord



4.1. Inleiding

Bij Scrum is **teamwerk** en **samenwerking** cruciaal.

Een Scrumteam is bij voorkeur **klein** en bestaat uit 7 personen plus of min 2.

De teamleden hebben elk hun eigen rol. Ze werken in dezelfde ruimte en staan voortdurend in **contact** met elkaar. Dit betekent een belangrijke tijdswinst en maakt gedetailleerde documentatie overbodig.

Het team is **zelfsturend**. Het managet zichzelf. Het wordt niet geleid of van bovenaf gestuurd door een projectmanager. Er is geen overkoepelende teamleader die bepaalt wie welke taken zal uitvoeren en hoe problemen opgelost kunnen worden.

Binnen een Scrumteam bestaat er **geen hiërarchie**. Elk teamlid is even belangrijk en heeft evenveel inspraak.

De medewerkers zijn multifunctioneel. De verantwoordelijkheid voor het eindresultaat is een **gedeelde verantwoordelijkheid**: ieder teamlid draagt bij tot en is even verantwoordelijk voor het resultaat van het geheel.

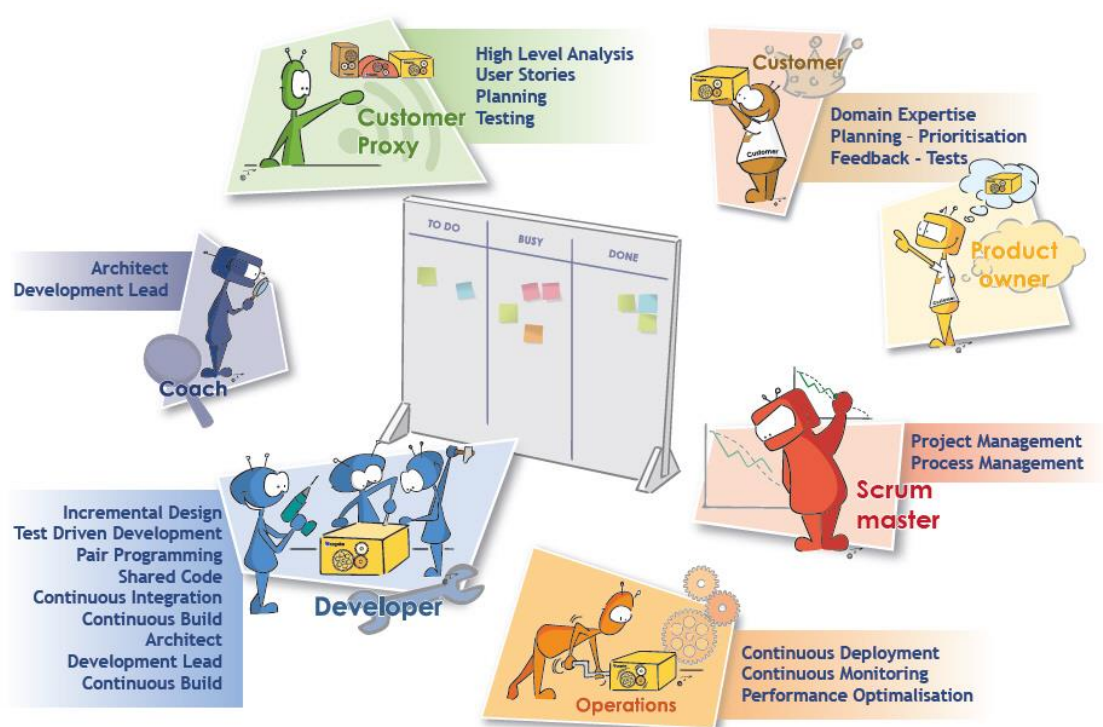
4.1.1. Overzicht Teamrollen

Onderstaande afbeelding geeft aan welke rollen er allemaal vertegenwoordigd kunnen zijn binnen

een Scrumteam. Niet elk Scrumteam zal alle rollen opnemen. In het vervolg van de cursus zullen we de verschillende rollen uitgebreid bespreken.

Belangrijk om weten: de gebruikte kleuren horen bij de rol.

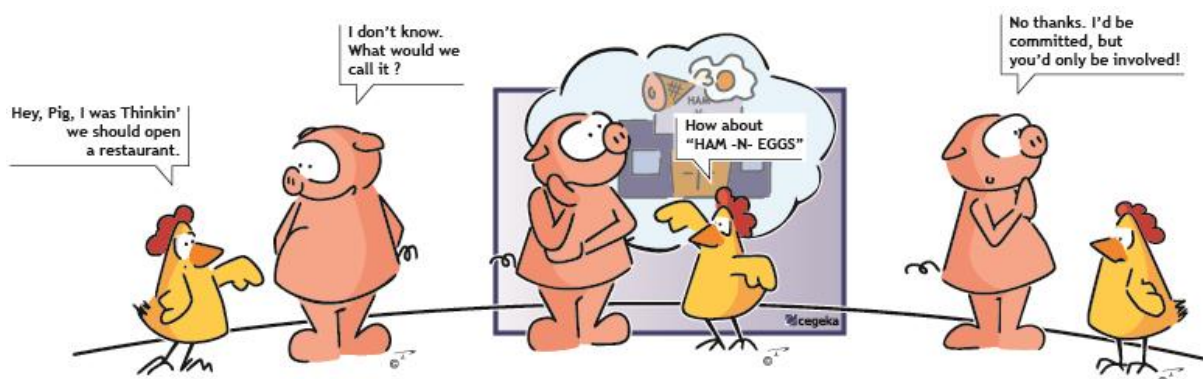
In het vervolg van de cursus zal een rood mannetje altijd de Scrum master zijn bijvoorbeeld.



4.1.2. De andere belanghebbenden of Stakeholders

Buiten het Scrumteam zijn er nog mensen **betrokken** bij het project. Dit kunnen eindgebruikers, managers of medewerkers van de business afdeling zijn. Zij worden **stakeholders** genoemd en hebben een belangrijke invloed op het werk van het scrumteam.

De ontwikkelaars moeten in hun werk voortdurend rekening houden met de noden, eisen en wensen van de stakeholders. De ontwikkelaars zijn verantwoordelijk voor het eindresultaat. Dit veronderstelt dus een grote mate van **commitment** of toewijding aan het project.



4.2. Basisrollen

4.2.1. Inleiding

Minimaal bestaat een Scrumteam uit:

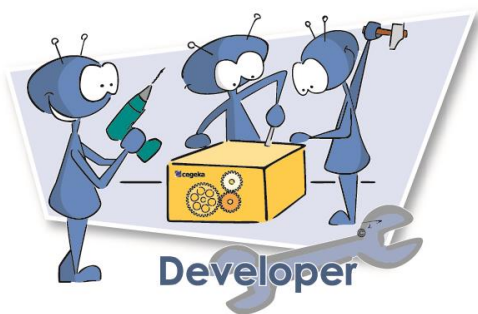
- de **ontwikkelaars** zelf,
- de **Scrum master**,
- de **product owner**.

De ontwikkelaars hebben als hoofdtak het ontwikkelen van de gevraagde software applicatie.

Zij worden hierbij **ondersteund** door de Scrum master en de product owner. Net zoals de ontwikkelaars zijn Scrum master en product owner volwaardige leden van het team.

Aangezien er geen sprake is van hiërarchie binnen een Scrumteam, begeleiden zij de ontwikkelaars **van binnenuit** en niet van bovenaf.

4.2.2. Ontwikkelaars



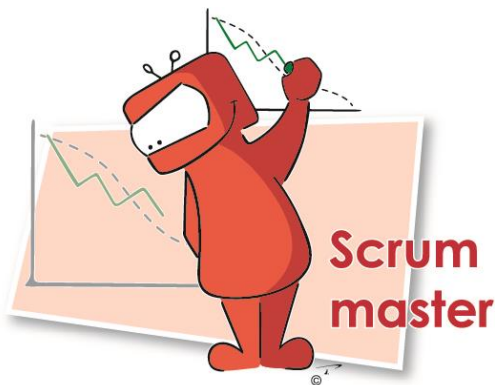
De ONTWIKKELAARS zetten de **requirements** of vereisten van de klant en de eindgebruiker om in werkende software. Zij houden zich hoofdzakelijk bezig met het **schrijven van code** en van geautomatiseerde tests voor deze code. Daarnaast werken zij het technisch ontwerp en de architectuur van de oplossing uit.

Natuurlijk **testen** zij ook of de geïmplementeerde oplossing werkt en dit zowel vanuit een functioneel als een niet-functioneel standpunt. Dit wil zeggen dat zij niet enkel nagaan of de software technisch gezien functioneert, maar ook of deze voldoende gebruiksvriendelijk is voor de eindgebruiker.

De ontwikkelaars zijn in principe **multifunctioneel**. Dit betekent dat er geen onderscheid gemaakt wordt tussen programmeurs, ontwerpers of architecten. Programmeurs zijn tegelijk ook ontwerpers en architecten, en vice versa.

In de webcursus kan je nog een filmpje bekijken van een ontwikkelaar.

4.2.3. Scrum master



De SCRUM MASTER is de bewaker van het project. Hij zorgt ervoor dat de teamleden zich kunnen bezighouden met het ontwikkelen van de software in de best mogelijke omstandigheden.

De Scrum master ontfermt zich in de eerste plaats over de **impediments**. Impediments zijn alle mogelijke problemen of blokkades die de medewerkers hinderen bij het uitvoeren van hun werk en een negatieve invloed hebben op de productiviteit van het team. Het is de verantwoordelijkheid van de teamleden om deze impediments te rapporteren aan de Scrum master. De Scrum master zoekt voor hen een oplossing. Impediments zijn zeer uiteenlopend en kunnen variëren van een kapot keyboard, een te warme werkruimte, een onduidelijke communicatie tot een teamlid dat onvoldoende meewerkt.

Door de Scrum master **regelmatig** in te lichten over dergelijke hindernissen, krijgt hij de kans om het team door en door te leren kennen. Op die manier is hij in staat in te schatten wat het team nodig heeft om optimaal te kunnen functioneren.

De Scrum master helpt het team voortdurend om zichzelf Agile te **organiseren**. Hij staat in voor de organisatie van de dagelijkse **stand up meetings** en volgt de **voortgang** van het project op. Ook plant hij de meetings met de klanten, de mensen van de business en andere interne en externe betrokkenen van het project.

Ondanks zijn taak, blijft de Scrum master gewoon deel van het team. Hij staat er niet boven.

In de webcursus kan je nog een filmpje bekijken van een Scrum master.

4.2.4. Product owner



De PRODUCT OWNER is het eerste en belangrijkste **aanspreekpunt** voor de klant, maar niet noodzakelijk het enige.

Voor elke sprint regelt de product owner de **scope** van het project. Dit doet hij in samenspraak met de klant. De scope is de omvang van het werk dat afgeleverd dient te worden binnen de afgesproken tijdspanne. Samen met de klant bepaalt de product owner tegelijk de **prioriteiten**. Bij deze besprekingen dienen zij sterk rekening te houden met het voorziene **budget**.

Vervolgens verduidelijkt de product owner de **requirements** of vereisten van de klant voor de ontwikkelaars.

In de loop van de sprint valideert de product owner de resultaten van het werk van de ontwikkelaars. Hij verifieert hierbij of hun werk voldoet aan de verwachtingen van de klant.

De product owner werkt nauw samen met het ganse project team. Hij neemt de eindbeslissingen in geval van onenigheid of conflicten met externe en interne belanghebbenden of 'stakeholders', bv. de medewerkers van de business.

In de webcursus kan je nog een filmpje bekijken van een product owner.

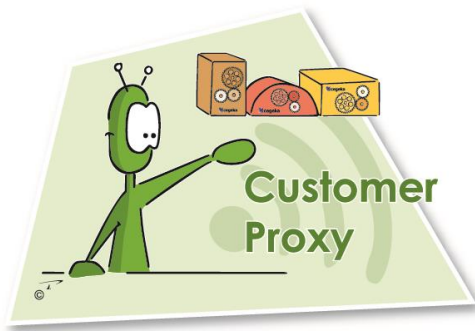
4.3. Bijkomende rollen

4.3.1. Inleiding

Elk bedrijf is vrij om andere rollen aan een Scrumteam toe te voegen. Zo kan het gebeuren dat in geval van een complex project de teamleden worden versterkt met één of meerdere **customer proxies** en een **coach**.

Een laatste, maar daarom niet minder belangrijke rol, is die van de **klant** en de **eindgebruiker**. In een Scrumproject wordt de klant zoveel mogelijk betrokken bij het ontwikkelproces. Het eindproduct wordt ontwikkeld op maat van de klant/eindgebruiker. Om dit te kunnen realiseren moet het voor de ontwikkelaars duidelijk zijn wat de klant precies verwacht en wenst. Hier heeft de klant een zware taak: hij heeft immers de verantwoordelijkheid om aan het Scrumteam duidelijk te maken wat hij precies wil.

4.3.2. Customer proxy



Een Scrumteam kan versterkt worden met een of meerdere CUSTOMER PROXIES. Zij nemen de rol op van de klant en/of de uiteindelijke gebruikers van de applicatie binnen het team ontwikkelaars.

Een customer proxy bewaakt de kwaliteit van het product vanuit het standpunt van de klant/eindgebruiker. Het is zijn taak om de wensen, eisen en behoeften van klant/eindgebruiker te analyseren en uit te schrijven: ze zetten de vereisten of **requirements** om in **user stories**.

Een **user story** bestaat uit gedetailleerde en technisch geformuleerde informatie over één kleine essentiële **functionaliteit** of werkend stukje van de software. Op basis van de informatie in de user story zullen de ontwikkelaars dit kleine stukje functionaliteit uitwerken.

De customer proxy formuleert ook de **acceptatiecriteria** of de criteria waaraan een functionaliteit moet voldoen vooraleer ze afgeleverd kan worden.

4.3.3. Coach



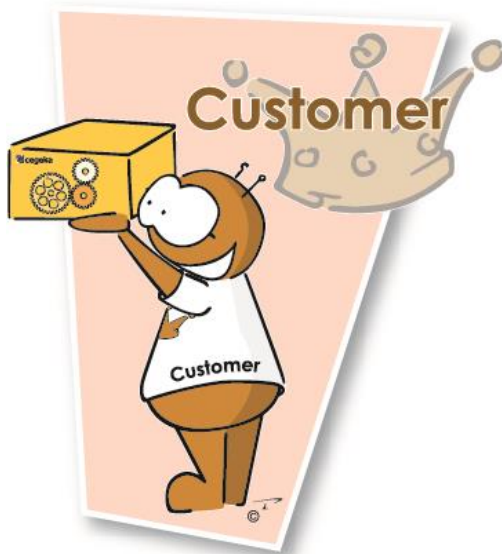
Soms beschikt een Scrumteam over een COACH of technisch begeleider. Hij **begeleidt** de ontwikkelaars bij het gebruik van de overeengekomen engineering practices, design en architectuur.

Hij zorgt ervoor dat elk **technisch probleem** zo snel mogelijk wordt opgelost.

Net zoals de andere ontwikkelaars houdt hij zich ook bezig met het schrijven van code en geautomatiseerde **tests** voor deze code.

Ook het **coachen** van nieuwe teamleden kan tot zijn taken behoren.

4.3.4. Klant



De Scrumaanpak beschouwt de **KLANT** als een **volwaardig lid** van het team. Hij speelt een belangrijke rol bij het realiseren van de doelstellingen van een scrumproject. In elke fase van het project is **direct contact** met de klant essentieel.

In ideale omstandigheden is de klant aanwezig op dezelfde locatie als het ontwikkelteam en verlopen de gesprekken **face-to-face**. In de praktijk is dit echter niet altijd mogelijk. Daarom wordt vooraf contractueel vastgelegd bij welke besprekingen de klant aanwezig moet zijn. Ook andere verantwoordelijkheden van de klant kunnen worden opgenomen in het contract.

Het is de verantwoordelijkheid van de klant om via de proxies of product owner aan het team duidelijk te maken wat zijn wensen en verwachtingen zijn. Concreet betekent dit dat hij de proxies helpt bij het opstellen van de **user stories**. Ook werkt hij mee aan het **uittesten** en valideren van de nieuwe functionaliteiten.

4.3.5. Operations



Een Agile projectaanpak heeft ook zijn weerslag op de **operationele activiteiten** die nodig zijn om de software voor een bepaalde omgeving te configureren, installeren, monitoren, etc.

De **samenwerking** tussen ontwikkelaars en operations is cruciaal is voor het welslagen van elk agile project want geen enkele klant heeft iets aan fantastische software als die niet op een goede manier in productie kan functioneren. Vroeger gebeurde dit vaak niet en kregen de mensen van operations helemaal op het einde van een project iets wat ze zo goed en zo kwaad als het kon in één of andere omgeving moesten uitrollen.

Centraal in die samenwerking staat een doorgedreven **automatisering**, bv. serverconfiguraties, software-installaties, sanitychecks, etc.).

Daarbij hoort bovendien een volledig versiebeheer van alle code, scripts en configuratiegegevens zodat het opzetten van een volledig werkende omgeving een snel, foutloos en reproduceerbaar proces wordt.

4.4. Opgaven

4.4.1. Opgave 1

Wie doet wat?

Link elke taak aan de juiste rol door de taak naar de juiste rol te slepen of de letter van de rol in te typen.

Taken

- ☐ Code testen
- ☐ Het proces bewaken
- ☐ Impediments oplossen
- ☐ Prioriteiten stellen
- ☐ Ontwikkelaars begeleiden
- ☐ Technische problemen oplossen
- ☐ Contact met de klant onderhouden
- ☐ Code schrijven

Rollen

- A) Ontwikkelaar
- B) Scrummaster
- C) Product owner
- D) Coach

Clear

Submit

4.4.2. Opgave 2

Scrum in een andere context

Bij een bouwproject zijn altijd een aantal personen betrokken:

- architect,
- aannemer,
- bouwvakker,
- veiligheidscoördinator
- en natuurlijk de bouwheer.

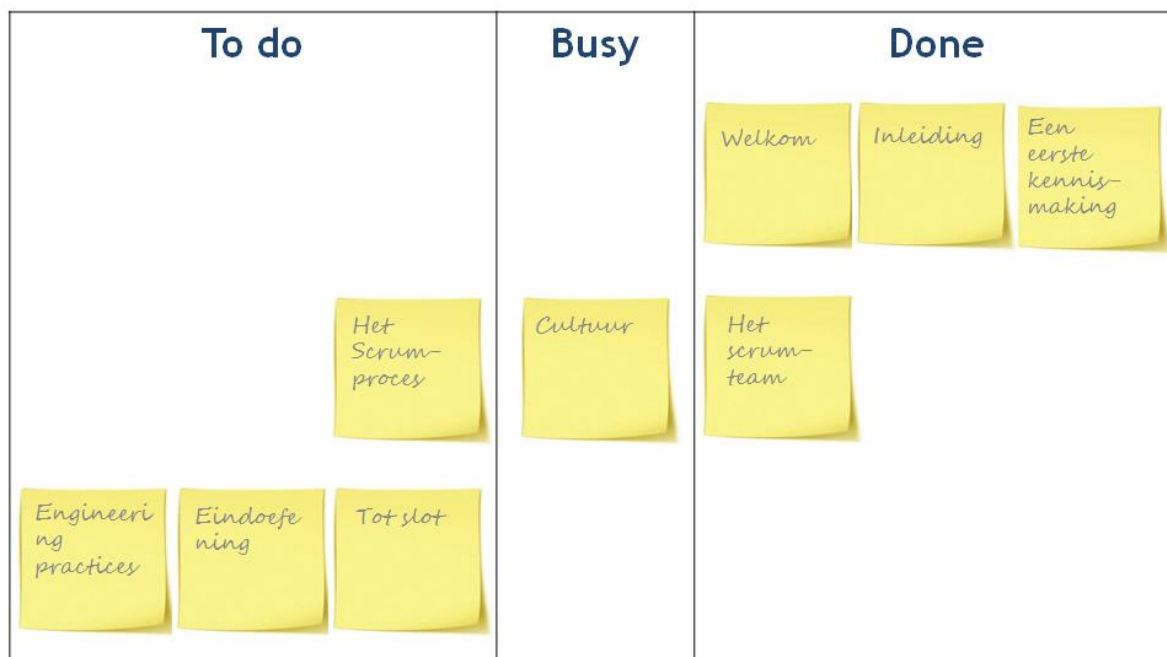
Wie zou welke rol op zich nemen als ze als Scrumteam te werk zouden gaan?

Scrummaster	↔	-- Maak uw keuze --	▼
Klant	↔	-- Maak uw keuze --	▼
Product Owner	↔	-- Maak uw keuze --	▼
Developer	↔	-- Maak uw keuze --	▼
Coach	↔	-- Maak uw keuze --	▼

Antwoorden versturen

Hoofdstuk 5. Cultuur

Scrumbord



5.1. Intro

Zoals gezegd is Agile een **methodologie**, een theoretisch concept. Het geeft een filosofische kijk op hoe verschillende medewerkers in een bedrijf op een open en creatieve manier samenwerken om een complex software ontwikkelingsproject te realiseren en de klantentevredenheid met hun product te verhogen.

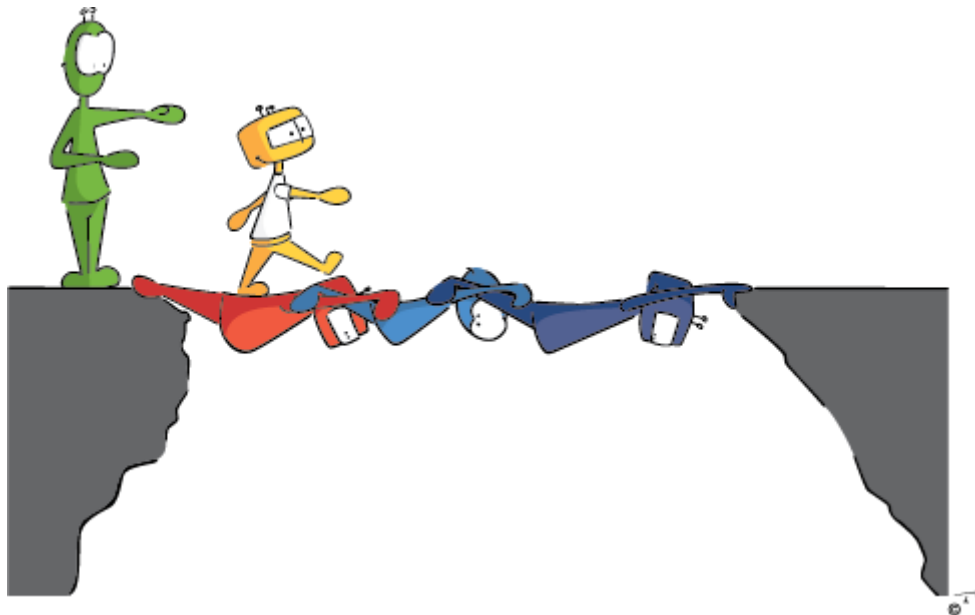
Men spreekt ook over Agile als **cultuur**, een ideologie die gestoeld is op een reeks waarden en normen die moeten voorkomen dat software ontwikkelaars verdwalen in de complexiteit van een ontwikkelingsproject.

In wat volgt zullen we 11 belangrijk values of waarden bespreken. De volgorde waarin ze besproken worden is willekeurig, de ene value is niet belangrijker dan de andere.

5.2. 11 Agile Values



5.2.1. Commitment



- Doe wat je zegt dat je zal doen.
- Als je problemen ondervindt bij het afwerken van je taken, bespreek dit zo snel mogelijk met je teamleden.
- Zoek samen met de anderen een oplossing zodat je je werk toch gedaan krijgt.
- Wees betrokken.
- Voor het slagen van het project is het belangrijk dat elk teamlid zich even betrokken en verantwoordelijk voelt voor het eindresultaat.

DO NOT

- Do not sit back and wait for orders!

5.2.2. Focus



- Beperk het aantal taken waar je op een bepaald moment aan bezig bent.
- Als je je op één taak concentreert, zal de kwaliteit van het resultaat beter zijn.
- Tijdens de sprint planning meeting spreken de teamleden af welke taken ze tijdens de volgende sprint zullen afwerken. Het team focust zich op wat er tijdens de sprint planning afgesproken is.
- Kaart problemen zo snel mogelijk aan en los ze op op het moment dat ze zich voordoen.
- Zo vermijd je dat onafgewerkte taken zich opstapelen.

5.2.3. Openness



- Creëer een open sfeer. Zorg voor een omgeving waarin iedereen zijn mening zonder

terughoudendheid kan en wil uiten.

Enkel als problemen openlijk besproken worden en vragen gesteld worden, kunnen deze ook effectief aangepakt worden.

Alle teamleden moeten openlijk kunnen spreken over:

- problemen bv. met het werk, de werkwijze, collega's, ...,
 - prioriteiten,
 - waarmee ze bezig zijn,
 - wat ze wel/niet gedaan hebben of niet afgewerkt krijgen,
 - het feit dat ze het doel van de sprint misschien niet zullen halen.
 - Stimuleer creatief denken of out-of-the-box-thinking.
 - Communiceer open met je klant, ook over mogelijke problemen of vertragingen.
- Eerlijk duurt het langst!

5.2.4. Respect



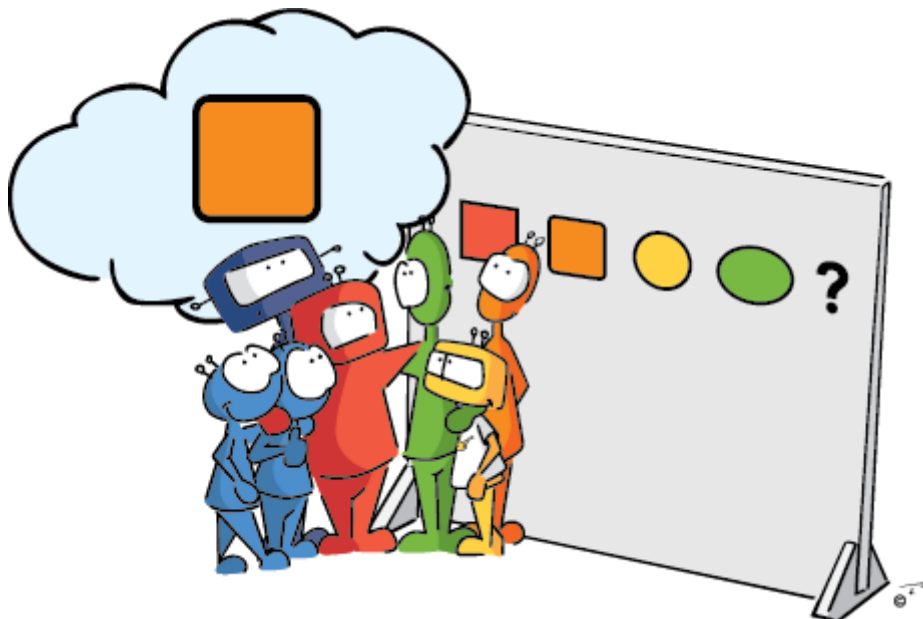
- Toon respect voor elkaar. Heb vertrouwen in elkaars inzet en betrokkenheid. Ongeacht wat het team achteraf vaststelt, begrijpen en geloven de teamleden oprecht dat alle betrokkenen hun uiterste best gedaan hebben, elk met zijn vaardigheden en capaciteiten, de beschikbare middelen, en de situatie van het moment.
- Toon respect voor elkaar door:
 - anderen toe te laten zich op hun manier toe te leggen op hun taken en hen niet te vertellen wat en hoe ze hun werk moeten doen;
 - in geval van een probleem echt te luisteren naar elkaars standpunt en alle betrokkenen de kans te geven deze te uiten;
 - niet te verwachten dat de teamleden overuren doen zodat iedereen een goede balans kan vinden tussen werk en gezin;
 - bij een evaluatie van het afgeleverde werk en de werking zich te concentreren op constructieve feedback.

5.2.5. Courage



- Moed heb je nodig om taken op te nemen die je uitdagen en die je uit je comfortzone halen.
- Heb de moed om iets te zeggen als
 - niet alles volgens plan verloopt.
 - het doel van de sprint niet bereikt zal worden.
 - een impediment serieuze gevolgen zal hebben voor de rest van het project.

5.2.6. Communication



- Communiceer als team dagelijks en openlijk over waarmee je bezig bent en welke problemen je ondervindt.
- Kies voor directe vormen van communicatie, bv. face-to-face gesprekken en meetings.
- Bespreek dagelijks met elkaar de vooruitgang van het werk.
Maak het werktempo zichtbaar voor alle betrokkenen.
- Maak gebruik van hulpmiddelen om verschillende aspecten van het werk en het werktempo te visualiseren, bv. een whiteboard tijdens ontwerpssessies.
- Ga niet uit van veronderstellingen, maar stel vragen zodra je twijfelt over iets of hulp nodig hebt.

DO NOT

- Do not be stubborn, do not assume, just ask!

5.2.7. Simplicity



- Vereenvoudig de taken door ze te analyseren en op te splitsen in deeltaken of zo klein mogelijke functionaliteiten.
- Begin met het oplossen van het probleem door het eenvoudigste ding dat zou kunnen werken te maken.

5.2.8. Feedback



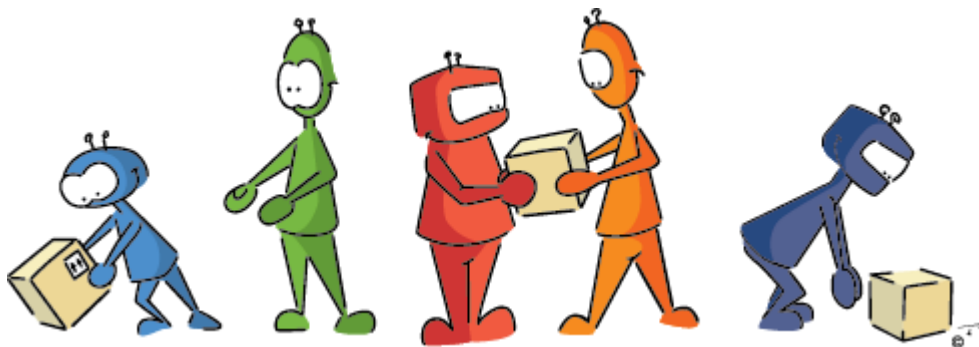
- Bespreek regelmatig de resultaten van elkaars werk en elkaars manier van werken. Geef constructieve feedback. Leg uit hoe de ander zichzelf kan verbeteren.

- Focus je bij de bespreking van een probleem op de mogelijke oplossingen. Werk naar een oplossing toe waar het hele team achter staat.
- Stel geen bekritiserende vragen, maar gebruik open en oplossingsgerichte vragen die het probleem verder uitklaren en leiden naar een oplossing.
- Las technieken in die feedback stimuleren bv. pair programming. Door met twee te werken aan eenzelfde stukje functionaliteit, krijg je direct feedback van elkaar. Anders moet je wachten op een code review.
- Organiseer je project in korte iteraties. Bespreek op het einde van elke iteratie met alle betrokkenen het resultaat van het geleverde werk. Op die manier wordt continue verbetering mogelijk gemaakt.
- Wijs de klant op het belang van zijn rol in het project. Organiseer regelmatig meetings met de klant en vraag hem om feedback over het afgeleverde werk. Worden zijn verwachtingen ingelost?

DO NOT

- Vermijd het gebruik van negatieve formuleringen bij het geven van feedback. Positieve feedback is motiverender dan negatieve. Een welgemeend compliment bevordert de sfeer onder de teamleden en werkt stimulerend.
- Sta open voor zowel positieve als negatieve feedback van je collega's. Beschouw ze als een kans om jezelf te verbeteren en bij te leren. Word in geval van een discussie niet emotioneel. Focus je op de feiten en op hoe je jezelf en je werk kan verbeteren.
- Geef elkaar niet de schuld van eventuele 'fouten'.

5.2.9. Collaboration



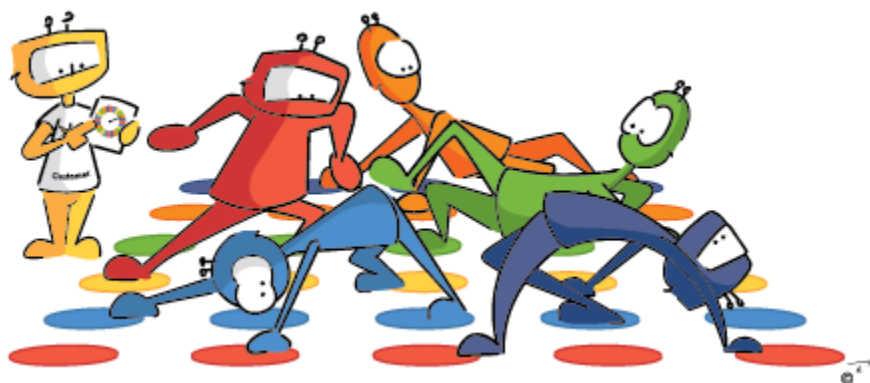
- Schrijf het resultaat van het werk toe aan het ganse team. Dit geldt ook als het resultaat niet aan de verwachtingen voldoet.
Zodra een teamlid een stukje functionaliteit ontwikkeld heeft of een stukje code geschreven heeft, is het resultaat eigendom van het ganse team. Bijgevolg heeft het geen zin om met de vinger te wijzen en te zeggen 'Die jongens schreven die lelijke code' als het over teamleden gaat. Het doet er immers niet toe welke personen verantwoordelijk zijn voor lelijke code. Alle code, zowel de 'propere' als de 'lelijke', geschreven door teamleden is eigendom van het ganse team.
- Stimuleer de samenwerking en teamgeest. Doe een beroep op de verantwoordelijkheid van alle teamleden bij het oplossen van problemen of zoeken naar antwoorden op vragen.
- Geef in geval van discussie iedereen de kans om zijn mening uit te spreken. Probeer een

consensus te bereiken, een oplossing waar het ganse team zich in kan vinden.

DO NOT

- Beschouw jouw code niet als jouw persoonlijke eigendom, maar als eigendom van het ganse team.

5.2.10. Flexibility

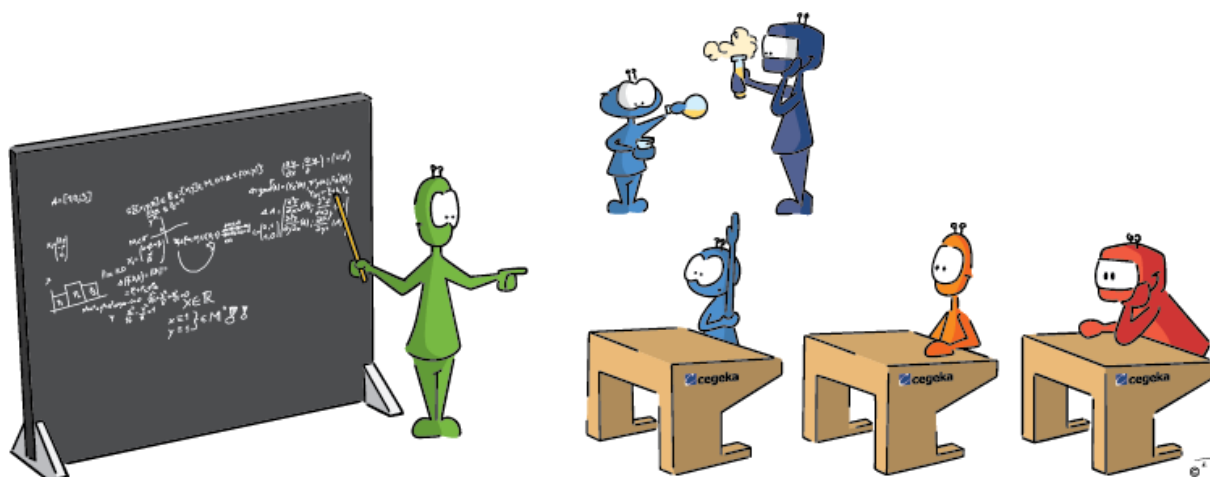


- Een ontwikkelingsproject is een organisch gegeven, een 'work in progress'. Stel je soepel op en verwelkom wijzigingen die zich opdringen.
- Wees bereid om van koers te wijzigen en te veranderen zodra dit een verbetering blijkt.

DO NOT

- Klamp je niet vast aan een vooropgesteld plan. Durf de planning los te laten.

5.2.11. Continuous learning



- School jezelf voortdurend bij.
In een ICT-omgeving volgen nieuwe ontwikkelingen elkaar snel op. Om te programmeren heb je steeds nieuwe kennis nodig. Leren is een essentieel onderdeel van het vak. Stilstaan is achteruitgaan.
- Leer van alles wat je doet, van iedereen met wie je samenwerkt.

DO NOT

- Ga er nooit vanuit dat je niets meer kunt leren.
- Wees niet bang om fouten te maken. Beschouw fouten als een uitdaging, als een kans om te verbeteren, ervaring op te doen en meer kennis te verwerven. Let us make new mistakes, and learn new things!
- Aarzel niet om taken op te nemen die je uitdagen en de grenzen van je kennis verleggen.

5.2.12. Tips

Nog een paar tips:

Do's:

- Listen really to each other (respectful).
- Ask open, clarifying questions.
- Focus on solutions.
- Build consensus and agreement in the team.
- Give positive, constructive feedback.
- Foster a culture where it is OK to learn.
- Create learning opportunities.
- Trust the team to get the job done.
- Fix problems on the spot.
- Celebrate the success of every release.

Dont's:

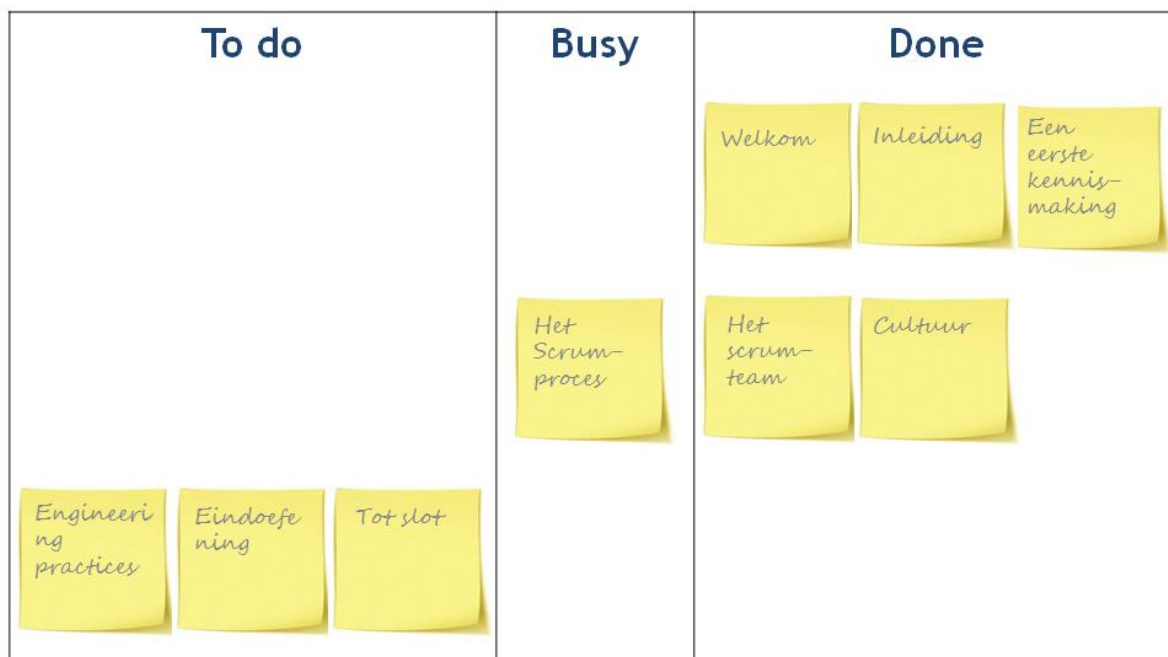
- Do not ask criticizing questions.
- Do not sit back and wait for orders.
- Do not get emotional, speak about facts.
- Do not be stubborn. Do not assume, just ask. Do not drag on with conflicts.
- Do not blame each other.
- Do not criticize for mistakes. Do not become complacent.
- Do not be protective over what you see as 'your' stuff.

5.3. Opgaven

Maak het kruiswoordraadsel in de webversie van deze cursus.

Hoofdstuk 6. Het scrumproces

Scrumbord



6.1. Inleiding

Je weet al hoe het Scrumproces in grote lijnen verloopt. In dit hoofdstuk gaan we dit verder in detail bekijken.

We beginnen met de voorbereiding van de eerste sprint: Wat heb je nodig voor je aan het proces kan beginnen?

Nadien bekijken we de verschillende fasen die telkens terugkeren:

- start van de sprint,
- tijdens de sprint,
- einde van de sprint.

6.2. Voor de eerste sprint

Voor je aan de eerste sprint begint, moeten er een aantal zaken geregeld worden.

6.2.1. Product backlog

De product backlog is een geprioriteerde **lijst met user stories** of korte beschrijvingen van alle functionaliteiten die nodig zijn om het eindproduct te realiseren.

De **product owner** is verantwoordelijk voor het opmaken van de product backlog. Doorheen het

project komt hij regelmatig samen met de klant om te bespreken welke vereisten zeker dienen ontwikkeld te worden en welke het dringendst zijn. Vervolgens vertaalt hij deze vereisten in user stories en lijst deze op in de product backlog. Hierbij rangschikt hij ze volgens prioriteit. Ze worden niet meer uitgebreid gedocumenteerd zoals bij de watervalmethode.



Een **user story** beschrijft kort een **functionaliteit** vanuit het standpunt van de klant of de gebruiker. Een user story is geen technische beschrijving, maar is opgesteld in een taal die begrijpbaar is voor de klant.

Een user story wordt vaak geformuleerd in de vorm:

Als wie kan ik wat zodat waarom

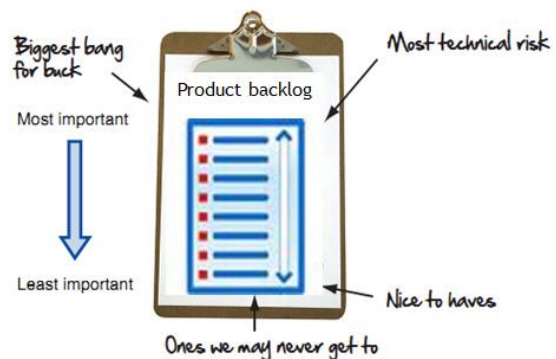
Een voorbeeld zou zijn: "Als klant, kan ik de items in mijn winkelwagentje bekijken voor het betalen *zodat* ik kan zien wat ik al gekozen heb."

De **prioriteit** van een functionaliteit wordt bepaald door

- de waarde die de functionaliteit aan het product toevoegt,
- de noodzakelijkheid van de functionaliteit,
- de dringendheid van de functionaliteit.

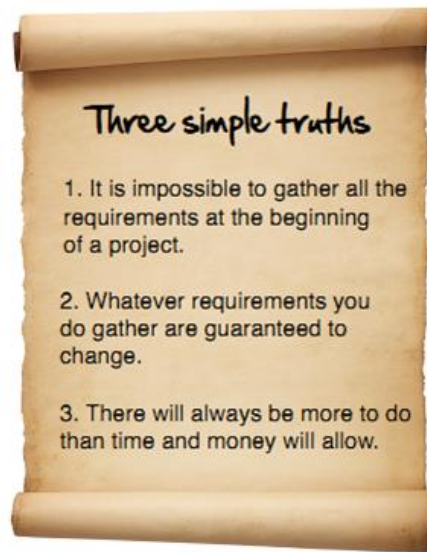
Het ontwikkelteam werkt altijd eerst aan de meest dringende vereisten.

De product backlog is een **dynamische lijst** die nooit af is. Zolang het project loopt en aan de software gewerkt wordt, wordt aan de product backlog gewerkt. Aangezien een Scrumteam werkt op maat van de klant kan de product owner op elk ogenblik functionaliteiten (of de dringendheid ervan) aanpassen, wijzigen of schrappen op vraag van de klant.



6.2.2. Enkele tips: Drie eenvoudige waarheden

Bij het opstellen van de product backlog wordt rekening gehouden met volgende "waarheden":



6.2.3. Scrum room

Scrum drijft op ad hoc-communicatie tussen de teamleden. Het hele team zit daarom in **één ruimte**. Altijd. Zelfs één deur tussen twee teamleden vermindert de communicatiebandbreedte tussen die twee teamleden.



De Scrum room...

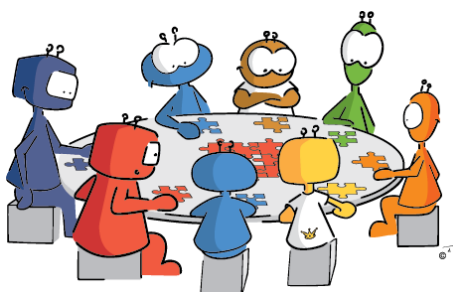
- hoeft niet toegewezen te zijn als "enkel voor het Scrumteam".
Er mogen niet-projectleden bij zitten. Maar een Agile Design project maakt veel lawaai, dus aan te raden is het niet.
- moet veel **muurruimte** hebben.
Eén van de belangrijkste pijlers van Scrum is: maak alle informatie zichtbaar en transparant.
- moet een ruime **voorraad** pennen, papier, post-its in alle kleuren en tape bevatten.
- moet echt **VAN** het team zijn. Maak het **eigen**.



6.3. Start van de sprint

6.3.1. Inleiding

Bij de start van elke sprint moet er eerst bekeken worden waar er in deze sprint aan gewerkt zal worden. In dit onderdeel bekijken we hoe dit gebeurt.



6.3.2. Sprint planning meeting

Aan het begin van elke sprint is er de **sprint planning meeting**. Tijdens deze meeting wordt het werk dat in de aankomende sprint uitgevoerd zal worden, besproken en gepland.

De teamleden overlopen en bespreken de product backlog. De product owner levert de input voor deze meeting.

De product owner **presenteert** de stories voor de aankomende sprint aan het ontwikkelteam. Hij geeft aan waar de prioriteiten liggen en bespreekt met de ontwikkelaars welke stories zij de volgende periode denken te realiseren.

De ontwikkelaars overlopen deze user stories en zetten ze om in losse taken of **work-items**. Hierbij gebruiken ze technische taal die enkel ontwikkelaars begrijpen.



6.3.3. Planning poker

Het planning poker spel wordt door de ontwikkelaars gebruikt om gezamenlijk in te schatten **hoeveel tijd** (of story points) het zal kosten om een work-item volledig af te werken.

Alleen het ontwikkelteam is in staat om in te schatten hoeveel stories zij kunnen realiseren - d.w.z. ontwikkelen, implementeren, testen en afleveren - binnen de geplande sprint. Ze baseren hun schatting op vroegere ervaringen met gelijkaardige taken.

Er wordt gespeeld tot er een consensus wordt bereikt.

De ontwikkelaars bepalen zo gezamenlijk de doelstelling of de scope van de sprint en leggen deze vast in de sprint backlog.

In de webcursus kan je in een filmpje bekijken hoe planning poker in zijn werk gaat.

6.3.4. Sprint backlog

Het resultaat van de sprint planning meeting is de **sprint backlog**.

Zoals gezegd bekijkt het team de product backlog en bepaalt samen aan welke items zij in de komende sprint zullen werken. Deze **user stories** worden uit de product backlog gelicht en komen terecht op de sprint backlog.

In het tweede deel van de sprint planning meeting wordt voor elke user story uitgeschreven welke **specifieke technische taken** nodig zijn om deze user story af te werken.

Meestal schat het team ook in hoeveel **tijd** of **story points** zij denken nodig te hebben om elke taak af te werken. Het aantal uur nodig om een taak af te werken, wordt met een simpel cijfer in de sprint backlog achter de taak vermeld.

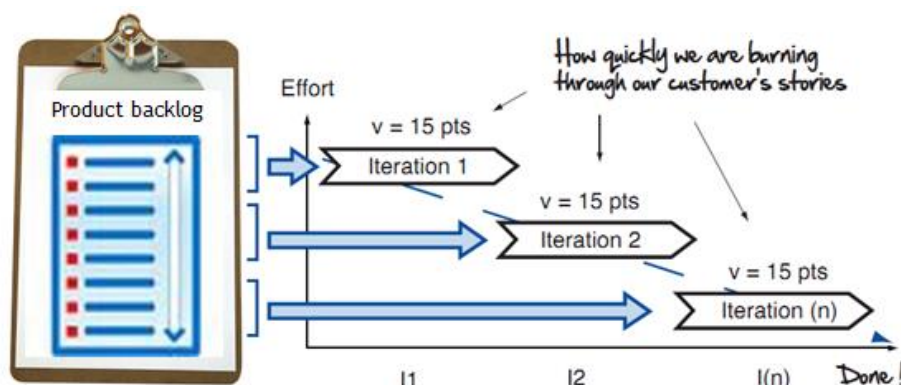
Een sprint backlog bestaat doorgaans in de vorm van een spreadsheet. Deze spreadsheet geeft een beeld van het te verrichten werk voor de eerstkomende sprint en kan door elk teamlid kan geraadpleegd worden. Tijdens de sprint mag enkel het team nog wijzigingen aanbrengen aan de sprint backlog.

Hieronder vind je een voorbeeld van zo'n spreadsheet:

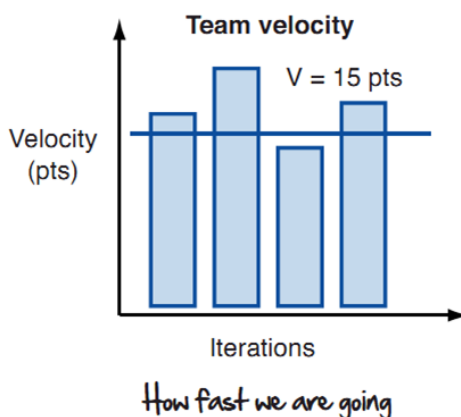
User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

6.3.5. Hoeveel stories in één sprint?

Bij het opstellen van de sprint backlog, ga je uit van een bepaald aantal story points dat je wil afwerken:



In onderstaande grafiek tonen we het aantal afgewerkte story points. Op basis van deze gegevens kunnen we bepalen hoeveel storypoints we gemiddeld per sprint kunnen afwerken.



Het inplannen van de sprint moet **realistisch** gebeuren. Het heeft immers geen zin om systematisch te overplannen. Dit is niet goed voor het vertrouwen van het team en de klant.

Op basis van hetgeen je de voorbije sprint heb gerealiseerd, ga je een vooruitspiegeling maken hoeveel je kan opnemen de volgende sprint. Je kan dit doen op basis van één sprint en telkens aanpassen voor elke nieuwe sprint. In de praktijk gaan we vaak met gemiddelde cijfers werken over verschillende sprints heen. Dit principe noemt men **yesterdays wether**.

6.4. Tijdens de sprint

6.4.1. Inleiding

In dit onderdeel bekijken we hoe het er tijdens de sprint aan toe gaat:

- Hoe wordt het Scrumbord gebruikt?
- Hoe verloopt de daily standup?
- Hoe wordt de vooruitgang van de sprint opgevolgd?



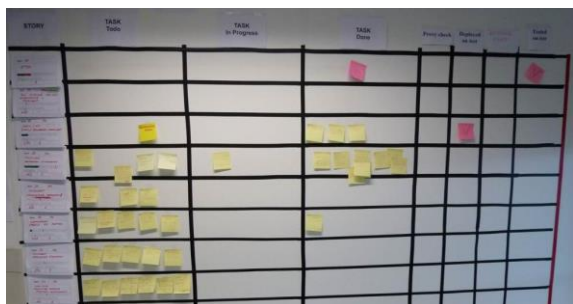
6.4.2. Het Scrumbord

De work-items van de sprint backlog worden op losse **taakbriefjes** geschreven en op het Scrumbord bevestigd. Bij elke taak wordt vermeld hoeveel story points ervoor voorzien zijn.

Het Scrumbord neemt een centrale plaats in binnen de werkruimte van het ontwikkelteam. Het is opgedeeld in 3 grote vakken: **TO DO** / **BUSY** / **DONE**. In het begin van de sprint worden alle taakbriefjes in het linkervak to do bevestigd. Bovenaan hangen de taken met de hoogste prioriteit. Deze worden eerst afgewerkt.

De ontwikkelaars nemen elk een taakbriefje uit het vak to do en verplaatsen dit naar het vak busy. Daar blijft het hangen zolang ze ermee bezig zijn. Zodra ze klaar zijn met de taak hangen ze het briefje in het vak done.

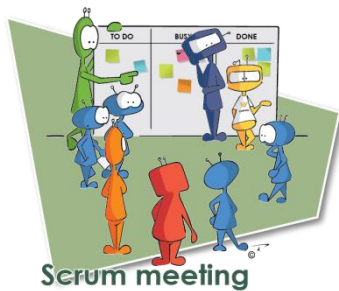
Het Scrumbord is een **visuele** voorstelling van de **verdeling** en de **voortgang** van het werk. Op die manier is voor alle betrokkenen zichtbaar wie met welke taak bezig is en hoe het werk vordert.



Als Scrumbord kan men gebruik maken van een simpel wit magneetbord waarop post-it's worden bevestigd. Maar men kan ook beroep doen op een gesofisticeerder digitaal systeem.

Elk ontwikkelingsteam beslist zelf op welke manier zij het scrumbord organiseren. Zo kan men er als team voor opteren om door middel van verschillende kleuren (bv. groen, rood, oranje) de urgentie van bepaalde taken weer te geven.

6.4.3. De Daily Scrum of Standup Meeting



Een belangrijk moment tijdens de sprint is de **daily scrum**. Deze korte meeting wordt dagelijks georganiseerd door de Scrum master. De leden van het ontwikkelteam verzamelen bij het Scrumbord. Om beurt beantwoorden ze 3 vragen:

- Wat heb ik **gisteren** gedaan?
- Wat ga ik **vandaag** doen?
- Heb ik ergens **problemen** mee?

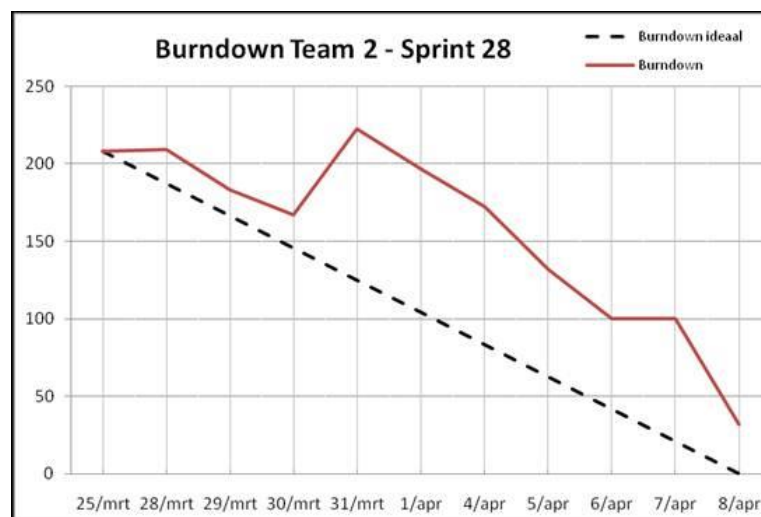
De daily scrum wordt steeds op hetzelfde tijdstip gehouden en duurt meestal niet langer dan 15 minuten. Het is een actieve vergadering waarbij iedereen even aan het woord komt. De bedoeling is dat iedereen gedurende de bijeenkomst blijft rechtstaan. Vandaar dat de daily scrum ook de **standup meeting** wordt genoemd. Problemen of obstakels die aangekaart worden tijdens deze meeting, kunnen daarna aangepakt worden.

Bij een daily scrum kan wel wat mis gaan. In het filmpje in de webcursus zie je een slechte standup-meeting.

6.4.4. Burndown chart

De **burndown chart** is een grafische samenvatting van de vooruitgang in de sprint.

In één oogopslag kan men zien of alles volgens plan verloopt. De grafiek laat zien hoe snel het team werkuren aan het 'verbranden' is en of het werk snel, traag of volgens plan vordert.



6.4.5. Inflexibiliteit

Tijdens de sprint gaat men zeer gedisciplineerd te werk: er is volledige inflexibiliteit ten aanzien van

- de hoeveelheid werk en
- de kwaliteitsdoelstellingen

die tijdens de sprintplanningsbijeenkomst werden vastgelegd. Voor de eerstvolgende sprint ligt alles nog open.

6.4.6. DoD - Definition of Done

Een taak is pas volledig afgewerkt als deze aan een aantal criteria voldoet. De DoD of **Definition of Done** is een **checklist** die wordt gebruikt om te bepalen of de taak aan alle kwaliteitseisen voldoet.

6.5. Einde van de sprint

6.5.1. Inleiding

Elke sprint wordt afgesloten met twee belangrijke vergaderingen:

- de **Sprint Retrospective**
- de Demo of **Sprint Review** meeting

In dit onderdeel bekijken we wat er tijdens beide meetings besproken wordt.



6.5.2. Sprint retrospective

Bij de Sprint Retrospective kijken de medewerkers terug op de voorbije sprint en het geleverde werk. Gezamenlijk gaan ze na wat er goed liep de afgelopen weken en wat kan verbeterd worden. De Agile methodologie gaat uit van een zelfkritische houding en legt de nadruk op **continue verbetering**. Elke sprint kan beter worden dan de sprint ervoor.

Tijdens de retrospective bijeenkomst **evalueren** de medewerkers elkaars werk, geven feedback en stellen verbeteringen voor. De verschillende leden worden niet als individuele medewerkers beschouwd, maar het team wordt gezien als één geheel. De resultaten - goed of slecht - zijn de resultaten van het ganse team en niet van één persoon.

Tijdens de retrospective kunnen verschillende technieken gebruikt worden. Het is aan het team om zelf een methode te kiezen. Dit kan ook elke sprint anders gebeuren.

Bv. Mad, sad, glad - Like to like - Timeline (sprint, release, project) - Stop / start / continue doing -





6.5.3. De Demo of Sprint Review meeting

De Demo of Sprint Review meeting is een live **demonstratie** van de functionaliteit(en) of het werkende stukje software dat volledig werd afgewerkt tijdens de voorbije sprint.

De klant en de product owner **evalueren** de functionaliteit en geven feedback. Indien er nog aanpassingen moeten gebeuren dan wordt dit opgenomen tijdens de volgende sprint.

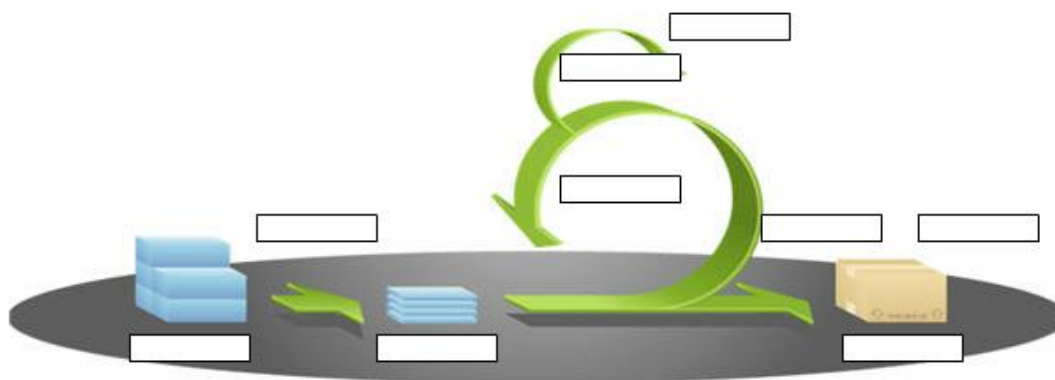
6.6. Opgaven

6.6.1. Opgave 1

Vul het schema aan

Sleep elk item naar de juiste plaats in het schema.

Sprint backlog	Sprint	Demo	Daily scrum
Retro spective	1 dag	Sprint planning	Afleverbaar product
Product backlog			



Opgave

2: Scrum in een andere context

We keren terug naar de bouw van een huis.

Zou je bij elke stap van het Scrumproces iets uit dit bouwproces kunnen plaatsen?

- product backlog
- sprint planning

- sprint backlog
- demo
- afleverbaar product

Stuur je oplossing van deze opdracht door aan je coach.

Hoofdstuk 7. Engineering practices

Scrubbord



7.1. Beproefde technieken

7.1.1. Wat?

Scrum gaat ervan uit dat kennis en vakmanschap ontstaan uit ervaring. De praktijk wijst uit wat de beste manier van werken is. Beslissingen worden genomen op basis van vroegere positieve en negatieve ervaringen.

Engineering practices zijn beproefde **werkwijzen** die tijdens eerdere Scrumprojecten met succes werden toegepast en hun effectiviteit hebben bewezen. Elk Scrumteam kan beslissen om deze werkwijzen in volgende projecten opnieuw te gebruiken. Uiteraard is men vrij om ze op basis van nieuwe ervaringen aan te passen of weg te laten.

Veel voorkomende engineering practices zijn:

- Pair programming,
- Refactoring,
- Test driven development,
- Collective ownership.

7.1.2. Pair programming

Bij pair programming wordt één user story door **2 ontwikkelaars** ontwikkeld. Zij zitten samen aan

één computer aan hetzelfde stuk code te werken. Afwisselend zijn zij navigator of driver.

De **driver** bedient het toetsenbord en voert de code daadwerkelijk in.

De **navigator** heeft een controlerende en tegelijk vooruitdenkende taak. Hij volgt mee en kijkt vooruit naar waar ze met de code naartoe gaan.

De 2 ontwikkelaars wisselen regelmatig van rol. Zo worden de sterke punten van beide programmeurs benut.

Door steeds met twee personen te programmeren neemt de programmeertijd misschien wel toe, maar:

- fouten worden snel gedetecteerd en gecorrigeerd;
- het programma bevat minder lijnen code;
- meerdere personen binnen het team kennen dat onderdeel van de code.



In de webcursus vind je nog een filmpje over pair programming.

7.1.3. Test driven development

Test driven development is een methode waarbij eerst de tests voor de code worden ontwikkeld en pas daarna de code zelf.

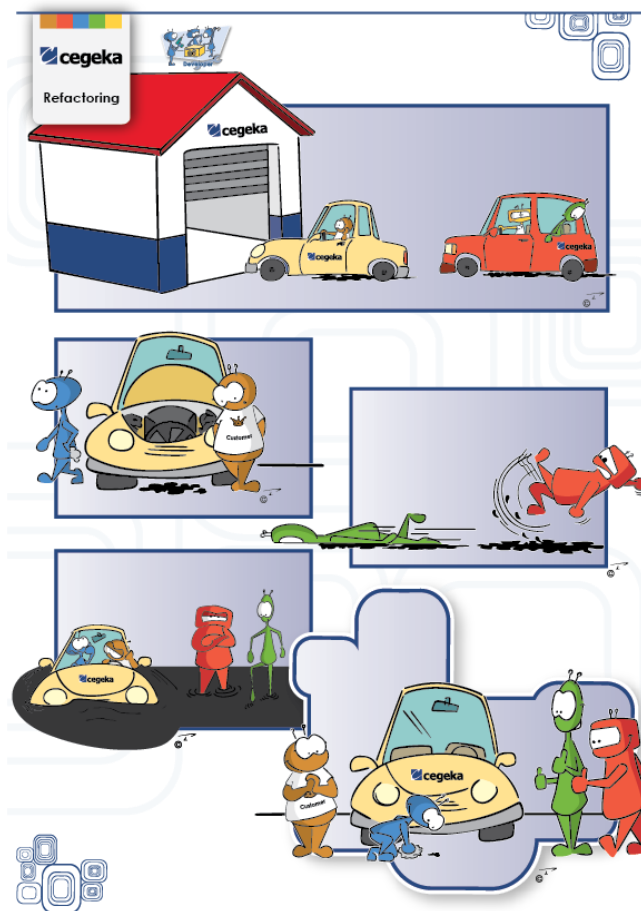
Ondanks de extra tijd die nodig is voor het schrijven van de tests, blijkt uit de praktijk dat de ontwikkeltijd toch korter wordt. Fouten worden al in een zeer vroeg stadium ontdekt en rechtgezet. Door de code van in het begin te testen spaart men dus tijd uit.

Ook hierover vind je een filmpje in de webcursus.

7.1.4. Refactoring

Bij refactoring wordt de broncode van een computerprogramma gedisciplineerd en stapsgewijs herlezen en herschreven. Refactoring heeft als doel de code te **vereenvoudigen** om zo de leesbaarheid en bijgevolg de onderhoudbaarheid ervan te verbeteren. Het refactoren of herschrijven van de code verandert de werking van de software niet.

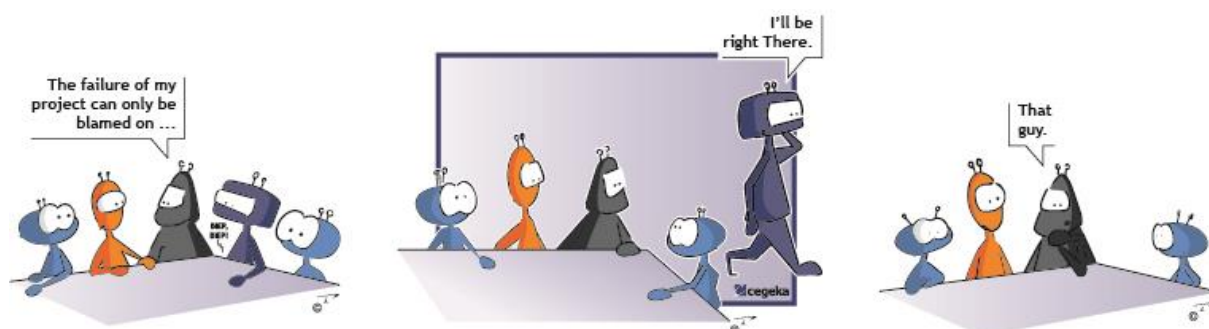
Het voordeel van 'propre' of eenvoudige en leesbare code is dat ze achteraf makkelijker onderhouden en aangepast kan worden.



7.1.5. Collective ownership

Collective ownership gaat ervan uit dat de verschillende stukjes code ontwikkeld door de afzonderlijke ontwikkelaars eigendom zijn van het ganse team. Hierdoor kan elk lid van het team elk stukje code in het systeem op elk moment wijzigen.

Samen met pair programming zorgt collective code ownership ervoor dat de kennis van de code en het systeem verspreid wordt over het hele team.



Hoofdstuk 8. Eindoefening

Scrumbord

To do	Busy	Done
		Welkom
		Inleiding
		Een eerste kennis-making
		Het scrum-team
		Cultuur
		Het Scrum-proces
Tot slot	Eindoefening	Engineer- ing practices

Download onderstaande lijst met stories in de webcursus.

Opdracht

1. Wat is volgens jou de kleinste story, m.a.w. de story die het minste tijd zal vragen om uit te voeren?
Waarom kies je deze story?
2. Sorteert de stories volgens workload. De stories die je het snelst kan afhandelen, zet je bovenaan.
3. Stel je hebt een sprint van 1 uur. Stel de sprintbacklog op voor de eerste sprint.
(Je kan gewoon een kruisje zetten in de excel file in de kolom Sprint 1.)

Stuur het antwoord van deze opdracht door aan je coach.

Hoofdstuk 9. Einde cursus

Scrumbord

To do	Busy	Done		
		Welkom	Inleiding	Een eerste kennis-making
		Het scrum-team	Cultuur	Het Scrum-proces
	Tot slot	Engineering practices	Eindoefening	

9.1. Tot slot

Beste cursist,

Je bent nu aan het einde gekomen van de webcursus Scrum. Proficiat!

Enquête

We horen graag wat je van deze cursus vindt. Vul het enquêteformulier zeker in!

Afdrukbare cursus

Wil je een afdrukbare versie van deze cursus? Stuur dan een bericht met als onderwerp "Afdrukbare versie naar de coach". Deze zal je vervolgens een downloadlink bezorgen waarmee je de cursus in pdf-formaat kan downloaden.

Attest

Wat de administratieve afhandeling betreft: je hoeft niets te doen om de cursus af te sluiten. Dit gebeurt automatisch na het verlopen van je leerperiode. Wanneer je precies je deelnameattest ontvangt, lees je hier .

We wensen je nog veel succes en tot een volgende keer.

De coaches.

9.2. Bronnen

Bij de ontwikkeling van deze cursus hebben we volgende bronnen geraadpleegd:

- *De kracht van scrum*, Rini van Solingen & Elco Rustenburg, Pearson Education