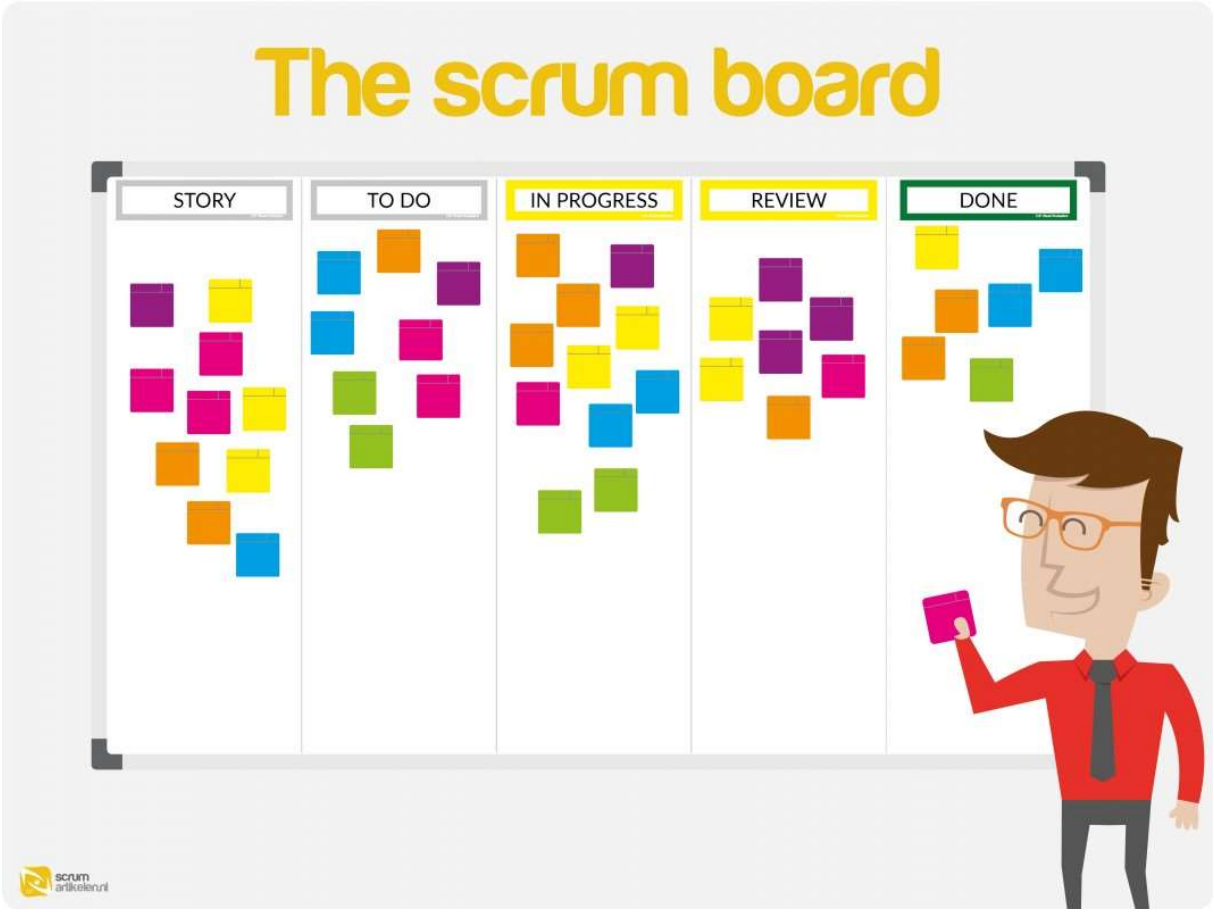
AGILE METHODIEKEN



Nick Kruize 307366

Adaptive Software Development (ASD)

- Activiteiten binnen een sprint worden vergeleken met het projectdoel. Mogelijk moet het projectdoel worden aangepast
- Nadruk ligt op het resultaat en de kwaliteit hiervan. Minder nadruk of de benodigde taken/processen om dit resultaat te bereiken.
- Iteratief proces.
- Time boxed -> d.m.v concrete deadlines wordt gezorgd dat beslissingen aan het begin van het project worden genomen en niet worden uitgesteld.
- Change-tolerant systeem er wordt continu gekeken of de gebouwde onderdelen zouden kunnen veranderen in de toekomst.
- Risk driven. Component die een hoger risico met zich meebrengen worden eerst ontwikkeld.

PROCES:

1. Speculate
2. Collaborate
3. Learn

Speculatie wordt gebruikt als alternatief voor planning. Doordat de eisen aan het project continu kunnen veranderen wordt er geen gespeculeerd over wat er bereikt gaat worden. Een planning impliceert dat hiervan niet kan worden afgeweken. Schema's en doelen per ontwikkelingscycli (sprint in scrum) worden hier opgesteld. Een cycli duurt normaal tussen de 4 en 8 weken.

Binnen Learn zijn kwaliteitsreviews van groot belang. Hierbij is de klant normaal aanwezig als expert over het te ontwikkelen product. Een ander belangrijk onderdeel binnen Learn is het "final Q/A and release". Binnen dit onderdeel staat het reflecteren op de geleerde lessen gedurende de iteratie centraal.

CONCLUSIE

ASD is resultaat gefocust wat niet de prioriteit heeft voor ons als studenten op dit moment. Echter het constant in de gaten houden van het projectdoel en of de gedane activiteit binnen dit doel vallen is een belangrijk onderdeel dat wij binnen ons groep hebben gebruikt. Wij hebben dit gedaan door een mvp te beschrijven en eerst deze functionaliteit te bouwen. Het reflecteren op geleerde lessen is een ander belangrijk onderdeel van ASD dat bruikbaar is voor ons.

Kanban

- Visualisatie van het werk dat wordt ondernomen
- Beperk de hoeveelheid werk
- Verplaatsen werk van kolom naar kolom
- Observeren, bijhouden en verbeteren

KANBANBORD

Het kanbanbord heeft de bekende To Do, Doing and Done categorieën. Elk kaartje op het bord vertegenwoordigt een product binnen het productieproces. De beweging van een kaartje van kolom naar kolom toont de flow van het product door het productieproces.

Binnen softwaredevelopment worden vaak de kolommen: Backlog, Ready, Coding, Testing, Approval en Done of anders Next, In Development, Test, Acceptatie, Live gebruikt.

TWO BIN

De twee bakken binnen het two bin systeem bevatten respectievelijk de werkvoorraad dit op dit moment gebruikt wordt in de productie en een reservevoorraad. Hierbij denken we aan de opgestelde requirements en de prioriteit die hieraan is gegeven door de klant. Een equivalent naar scrum zou de productbacklog zijn.

CONCLUSIE

Het bord beheer dat komt kijken bij kanban heb ik binnen meerdere projectgroepen al gebruikt en geeft een handig overzicht van de staat van het proces. Het systematisch bijhouden van het bord is zeer belangrijk anders zorgt het voor meer verwarring dan het inzicht verschaft.

Het onderhouden van het bord hebben wij gedurende de loop van het project verbeterd. Nu worden taken direct afgevinkt als ze af zijn. Hierdoor is voor mij veel duidelijker geworden waar iedereen staat binnen zijn taak.

Extreme Programming (EP)

based on principles of simplicity, communication, feedback, and courage

Binnen EP draait alles om code maken. Ontwikkelaars reviewen elkaar onderling dit gebeurt continu. Alle software wordt ontwikkeld in koppels, Pair Programming. Koppels worden regelmatig gerouleerd. Dit zorgt ervoor dat de hele productiestaf zich meer betrokken voelt bij het product.

- Peer-review bewezen krachtiger tegen bugs dan systematisch testen.
- Zorgt voor groepsgevoel bij het ontwikkelen
- Uiteindelijke resultaat bestaat minder uit losse stukjes code die aan elkaar zijn gebonden, productiegroep raakt op elkaar afgestemd en vormt een geheel.
- Voordeel bij het inwerken van nieuwe teammembers, want er is een natuurlijke werkend-leren omgeving.
- Iedereen die aan het systeem werkt (inc klant) leert te spreken in dezelfde benamingen voorkomt verwarring.

TEST-DRIVEN-DEVELOPMENT (TDD)

Binnen XP worden unit tests geschreven voordat het uiteindelijke product wordt ontwikkeld. De reden hiervoor is dat het de ontwikkelaar dwingt na te denken over de functies en uitzonderingen van het stuk programma dat hij gaat maken. **Wat moet het programma doen**? Wordt belangrijker dan **Werkt het programma?**

Er wordt ontwikkeld vanuit test cases in plaats van functionele requirements. De klant wordt nauw betrokken bij de ontwikkeling van het product en het opstellen van de test cases.

Het ideaalbeeld van XP is om de iteraties zo kort mogelijk te maken. In de praktijk duren iteraties 1 tot 3 weken.

Binnen XP verloopt het plannen van het werk in twee fases Releaseplanning en Iteratieplanning. In de releaseplanning wordt bepaald in welke release welke functionaliteit bevat. In de iteratieplanning worden de stories uit de releaseplanning onderverdeeld in taken. Bij de releaseplanning is de klant aanwezig. Bij de iteratieplanning enkel het team. Elke planning bestaat uit onderdelen: Exploration, Commitment en Steering.

Releaseplanning

- Exploration: opstellen belangrijkste user story's van het systeem
- Commitment: user story's voor de volgende release bepalen en release oplevermoment bepalen
- Steering: aanpassing aan het plan, toevoegen/verwijderen story's

Iteratieplanning

- Exploration: story's vertalen naar taken, taakkaarten maken.
- Commitment: schatting maken van benodigde tijd per taak. Ontwikkelparen maken en taken toewijzen
- Steering: Uitvoeren toegewezen taken, de bestede tijd bijhouden en vergelijken met de geplande tijd voor volgende iteraties.

CONCLUSIE

Binnen het project werken we al vanuit requirements. Ontwikkelen vanuit unit tests lijkt een lastige overstap midden in een project.

Ontwikkelen in koppels is een onderdeel van XP dat wel geïntegreerd zou kunnen worden. Binnen mijn project in het eerste jaar heb ik hier bij de lastigere onderdelen van ons project al gebruik van gemaakt. Dat werkte toen goed, omdat het voorkwam dat mensen vast kwamen te zitten op een onderdeel en te lang wachtte met het vragen om hulp. De directe feedback/review/input zorgde ook voor het minder moeten herschrijven van code.

Tijdens dit project heb ik ondervonden dat pair-programming lastiger was in een online werkomgeving. Als je de persoon waarmee je samenwerkt niet kunt zien is het moeilijk om te zien hoezeer zij meekijken. Ik zou dan ook als tip geven om regelmatig een mening van je partner te vragen met als doel om de aandacht vast te houden. Regelmatig wisselen van rol binnen het duo zou ook kunnen helpen om het soepeler te laten verlopen.

Bronnen

1. https://nl.wikipedia.org/wiki/Agile-softwareontwikkeling
2. https://nl.wikipedia.org/wiki/Kanban
3. https://nl.wikipedia.org/wiki/Extreme_programming
4. https://nl.wikipedia.org/wiki/Adaptive_Software_Development
5. https://www.guru99.com/agile-scrum-extreme-testing.html
6. https://www.blueprintsys.com/agile-development-101/agile-methodologies