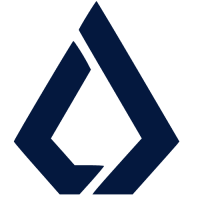
Case study: DevSecOps

Seven step approach



Project: Lisk Delegates

Project team: BBB-BC

Team: Joeri Berman (2834499)

Luc Urlings (3071081)

Stijn Baltessen (3064565)

Wouter Vandenboorn (3099571)

Floris Feddema (3188256)

Davy de Haas (3099695)

Opdrachtgever: Study case, Bartosz

Versie: 0.1

Versie datum: 21 april 2020

Status: Concept

# Document revisies

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Wijzigen** | **Auteur** | **Datum** |
| 0.1 | Opzet Document | Stijn Baltessen | 21-04-2020 |

# Inhoudsopgave

[**Document revisies**](#_bu2l4jqkimrr) **2**

[**Inhoudsopgave**](#_7rakfz1zgi71) **3**

[**Introductie**](#_eko9gepalqi) **5**

[**Beschrijving case**](#_2lz4yx4irrfw) **6**

[**Seven step approach**](#_djfvx4apkl6h) **7**

[**1. Bespreek de casus**](#_rmnkvys08yig) **8**

[**2. Identificeer de vragen**](#_ihx67e8uutgv) **9**

[**3 + 4. Brainstorm & Analyseer**](#_uaauncq6r1hc) **10**

[**5. Formuleer de leerdoelen**](#_gbd35qg6f1fh) **12**

[**6. Individuele studie**](#_s348kggz7kw0) **13**

[Wat zijn de uitdagingen die organisaties tegenkomen op een groot schaalbaar project met betrekking tot SDLC?](#_irlvcbu997la) 13

[Communicatie tijdens de eerste fase van het de life cycle](#_uio41zpw27k2) 13

[Schatting van capaciteiten en planning](#_4ulwpw9o4zgk) 13

[Input van klanten en de communicatie hiervan](#_xxgk9oecw18s) 13

[Het testen van het product](#_ngjk7yy09qm9) 14

[Wat is MTTR en waarom is dit belangrijk voor de klant?](#_alcu57tpxls6) 14

[Wat houd DevSecOps in in toevoeging tot DevOps?](#_brkkd5fl0h6g) 15

[Wat houdt DevOps in?](#_m16ft7sz00k) 15

[Wat houdt DevSecOps in?](#_orzp41ph99jz) 16

[Hoe geef je een DevSecOps setup schematisch weer?](#_1j2c3znujkfx) 17

[Wat is de beste manier om versiebeheer te implementeren?](#_ul5pqmuvv87h) 18

[Wat is de beste manier om agile toe te passen in de case?](#_15se45fautah) 19

[**7. Discussieer de bevindingen**](#_2mq0whlug7jd) **21**

[**Advies voor casus**](#_3kkm29337r15) **21**

# Introductie

Om de study case DevSecOps op te lossen gaan we de seven step approach toepassen, in dit document zal elke stap worden uitgelegd. Uiteindelijke conclusie wordt op het einde opgeschreven.

Eerst wordt de study case uitgelegd en hierna de zeven stappen die genomen worden.

# Beschrijving case

# Seven step approach

Hier staan de zeven stappen van de [seven step approach](https://portal.fhict.nl/Studentenplein/LMC/1920vj/Software%20engineering/Afstudeerfase/EnterpriseSoftware/CaseStudies/SevenStepsApproach.docx) methode die bij probleem gebaseerd leren voorkomt.

1. Bespreek de casus

2. Identificeer de vragen

3. Brainstorm over potentiële bekende oplossingen

4. Analyseer resultaat van brainstorm sessie

5. Formuleer de leerdoelen en wat er nog geleerd moet worden

6. Individuele studie om de kennis van 5 te kunnen vergaren

7. Discussieer de bevindingen van 6 en hoe dit 5 en 2 beantwoord

# 1. Bespreek de casus

Alle development gaat naar één branch, de development branch. Deze branch is niet overzichtelijk en kan veel tijd kosten om hier op te developen.

Runtime van tests is te lang en de reden van falen van tests kan onbekend zijn.

Het rapporteren van problemen is te vaag, waardoor er geen goede beslissing kan worden gedaan over welk team het moet oppakken.

Er is geen correct agile board, hierdoor is het niet duidelijk welke features af zijn en welke niet.

Het oplossen van problemen duurt lang.

De klant heeft geen vertrouwen in de manier waarop hij of zij features moet requesten.

Het development team is gestresst en dit beïnvloedt de prestaties van het team in een negatieve vorm.

Er zijn 1000 tests, maar uit de zin “*The runtime of the test suite is large (e.g. a couple of hours) and sometimes tests fail without any known reason”* blijkt dat de tests geen hoge kwaliteit zijn.

Development team heeft CI/CD pipeline opgezet en de deployment van de applicatie, is dit niet de verantwoordelijkheid van het operations team.

**De bovenstaande problemen komen neer op:**

* Geen juist overzicht bij het creëren van testen, issues, features, versiebeheer en de verantwoordelijkheid per team.
* Werkdruk is te hoog voor het development team.
* Versiebeheer methode.
* Unit test resultaat fout geschreven.
* Het team is niet in staat om security threats te kunnen oplossen.

# 2. Identificeer de vragen

* Wat is SDLC precies en wat zijn de uitdagingen die organisaties tegenkomen op een groot schaalbaar project?
* Wat is een change request procedure en waarom moeten zowel bedrijven als klanten deze procedure volgen?
* Wat is MTTR en waarom is dit belangrijk voor de klant?
* Zijn er bestaande tools, technieken, patterns of best practices die kunnen worden gebruikt binnen deze case study?
* Wat is de beste manier om versiebeheer te implementeren?
* Wat is de beste manier om agile toe te passen?
* Hoe zorg je ervoor dat alle unit en integration tests van hoge kwaliteit zijn?
* Hoe kan je een overzicht krijgen van de tests?
* Hoe geef je een DevSecOps setup schematisch weer?
* Wat zijn de extra punten die komen bij DevSecOps in vergelijking met DevOps?

# 3 + 4. Brainstorm & Analyseer

* Wat is SDLC precies en wat zijn de uitdagingen die organisaties tegenkomen op een groot schaalbaar project?

Software development life cycle, uitdagingen moeten worden onderzocht voor een schaalbaar project.

* Wat is een change request procedure en waarom moeten zowel bedrijven als klanten deze procedure volgen?

Dit staat meestal beschreven in binnen het contract tussen de klant en het uitvoerend bedrijf. Als een klant aanpassingen of nieuwe features wilt binnen een project moet dat via een vooraf afgesproken weg worden geregeld. Hierdoor is er een vaste plek waar deze aanvragen kunnen worden ingediend en aan de hand van het contract wordt bepaald wat de prioriteit van een change request is.

* Wat is MTTR en waarom is dit belangrijk voor de klant?

MTTR staat voor mean time to repair. Dit betekent de tijd die het kost om een probleem op te lossen. Hierbij wordt niet beschreven wat hoelang een oplossing mag duren maar hoe lang het daadwerkelijk duurt om een fout op te lossen.

* Zijn er bestaande tools, technieken, patterns of best practices die kunnen worden gebruikt binnen deze case study?

Agile board, Git, Mercurial, Jenkins, TravisCI, Github Actions, Azure DevOps, Docker, Kubernetes, Event Storming, Jira.

* Wat is de beste manier om versiebeheer te implementeren?

Gitflow methode toepassen. verder onderzoek naar andere methodes.

* Wat is de beste manier om agile toe te passen?

Er zijn verschillende Agile methodes, ieder heeft z’n eigen voor en nadelen en zijn vaak sterk afhankelijk van de voorkeuren van het team en de toepasbaarheid in de context.

Deze gaan we niet beantwoorden.

* Hoe zorg je ervoor dat alle unit en integration tests van hoge kwaliteit zijn?

Bedenk goed wat het doel is van het project en hoe tests kunnen worden gebruikt voor het verhogen van de kwaliteit van de software.

Plannen en definiëren wat de strategie is. Dit houdt in;

* Het selecteren van de juiste tooling.
* Het identificeren van de risico's die kunnen opkomen als er geen tests voor zijn geschreven.
* Het maken van een testschema.
* Zorgen dat er zowel een goede flow wordt getest als alle foutieve.
* Communiceren met elkaar welke tests niet gehaald worden en waarom.
* één unit test is verantwoordelijk voor één ding, dit is belangrijk om te identificeren wat er mis gaat op het moment dat de tests falen.
* Hoe kan je een overzicht krijgen van de tests?

Het gebruikmaken van onder andere SonarQube of een Test Matrix. test rapport of coverage rapport.

* Hoe geef je een DevSecOps setup schematisch weer?

Hier hebben wij nog geen ervaring mee. Dit wordt binnen deze study case onderzocht.

# 5. Formuleer de leerdoelen

* De student weet hoe hij een juiste versiebeheer flow opzet en wat de grootste valkuilen zijn.
* De student weet wat DevSecOps is en wat de best practices zijn.
* De student weet de best practices van het maken van unit en integration tests.
* De student weet het tussen verschillende implementaties van versiebeheer workflows en de pro’s en cons hiervan.

# 6. Individuele studie

## Wat zijn de uitdagingen die organisaties tegenkomen op een groot schaalbaar project met betrekking tot SDLC?

### Communicatie tijdens de eerste fase van het de life cycle

Een van de grootste problemen komen op tijdens het samenstellen van de eisen van het product en de communicatie hiervan tussen de verschillende partijen.

Methodologieën zoals de watervalmethode leiden vaak tot problemen als het einddoel en de eisen niet goed zijn gedefinieerd. Hierdoor kan het voorkomen dat bepaalde stappen on hold moeten worden gezet voordat er verder kan worden gewerkt of dat deze kleine problemen in een later stadium grote problemen worden.

Andere methodologie kunnen ervoor zorgen dat deze problemen minder vaak voorkomen door betere communicatie tussen de verschillende partijen. Door bijvoorbeeld AGILE toe te passen wordt er om de zoveel tijd gecommuniceerd met de andere partijen zodat deze op een lijn blijven.

### Schatting van capaciteiten en planning

Binnen het team is het belangrijk om te weten wie welke capaciteiten heeft. Rond deze informatie kunnen duidelijke en goede planningen gemaakt worden. Het kan echter zo zijn dat grote bedrijven met grote of nieuwe teams niet van elkaar weten waar ze tot in staat zijn en wat ze kunnen. Dit kan gebrek aan kennis zijn of simpelweg het verkeerd begrijpen van iemand.

Ook kan gebrek aan ervaring een grote rol spelen binnen dit onderwerp. Als iemand een bepaalde rol krijgt zonder voorgaande ervaringen binnen die rol kan het langer duren of zelfs zorgen voor fouten.

### Input van klanten en de communicatie hiervan

Communicatie met een klant is een belangrijk deel van software ontwikkeling. Het is namelijk vrijwel onmogelijk om software te creëren die volledig aansluit op de wensen van een klant en hier en daar moeten mogelijk compromis worden gesloten. Hiervoor is het belangrijk om een constante communicatie tot stand te houden met de klant en ze te informeren over de status van het project. Het kan zomaar gebeuren dat deze communicatie verzwakt raakt, in zo'n gevallen kan het gebeuren dat de ontwikkelde onderdelen mogelijk niet aansluiten op de wensen van de klant.

Een klant kan ook tijdens de ontwikkeling zich bedenken dat een bepaalde feature toch belangrijker is dan aanvankelijk werd gedacht en desnoods nog bij het project moet worden inbegrepen. Het is dan aan het team om te communiceren met de klant of dit haalbaar is en hoe dit samenloopt met de bestaande planning. Het kan goed zo zijn dat om deze feature te implementeren een andere feature moet worden uitgesteld of soms zelfs worden geschrapt.

Een manier om deze problemen op te lossen zijn de Agile methodologieën, hierin worden klanten ver meegenomen binnnen het process. En door de korte planningen zorgt men ervoor dat er gemakkelijk ruimte te maken is voor veranderingen aan het originele plan.

### Het testen van het product

Het is belangrijk om het product goed en volledig te testen, hierdoor voorkom je bugs en mogelijke fouten binnen de software.

Echter kan het gebeuren dat er geen goed plan van aanpak is voor het management van de te maken testomgeving. In deze situatie kan het gebeuren dat er niet goed of volledig getest wordt. Het gehele process kan hierdoor ook langer gaan duren, uiteindelijk resulterend in meer kosten voor de klant.

## Wat is MTTR en waarom is dit belangrijk voor de klant?

MTTR (Mean Time To Repair) is een meetmethode voor het onderhouden en repareren van een defect onderdeel binnen het project. Het representeert de gemiddelde tijd nodig om van een defect te herstellen.

Dit onderdeel is belangrijk voor een klant omdat het een duidelijk inzicht geeft in de duratie van reparaties van het uitvoerende bedrijf. Als een storing zich voordoet heeft de klant aan de hand van de MTTR een indicatie van de mogelijk tijd dat het zal kosten om de storing te verhelpen.

## Wat houd DevSecOps in in toevoeging tot DevOps?

stappjes:

* devops inhoud kort, <https://blogs.cornell.edu/react/devops-vs-devsecops-what-is-the-difference/>
* devsecops inhoud kort
* devsecops inhoud dieper op ingaan toegevoegde waarde voor projecten

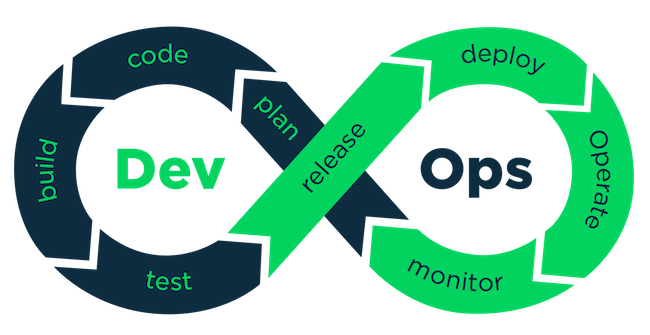
### Wat houdt DevOps in?

Om de vraag te beantwoorden wat het verschil is tussen DevSecOps en DevOps moeten we eerst definiëren wat devops is.

De DevOps-methodologie combineert twee elementen, development en operations. De afkorting ‘Dev’ staat voor development en ‘Ops’ staat voor operations.

Het doel van DevOps is om de snelheid van software delivery te verhogen door continuous collaboration, communication, automation en integration. Door het toepassen van DevOps in de gehele development pipeline hoeven developers minder tijd te besteden aan delivery waardoor ze meer tijd kunnen besteden aan andere prioriteiten.

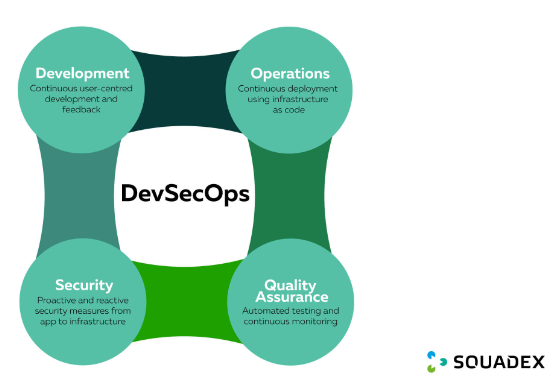
bron: <https://blogs.cornell.edu/react/devops-vs-devsecops-what-is-the-difference/>



### Wat houdt DevSecOps in?

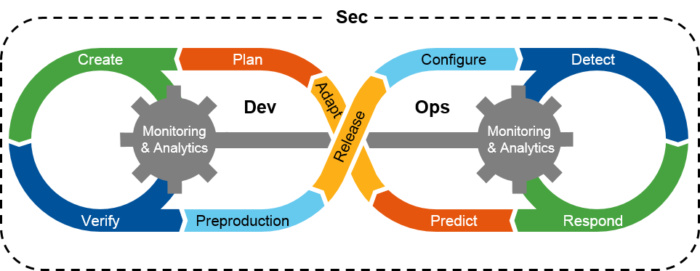
Het grote verschil van DevSecOps en DevOps is security (Sec). Bij DevSecOps hou je het streven aan om bij keuzes op security te letten, dit doe je in de kader van de gehele ontwikkeling van de organisatie en applicaties.

<https://github.com/limiw/open-source-discussions/wiki/What-is-SecOps-and-How-it-Can-Benefit-You#what-are-the-goals-of-secops>



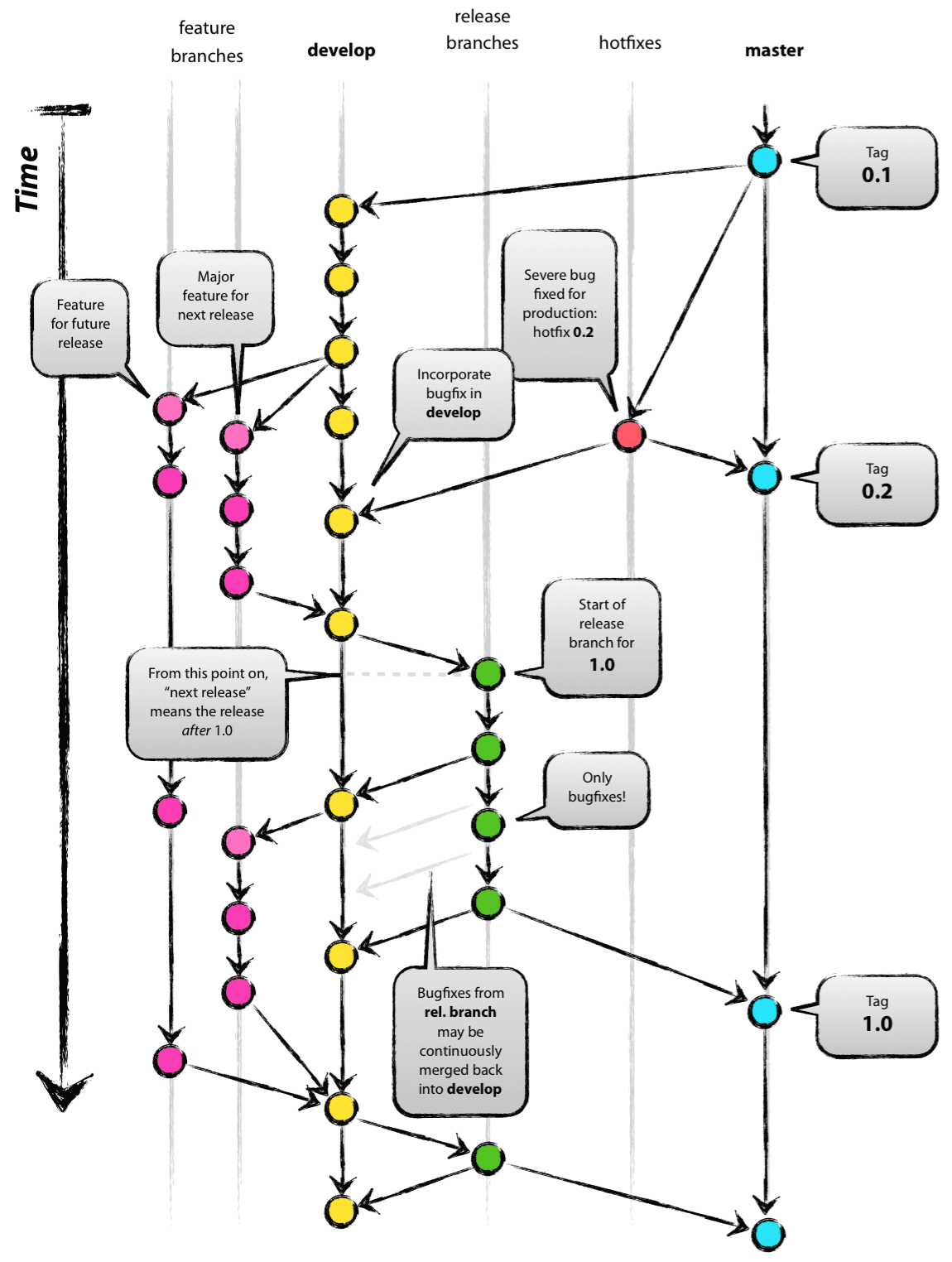
## Hoe geef je een DevSecOps setup schematisch weer?

Na onderzoek blijkt dat er geen schematische weergave is van DevSecOps, deze is hetzelfde als DevOps maar dan als omliggende kader ook nog Security.



## Wat is de beste manier om versiebeheer te implementeren?

Gebruik GitFlow als branching model. Een typische GitFlow workflow ziet eruit als volgt:



**GitFlow Best Practices**

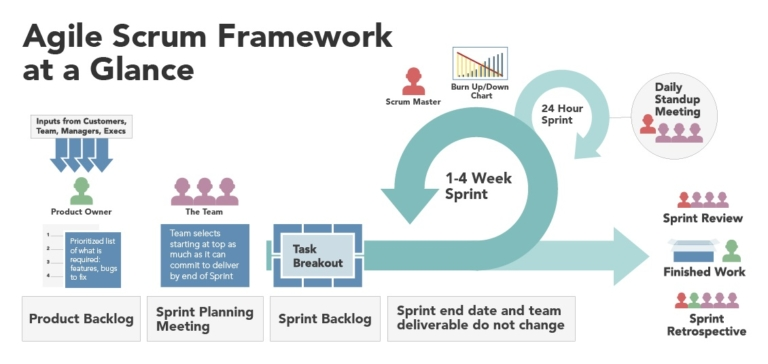
* Om merging issues te voorkomen is het verstandig om eerst een branche te rebasen op de te mergen branche zodat merging issues tot een minimum wordt gehouden.
* Gebruik feature branches voor het creëren van nieuwe features. Hierdoor wordt er voorkomen dat features elkaar in de wegzitten en dat bij het testen alleen de code die relevant is voor de feature die gemaakt wordt wordt getest. Ook wanneer een feature uiteindelijk toch niet toegevoegd wordt aan het project is het handig om deze in een feature branche te plaatsen omdat deze dan simpelweg weggegooid kan worden zonder dat het invloed heeft op andere features.
* Gebruik hotfix branches voor het oplossen van kritieke bugs. Een hotfix branche zorgt ervoor dat bugfixes snel in de master branche kunnen worden gepushed zonder dat dit invloed heeft op de development. De development branche kan deze hotfix opnemen wanneer het daar klaar voor is.
* Gebruik release branches voor het preparen van een release naar de master branche. Hier worden alle features en hotfixes samengevoegd en wordt er uitvoerig getest voordat de code gepushed wordt naar de master branche en zo ook de klant. Zodra een release gemaakt is wordt er alleen maar testing en bugfixing gedaan op deze branche.

## Wat is de beste manier om agile toe te passen in de case?

Het Agile Scrum Framework is een goede manier om te werken. Hierbij wordt de klant veel betrokken bij de ontwikkeling en het onderhoud van het product.

Hier komen de volgende aspecten bij kijken:

* Product Owner
* Product Backlog
* Sprint planning
* Sprint Backlog
* Sprints
* Daily standups
* Scrum Master
* Burn Up/Down chart
* Sprint review
* Sprint retrospective
* Poker planning



Het gebruiken van een software suite waar een scrumboard en moderne incident management tooling in zit. Deze tools kunnen veel van de problemen oplossen en zullen de agile manier van werken verbeteren.

Een van de grootste spelers op dit vlak is Atlassian. Hieronder staan een aantal van hun tools met een beschrijving hiervan. Door een suite van een bedrijf te gebruiken is het mogelijk om dit allemaal aan elkaar te koppelen.

Het platform van Atlassian biedt ook veel integratie met andere partijen.

* **Jira**: Scrumboard voor het bijhouden en plannen van stories en bugfixes.  
  Dit geeft de mogelijkheid om van de volgende features gebruik te maken die het ontwikkel en onderhoudsproces makkelijker kunnen maken.
  + Hier kunnen bepaalde velden geforceerd aanwezig zijn.
  + Time tracking van stories.
  + Labels gebruiken om verschillende soort taken aan te geven.
  + Releases aangeven.
* **Jira Service Desk**: Een service desk tool waar klanten problemen in kunnen schieten
  + Eenvoudig processen aanmaken
  + Snelle communicatie met het team dat eraan werkt
  + Klant kan status zien van bugs en het ontwikkelteam ook
* **Opsgenie**: Een incident management tool voor services
  + Automatisch notificaties wanneer services niet goed lopen
* **Statuspage**: realtime status van services voor klanten

# 7. Discussieer de bevindingen

Iedereen heeft zijn individuele studie gepresenteerd, wanneer iemand een vraag had is dit beantwoord totdat iedereen elkaars studies snapte.

# Advies voor casus

Het team gaf aan dat ze de volgende uitdagingen hebben, per uitdaging geven we advies hoe je deze het beste kan aanpakken.

*“The runtime of the test suite is large (e.g. a couple of hours) and sometimes tests fail without any known reason;”*

* Het juist inrichten van testen is heel belangrijk kijk naar 3 + 4 de vraag : **Hoe zorg je ervoor dat alle unit en integration tests van hoge kwaliteit zijn?**

*“In case of reported issues, both teams have a hard time to figure out which team is responsible for investigating and fixing the issue;”*

* Stel duidelijk af welk team voor welke taken verantwoordelijk is. En zorg dat er overzicht is van de taken die opgenomen moeten worden en welke onderdelen van het systeem dit betrekking op heeft.

*“It is not clear when and what features and bug fixes are released”*

* Agile board, waarin je aangeeft wat done is en wat in progress is. Bugs en fixes kunnen op dit board weergeven worden of op een apart board. Hiermee kan een overzicht worden weergeven van de progress van features en bugfixes.

“*Team is unable handle security threats (e.g. recent SQL injection attack could have been avoided)*”

* DevSecOps, zorg ervoor dat je tijdens het design goed let op security en dit niet alter probeert op te lossen. Doe een risico analyse en kijk welke potentiële security risks het grootste gevolg hebben en pak deze aan.

*“It takes too long for the company to resolve production issues (MTTR is too high); it is not clear when (and if) a bug was solved and in what conditions it was appearing in the first place; company already paid fines for not complying to SLA with the clients”*

* overzicht met agile boards, en agile methodieken daadwerkelijk implementeren.
* Zorg voor een overzicht met tijdsindicatie voor alle issues op een board zichtbaar voor alle deelnemers van het team. Gebruik Agile methodieken en implementeer deze om tijdsgebonden te kunnen werken.
* Deel sprints op in korte termijnen om zo gemakkelijk een planning te kunnen aanpassen om ruimte te kunnen creëren voor issues met een hoge prioriteit.

*“The client has the feeling that the current change request procedure via the service desk doesn't add any value to the project;”*

* Bespreek dit met de klant en kijk of dit proces ook daadwerkelijk iets bijdraagt en of je het wellicht efficiënter kan maken.

“*The development team members feel a lot of pressure, which reduces their focus on delivering quality software and they are starting to cut corners.”*

* Zorg dat je niet in de valkuilen valt dat bij de individuele studie in kopje: *“****Wat zijn de uitdagingen die organisaties tegenkomen op een groot schaalbaar project met betrekking tot SDLC?****”.*