



TMMi® in de DevOps wereld

#SuccessWithTMMi

EDITOR: ERIK VAN VEENENDAAL

www.tmmi.org

TMMi heeft de intentie om onafhankelijk te zijn van de software levenscyclus. In dit document wordt uitgelegd hoe het TMMi model kan worden toegepast om aan te sluiten bij een DevOps context. De TMMi level 2 procesgebieden en hun doelstellingen worden één voor één behandeld. In volgende versies van dit document zullen ook de overige TMMi level procesgebieden aan bod komen. Er wordt toegelicht hoe deze gerelateerd kunnen worden aan DevOps en hoe de toepassingen er in de regel uit komen te zien. Als verwacht wordt dat een toepassing niet wordt uitgevoerd in een DevOps context vanwege het ontbreken van toegevoegde waarde, dan zal dit expliciet benoemd worden. Toekomstige versies van dit document zullen procesgebieden op hogere TMMi levels beschrijven. Dit document is niet bedoeld als een volledige DevOps testing syllabus, maar vooral als een document dat laat zien hoe de TMMi doelstellingen het best geïnterpreteerd kunnen worden in een DevOps context en hiernaast ruimte biedt voor nieuwe ideeën, richtingen en verder onderzoek.

© TMMi Foundation 2011–25

Alle rechten voorbehouden. Niets uit deze publicatie mag worden uitgeleend, verkocht, overgedragen, gereproduceerd of overgedragen, geheel of gedeeltelijk, in welke vorm of op welke wijze dan ook, zonder voorafgaande toestemming van de TMMi Foundation, behalve op de wijze zoals beschreven in de bijbehorende licentie documentatie. Waar enige vorm van kopiëren is toegestaan onder de voorwaarden van de bijbehorende licentie documentatie, is dit onderworpen aan de voorwaarde dat deze kennisgeving in dergelijke kopieën wordt gereproduceerd.

Woorden waarvan wij reden hebben om aan te nemen dat ze handelsmerken vormen, zijn als zodanig aangemerkt. De aanwezigheid of afwezigheid van een dergelijke aanduiding mag echter niet worden beschouwd als een aantasting van de juridische status van een handelsmerk.

TMMi een geregistreerd handelsmerk van de TMMi Foundation (UK).

©2025 TMMi Foundation

Bijdragers

Katalin Balla (Hongarije)

Suresh Chandra Bose (VS)

Ester Greenwood (Nederland – vertaling)

Kari Kakkonen (Finland)

Willem Keesman (Nederland – vertaling)

Mattijs Kemmink (Nederland)

Rik Marselis (Nederland)

Marcel Mersie (Nederland – vertaling)

Tim Schijffelen (Nederland – vertaling)

Gilbert Smulders (Nederland – vertaling)

Szilard Szell (Hongarije)

Erik van Veenendaal (Nederland)

Revisies

Dit onderdeel geeft de revisies en wijzigingen weer tussen de verschillende versies van dit document
Dit onderdeel is toegevoegd ter informatie.

Release	Revisie Notities
V1.0	Initiële vertaling van het “TMMi in the DevOps world” document

Table of Contents

Bijdragers	2
Revisies	3
Table of Contents.....	4
1 TMMi® in de DevOps wereld.....	5
1.1 Introductie	5
1.2 DevOps en DevSecOps.....	5
1.3 Test Maturity Model integration (TMMi)	6
1.4 Testen binnen een DevOps context.....	7
1.4.1 Uitdaging bij het testen in een DevOps context.....	7
1.4.2 TMMi en DevOps	8
1.4.3 Relatie met de ISTQB DevOps syllabus.....	10
2 TMMi Level 2 Beheerst.....	11
2.1 Procesgebied 2.1 Testbeleid en Strategie	11
2.1.1 SG1 Opstellen Testbeleid	11
2.1.2 SG2 Opstellen Teststrategie	13
2.1.3 SG3 Opstellen Testprestatie-indicatoren.....	15
2.2 Process Area 2.2 Test Planning.....	16
2.2.1 SG1 Uitvoeren Productrisicoanalyse	16
2.2.2 SG2 Opstellen Testaanpak	17
2.2.3 SG3 Opstellen Testbegroting	19
2.2.4 SG4 Opstellen Testplan.....	20
2.2.5 SG5 Verkrijgen Testplan Commitment.....	21
2.3 Procesgebied 2.3 Testbewaking en -beheersing	21
2.3.1 SG1 Bewaken testvoortgang t.o.v. het testplan	22
2.3.2 SG2 Bewaken productkwaliteit t.o.v. het testplan en verwachtingen	23
2.3.3 SG3 Managen correctieve acties	24
2.4 Procesgebied 2.4 Testontwerp en -uitvoering.....	25
2.4.1 SG1 Uitvoeren testanalyse en -ontwerp waarbij gebruik wordt gemaakt van testontwerptechnieken.....	25
2.4.2 SG2 Uitvoeren testimplementation.....	26
2.4.3 SG3 Uitvoeren testen	27
2.4.4 SG4 Managen testbevindingen.....	28
2.5 Procesgebied 2.5 Testomgeving	29
2.5.1 SG1 Opstellen testomgevingseisen	29
2.5.2 SG2 Implementeren testomgeving	30
Bronnen	32

1 TMMi® in de DevOps wereld

1.1 Introductie

Dit document beschrijft hoe het combineren van een DevOps aanpak voor softwareontwikkeling en het TMMi model een mogelijke route is om bedrijfsdoelstellingen te behalen. Het legt uit hoe TMMi ten voordele kan worden toegepast in een DevOps context. In dit document worden bewezen DevOps-gebaseerde mogelijkheden gedeeld, die een alternatief bieden voor TMMi toepassingen. In dit document wordt beschreven hoe TMMi DevOps organisaties kan ondersteunen, ongeacht hoe volwassen deze zijn op het gebied van DevOps werken, door het aanstippen van kritieke test toepassingen welke vaak zichtbaarheid verliezen zodra organisaties groeien en de druk vanuit projecten toeneemt. Elk TMMi procesgebied en de bijbehorende specifieke doelstellingen zullen één voor één beschreven worden. Er wordt toegelicht hoe deze gerelateerd zijn aan DevOps en hoe de bijbehorende toepassingen er in de regel uit kunnen zien. Als er een verwachting is dat een toepassing niet zal worden uitgevoerd in een DevOps context omdat deze geen waarde toevoegt, dan zal dit expliciet benoemd worden.

In de dynamische DevOps wereld die wij kennen, waar Continuous Integration (CI), Continuous Deployment (CD) en automatisering de ontwikkeling van software sturen, wordt het waarborgen van kwaliteit in elke stap doorslaggevend voor succes. Het test volwassenheidsmodel TMMi biedt een gestructureerd framework voor het vaststellen en verbeteren van test toepassingen. DevOps teams streven naar snelle releases zonder dat daarbij wordt ingeboet op kwaliteit. Het integreren van TMMi biedt een routekaart om testvolwassenheid te verbeteren, de aansluiting te vinden bij bedrijfsdoelstellingen en resultaten op te leveren. Deze synergie tussen TMMi en DevOps zorgt voor een gebalanceerde aanpak om ontwikkeling te versnellen en hierbij hoge standaarden te behouden op het gebied van kwaliteit en betrouwbaarheid.

Dit document richt zich op verschillende doelgroepen, waarvan de belangrijkste groepen zijn:

- Volwassen, traditionele organisaties die willen overstappen naar DevOps en tegelijkertijd hun procesvolwassenheid willen behouden.
- Minder volwassen organisaties die TMMi willen gaan toepassen om hun testvolwassenheid te verhogen en parallel daaraan naar DevOps willen overstappen.
- Agile organisaties met een gestructureerd Agile testproces en die nu willen overstappen naar DevOps.
- DevOps-organisaties die succesvol zijn en groeien. Daarom hebben ze een zekere procesvolwassenheid nodig, maar willen ze tegelijkertijd de voordelen van Agile blijven behouden.

Dit document is niet bedoeld om los van het oorspronkelijke TMMi-model te worden gebruikt. Het is bedoeld om degenen die testprocesverbeteringen in een DevOps-omgeving uitvoeren, te begeleiden bij de interpretatie en betekenis van de verschillende TMMi doelstellingen en toepassingen. Dit document is een aanvulling op het oorspronkelijke TMMi-model en dient als een aanvullend document te worden gebruikt.

1.2 DevOps en DevSecOps

DevOps is een filosofie en een reeks werkwijzen die gericht zijn op het overbruggen van de kloof tussen softwareontwikkelings- en IT-operationele teams. Door een cultuur van samenwerking en gedeelde verantwoordelijkheid te bevorderen, streeft DevOps ernaar de ontwikkeling, implementatie en het onderhoud van software te stroomlijnen. De kerngedachte is het doorbreken van de traditionele silo's tussen ontwikkeling en operations, waardoor software efficiënter en effectiever kan worden geleverd. DevOps legt in de kern de nadruk op automatisering en continue processen. Automatisering speelt een cruciale rol bij het minimaliseren

van handmatige taken, waardoor fouten worden verminderd en de oplever snelheid wordt verhoogd met als doel het vaker leveren van waarde.

Dit omvat het automatiseren van de integratie van codewijzigingen, test- en implementatieprocessen door middel van Continuous Integration en Continuous Deployment. Een andere sterke nadruk ligt op de noodzakelijke verandering naar de DevOps-cultuur van autonomie, leiderschap, samenwerking, frequent experimenteren, kortere ontwikkelcycli, hogere implementatiefrequentie en betrouwbaardere releases, in nauwe afstemming op bedrijfsdoelstellingen en klanten.

De DevOps-aanpak biedt tal van voordelen, waaronder een snellere time-to-market, verbeterde samenwerking tussen teams, hogere softwarekwaliteit en -betrouwbaarheid, en verhoogde schaalbaarheid en flexibiliteit. Door snellere detectie en oplossing van problemen mogelijk te maken, vermindert DevOps ook downtime en verbetert het de veerkracht van het systeem. Uiteindelijk draait DevOps net zo goed om culturele verandering als om technologie en processen. Het vereist een mentaliteitsverandering waarbij zowel ontwikkel- als operationele teams werken aan gemeenschappelijke doelen, feedback omarmen en continu streven naar verbetering. Door deze elementen te integreren, helpt DevOps organisaties om software efficiënter, betrouwbaarder en veiliger te leveren.

DevSecOps

DevSecOps, een afkorting voor Development, Security en Operations, is een aanpak die security integreert in het DevOps-proces. Deze methodologie is erop gericht ervoor te zorgen dat security een gedeelde verantwoordelijkheid is gedurende de gehele IT-levenscyclus, van initiële ontwikkeling tot implementatie en daarna. DevSecOps streeft ernaar om veilige software sneller en efficiënter te leveren en zo het risico op beveiligingsinbreuken te verkleinen.

1.3 Test Maturity Model integration (TMMi)

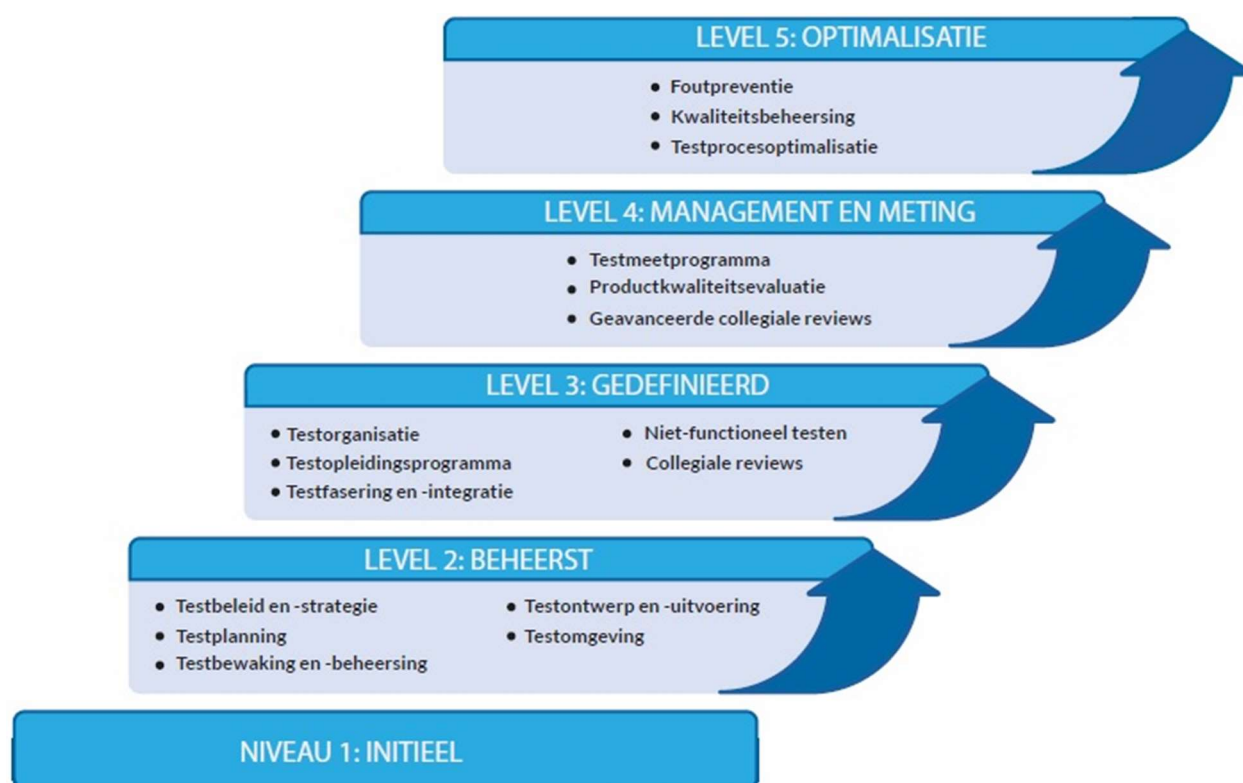
Het TMMi-framework is ontwikkeld door de TMMi Foundation als leidraad en referentiekader voor het verbeteren van testprocessen binnen een organisatie. Het richt zich op kwesties die belangrijk zijn voor testmanagers, testengineers, ontwikkelaars en professionals op het gebied van softwarekwaliteit. Testen wordt binnen TMMi in de breedste zin van het woord gedefinieerd en omvat alle activiteiten die verband houden met de kwaliteit van softwareproducten.

TMMi gebruikt het concept van volwassenheidsniveaus voor procesevaluatie en -verbetering. Daarnaast worden procesgebieden, doelstellingen en toepassingen geïdentificeerd. Het toepassen van de TMMi-volwassenheidscriteria verbetert het testproces en heeft aantoonbaar een positieve impact gehad op de productkwaliteit, de productiviteit van test engineers en de doorlooptijd. TMMi is ontwikkeld om organisaties te ondersteunen bij het evalueren en verbeteren van hun testprocessen.

TMMi heeft een gefaseerde architectuur voor procesverbetering. Deze bevat niveaus of levels die een organisatie doorloopt naarmate het testproces evolueert van een ad-hoc en onbeheerd proces naar een proces dat beheerd, gedefinieerd, gemeten en geoptimaliseerd wordt. Het behalen van elk level betekent dat alle doelstellingen welke in scope zijn, behaald zijn en dat de toegepaste verbeteringen de basis vormen voor het volgende level. De interne structuur van TMMi bevat test toepassingen die systematisch kunnen worden aangeleerd en toegepast ter ondersteuning van een kwalitatief hoogstaand testproces dat stapsgewijs verbeterd. TMMi kent vijf niveaus die de volwassenheidshiërarchie en het pad naar testprocesverbetering voorschrijven. Elk niveau heeft een set procesgebieden die een organisatie moet implementeren om

volwassenheid op dat niveau te bereiken. De procesgebieden voor elk volwassenheidsniveau van TMMi worden weergegeven in Afbeelding 1.

Een belangrijk onderliggend principe van TMMi is dat het een generiek model is dat toepasbaar is op verschillende levenscyclusmodellen en -omgevingen. De meeste doelstellingen en toepassingen zoals gedefinieerd door TMMi zijn toepasbaar op zowel sequentiële als iteratieve levenscyclusmodellen, waaronder Agile. Op het laagste niveau van het model zijn veel van de subtoepassingen en (tussen)producten echter (zeer) verschillend, afhankelijk van het toegepaste levenscyclusmodel. Binnen TMMi zijn alleen de doelstellingen verplicht om te behalen, de toepassingen niet. TMMi is gratis beschikbaar op de website van de TMMi Foundation en ook verkrijgbaar als boek. Het model is naar veel talen vertaald, waaronder Spaans, Frans en Chinees.



Afbeelding 1: TMMi volwassenheidsniveaus en procesgebieden

1.4 Testen binnen een DevOps context

1.4.1 Uitdaging bij het testen in een DevOps context

Testen in een DevOps-context brengt unieke uitdagingen met zich mee vanwege de aard van CI/CD-pipelines, de samenwerking tussen ontwikkel- en operationele teams en het snelle tempo van ontwikkelcycli. Enkele van de belangrijkste test uitdagingen zijn:

- *Continu testen:* Tests moeten snel en frequent worden uitgevoerd om geïntegreerd te kunnen worden binnen CI/CD-pipelines. Er is een grote afhankelijkheid van geautomatiseerd testen, wat robuuste en betrouwbare testscripts en frameworks vereist.
- *Omgevingsbeheer:* Het waarborgen van goed beheer van de testomgeving en het inrichten van realistische testomgevingen voor continu testen kan ingewikkeld zijn. Er is vaak een gebrek aan pariteit tussen productie- en testomgevingen, wat kan leiden tot onopgemerkte problemen. Een andere uitdaging is het

beheren en onderhouden van consistente configuraties in meerdere omgevingen. Omgevingen moeten automatisch worden geïmplementeerd en geconfigureerd, meestal on-demand, en na het testen worden verwijderd.

- *Test Databeheer:* Hoewel testgegevens ook productie-like, consistent en on-demand beschikbaar moeten zijn, is naleving van privacyrichtlijnen eveneens een must.
- *Integratie- en regressietesten:* Het integreren van meerdere componenten en services met complexe onderlinge afhankelijkheden, waardoor het testen ervan een uitdaging wordt. Ook moet ervoor worden gezorgd dat nieuwe wijzigingen de bestaande functionaliteit niet verstoren, wat uitgebreide regressietest-suites vereist.
- *Tools en infrastructuur:* Het integreren van verschillende testtools met CI/CD-pipelines en andere DevOps-tools en ervoor zorgen dat de testinfrastructuur schaalbaar is om de toegenomen belasting en complexiteit aan te kunnen.
- *Culturele en organisatorische uitdagingen:* Het bevorderen van samenwerking tussen ontwikkelaars, testers en operations, die traditioneel regelmatig in silo's opereerden met vaak tegenstrijdige belangen. Ervoor zorgen dat teamleden over de nodige vaardigheden beschikken om te werken met geautomatiseerde tests, continue integratie- en deployment activiteiten..
- *Performance testen:* Het simuleren van realistische belastingen en omstandigheden in een gecontroleerde testomgeving, maar ook het integreren van performancetesten in de CI/CD-pipeline om de prestaties continu te bewaken en te optimaliseren.
- *Security testen:* Bij snel veranderende pipelines worden security testen vaak over het hoofd gezien. Het integreren van security tests (DevSecOps) zonder de ontwikkeling te vertragen en ervoor te zorgen dat kwetsbaarheden vroegtijdig worden ontdekt, vereist gespecialiseerde tools en strategieën en is een uitdaging voor veel teams. Beveiligingstests moeten bij voorkeur vroeg in de ontwikkelingscyclus worden geïntegreerd met behulp van geautomatiseerde tools om statische en dynamische security tests uit te voeren.
- *Feedback en rapportage:* Snelle, duidelijke en bruikbare feedback aan het team geven op basis van testresultaten. Uitgebreide rapporten genereren die inzicht geven in de kwaliteit van de build, maar het proces toch agile houden.
- *Shift-left:* Verbeteren van kwaliteit door taken zo vroeg mogelijk in de levenscyclus naar links te verplaatsen. Shift-left testen betekent eerder in het proces testen, bijvoorbeeld door middel van (code)reviews, statische analyses en grondige unittesten.
- *Shift-right:* Omdat het zo vaak mogelijk implementeren van wijzigingen in productie essentieel is, moet een teststrategie een balans vinden tussen testactiviteiten in het CI-gedeelte en het monitoren en observeren van het systeemgebruik in productie in CD.
- *Monitoring en feedback loops:* In DevOps stopt testen niet bij de implementatie; continue monitoring en het verzamelen van feedback uit productieomgevingen zijn cruciaal. Het instellen van effectieve controletools en het interpreteren van productiegegevens om defecten op te sporen en de prestaties te optimaliseren, wordt vaak over het hoofd gezien of is onvoldoende ontwikkeld.

1.4.2 TMMi en DevOps

De misvatting bestaat dat TMMi en DevOps-benaderingen haaks op elkaar staan. DevOps-benaderingen en TMMi kunnen niet alleen naast elkaar bestaan, maar zullen, wanneer ze succesvol geïntegreerd zijn, aanzienlijke voordelen opleveren. Er is ook een uitdaging om anders naar testen te kijken, volledig geïntegreerd te zijn in de ontwikkeling en een CI/CD-pipeline te hebben, en wat dit betekent in de context van een 'test'-verbeterprogramma. Merk op dat de 'i' van TMMi verwijst naar het feit dat testen een geïntegreerd onderdeel van softwareontwikkeling moet zijn en niet als iets volledig losstaands moet worden behandeld.

Het TMMi-model schenkt aandacht aan kritieke test werkwijzen, zoals productrisicoanalyse, welke vaak 'vergeten' worden in een DevOps-context. Dit document laat met voorbeelden zien dat TMMi en DevOps-methoden zeer effectief kunnen samenwerken.

Bij de implementatie van TMMi moet men er rekening mee houden dat het TMMi-model niet bedoeld is om een reeks werkwijzen aan een organisatie op te leggen, noch om te worden toegepast als een standaard waaraan men moet "bewijzen dat men eraan voldoet". TMMi kan, indien correct gebruikt, helpen bij het vinden van de specifieke testgebieden waar verandering waarde kan opleveren, aansluitend bij de bedrijfsdoelstellingen. Dit geldt ongeacht het toegepaste levenscyclusmodel voor softwareontwikkeling. Het is belangrijk om te benoemen dat TMMi-toepassingen een verwacht onderdeel zijn, maar ook kunnen worden bereikt door wat een "alternatieve" toepassing wordt genoemd ten opzichte van een gedefinieerde TMMi toepassing. Belangrijk is hierbij om stil te staan bij de bedoeling van de werkwijze, wat de rationale is en hoe deze waarde toevoegt aan de organisatie. Vaak wordt de doelstelling in een DevOps-omgeving al bereikt, maar dan via een alternatieve werkwijze. Doorgaans is "elke" oplossing compliant zolang deze maar aansluit bij de bedrijfsbehoeften! Wees niet te voorschrijvend bij het gebruik van TMMi, zo is TMMi niet bedoeld. Interpreteer de TMMi doelstellingen en toepassingen altijd binnen de context van de situatie. Door eerst de proces behoeften binnen de specifieke bedrijfscontext vast te stellen kun je over het algemeen beslissingen nemen hoe je focus krijgt en stuurt op prioriteiten in procesverbetering.

De meeste conflicten tussen TMMi en DevOps zijn gebaseerd op een gedateerde TMMi-visie op hoe een "goede toepassing" eruit zou moeten zien na implementatie, of op een misvatting over de rationale achter DevOps-toepassingen, gebaseerd op hoe deze de DevOps-werkwijze en -waarden zouden moeten ondersteunen. TMMi-experts, waaronder (lead) assessoren, zullen berichten die mogelijk onbedoeld worden gedeeld met betrekking tot hoe een "goede TMMi-conforme" toepassing eruit zou moeten zien na implementatie, opnieuw moeten evalueren en mogelijk heroverwegen. Wanneer DevOps-benaderingen op de juiste manier worden geïmplementeerd, samen met TMMi-processen, zal dit resulteren in de effectieve implementatie van test werkwijzen, niet in hun afschaffing. Merk op dat er naast de specifieke (test)toepassingen binnen TMMi ook generieke toepassingen zijn. Het doel van de generieke toepassingen is om de institutionalisering van een procesgebied te ondersteunen, wat in feite betekent dat de organisatie over een infrastructuur beschikt om het procesgebied te ondersteunen wanneer nieuwe mensen starten bij de organisatie of er andere veranderingen binnen de organisatie plaatsvinden.

De overstap van traditionele softwareontwikkeling naar DevOps kan ook het initiatief doen ontstaan om de processen zoals ze nu gedefinieerd zijn te snoeien en LEAN te maken. Zo profiteren TMMi-gecertificeerde organisaties van de Agile en lean-denkwijze die beide binnen DevOps aanwezig zijn. Er is een tendens geweest om elementen binnen het TMMi-model verkeerd te interpreteren, waardoor onnodige processen en werkproducten zonder toegevoegde waarde ontstonden. Door terug te gaan naar de basis en verbeterdoelen en het TMMi-model te gebruiken zoals het bedoeld is, ondersteunt men de afstemming van echte processen met toegevoegde waarde op echte procesbehoeften en -doelstellingen. Het snoeien en lean maken van processen met een Agile/Lean-mentaliteit resulteert in processen die weerspiegelen wat mensen echt doen en zorgt ervoor dat alleen de data wordt verzameld die daadwerkelijk wordt gebruikt. Deze mentaliteit brengt ook een focus met zich mee om zaken zo eenvoudig mogelijk te houden, wat doorgaans niet eenvoudig is, maar wel voordelen oplevert voor degenen die een TMMi-implementatie uitvoeren.

Verbeteringen binnen DevOps, gebaseerd op de DORA-metrieken van doorlooptijd van wijzigingen, implementatie frequentie, percentage mislukte wijzigingen en hersteltijd van mislukte implementaties [DORA16], [DORA24], vinden doorgaans plaats via kleine en autonome teams die snel actie kunnen ondernemen. Dit is een andere manier waarop TMMi kan profiteren van de toepassing van DevOps. Hoewel

©2025 TMMi Foundation

TMMi uitgebreid is, kunnen organisaties niet elk TMMi-element volledig implementeren. Om succesvol te zijn, moeten organisaties hun eigen belangrijkste test werkwijzen en verbeteringen identificeren die prioriteit en focus vereisen.

1.4.3 Relatie met de ISTQB DevOps syllabus

TMMi en de ISTQB zijn een alliantie aangegaan om het vakgebied softwaretesten verder te promoten. Ook tijdens de ontwikkeling van dit document hebben de technische commissie van TMMi en de werkgroep van de ISTQB-syllabus "Quality in DevOps" samengewerkt. TMMi is een testverbeteringsmodel en biedt als zodanig een vooraf gedefinieerde verbeteraanpak met prioriteiten gebaseerd op de structuur van het TMMi-model. In dit document ligt de nadruk op het geven van een interpretatie van de TMMi-verbeteringsdoelen voor degenen die in een DevOps-context werken. Het biedt geen gedetailleerde, volledige beschrijving van goede DevOps engineering werkwijzen. De ISTQB-syllabus "Quality in DevOps" is een inhoudelijk document en beoogt als zodanig een gedetailleerde beschrijving te geven van goede DevOps engineering werkwijzen, bijvoorbeeld kwaliteitsengineering (inclusief testen). De twee documenten vullen elkaar in essentie uitstekend aan: TMMi geeft aan wat er verbeterd moet worden (welke processen), het ISTQB DevOps-syllabus beschrijft hoe dingen gedaan moeten worden (engineering best practices).

2 TMMi Level 2 Beheerst

2.1 Procesgebied 2.1 Testbeleid en Strategie

Het doel van het procesgebied Testbeleid en -strategie is het ontwikkelen en vaststellen van een testbeleid en een organisatie- of programmabrede teststrategie. In de teststrategie worden testsoorten eenduidig gedefinieerd. Om de kwaliteit van het testproces te meten worden kwaliteitsindicatoren geïntroduceerd.

2.1.1 SG1 Opstellen Testbeleid

Elke organisatie die een testverbeteringsproject start, zou moeten starten met het definiëren van een testbeleid. Het testbeleid definieert de algemene test doelstellingen, doelen en strategische visies van de organisatie met betrekking tot testen en testprofessionals. Het is belangrijk dat het testbeleid is afgestemd op het algemene bedrijfs- en kwaliteitsbeleid van de organisatie. Testverbeteringen moeten worden gedreven door duidelijke bedrijfsdoelen, die op hun beurt moeten worden vastgelegd in het test(verbeterings)beleid. Een testbeleid is noodzakelijk om een gemeenschappelijke visie op testen en de bijbehorende doelstellingen te bereiken tussen alle stakeholders binnen een organisatie. Deze gemeenschappelijke visie is nodig om test(procesverbeterings)activiteiten binnen de hele organisatie op elkaar af te stemmen. Merk op dat testdoelstellingen nooit op zichzelf staande doelstellingen mogen zijn; ze zijn afgeleid van het hogere doel om werkende software en productkwaliteit te realiseren.

Het bovenstaande geldt ook voor een organisatie die de DevOps-cultuur heeft omarmd en DevOps softwareontwikkeling toepast. Sterker nog, binnen veel organisaties wordt er gesproken over de veranderende rol van testen, de onafhankelijkheid van testen, testautomatisering en de rol van professionele testers in DevOps-softwareontwikkeling. Deze en andere punten zijn doorgaans kwesties die in een gesprek met het management en andere stakeholders aan bod moeten komen en vastgelegd moeten worden in een testbeleid. Elke organisatie, inclusief organisaties die DevOps toepassen, die een testverbeteringsproject wil starten, moet de business KPI's en behoeften voor een dergelijk initiatief identificeren en definiëren. Want zonder deze is er geen basis om een verbeteringstraject te starten. Door tijd te besteden aan het vastleggen van de werkelijke businessbehoeften, kan men een context creëren om te beslissen waar de (test)verbeteringsprioriteiten op gericht moeten zijn, bijvoorbeeld op welk procesgebied. Houd er rekening mee dat een testbeleid doorgaans een lean document van één pagina, een webpagina of een wandkaart is op organisatieniveau, en geen document op projectniveau.

De specifieke doelstelling van TMMi, 'Een testbeleid opstellen', inclusief de bijbehorende specifieke toepassingen, is volledig van toepassing op organisaties die DevOps softwareontwikkeling toepassen. Binnen DevOps vallen ontwikkeling, testen en operations echter meestal onder hetzelfde proces, of zijn ze in ieder geval nauw op elkaar afgestemd. Daarom wordt van een testbeleid ook verwacht dat het een integraal onderdeel is van een overkoepelend DevOps-beleid, met meer aandacht voor quality engineering dan alleen testen. Testen binnen DevOps is georganiseerd met een evenwichtige verdeling van shift-left en shift-right benaderingen, waarbij gebruik wordt gemaakt van samenwerkingsmethoden en Test First-denken. Binnen DevOps moeten testactiviteiten alle drie de benaderingen (zoals gedefinieerd in het DevOps-handboek) ondersteunen: de flow, de feedback en continu experimenteren en leren.

Zoals aangegeven, zullen de testdoelen zich richten op een duidelijke balans tussen shift-left en shift-right; proactief, design- en testgericht denken om de benodigde kwaliteits- en testactiviteiten op te zetten vóór de implementatie (binnen en rond de CI/CD-straat) om defects te vinden, en reactieve maatregelen om de

downtime van de service in productie te verminderen in geval van een fout die door de mazen van het net glipt, bijvoorbeeld een automatisch rollback-mechanisme om downtime te verminderen, blue-green deployment¹ of canary release om de impact te beperken. Hoewel het doel van testactiviteiten is om de kans op implementatie van een fout (defecte software) in productie te verkleinen, worden implementatiestrategieën toegepast om de impact van een bepaalde fout te verminderen, wat resulteert in een betere algehele kwaliteit. Om de impact te verminderen, zullen de testactiviteiten van de implementatie pipeline worden gestroomlijnd om de doorstroming en doorlooptijden te verbeteren, terwijl de algehele risiconiveaus laag blijven.

Naast testen tijdens de softwareontwikkeling is shift-right (testen in productie) ook een gangbare werkwijze binnen DevOps. Dit heeft ook invloed op de definitie van de testdoelen zoals gedefinieerd in het beleid. De focus ligt hierbij op snelle reacties op problemen in productie, gebaseerd op telemetrie-informatie, om downtime te verminderen, de beschikbaarheid te verbeteren en zo de kwaliteit te verbeteren. Site Reliability Engineering (SRE) omvat het monitoren van de gezondheid van systemen, services en applicaties en het creëren van closed-loop automatisering mechanismen om te reageren op systeemgezondheid/afwijkingen [Beyer]. Dit wordt ook wel zelfhelend genoemd.

Vanwege het aantal verwachte veranderingen in DevOps, richten de testdoelen zich sterk op regressietesten, zowel op functionele als niet-functionele aspecten. Sommige niet-functionele aspecten kunnen in productie (shift-right) na implementatie worden gemeten, terwijl andere moeten worden getest voordat ze in productie gaan, bijvoorbeeld beveiliging.

Voorbeelden van testdoelen:

- Testen is gericht op het ervoor zorgdragen dat wijzigingen de verwachte waarde opleveren (acceptatietest) en dat ze geen bestaande functionaliteit verstoren (regressietest).
- Testen moet de ontwikkelingsdoelen van de organisatie ondersteunen, zoals het verminderen van het aantal mislukte wijzigingen of het verhogen van de implementatie frequentie.

Ook onafhankelijkheid en de definitie van het testproces op hoog niveau per testbeleid worden doorgaans beïnvloed in de context van DevOps. Wat betreft onafhankelijkheid, is testen een rol binnen de ontwikkelorganisatie, vaak aangeduid als DevTest om een dubbele rol te vertegenwoordigen. De onafhankelijkheid is daarom meestal laag. Organisaties vertrouwen meestal op een zogenaamd platform team met Software Development Engineers in Test (SDET), wiens belangrijkste rol het ontwikkelen en onderhouden van testbaarheidsaspecten van het product en de door DevTest gebruikte test automatiseringsoplossingen is.

Daarnaast kunnen speciale test teams worden aangesteld voor complexere gebieden, zoals niet-functioneel testen of complexe end-to-end scenario's. In dit geval is de onafhankelijkheid hoger, maar is de samenwerking nog steeds nauw. Wat betreft de definitie van het testproces op hoog niveau, is testen nu een integraal onderdeel van het ontwikkelproces, of de CI/CD-strategie, en dus ook van het testproces. Binnen een omgeving waar meerdere DevOps teams opereren zijn er vaak gespecialiseerde teams voor de niet-functionele testen en het onderhoud van generieke automatiseringsoplossingen.

Het is belangrijk ervoor te zorgen dat het testbeleid, meestal een lichtgewicht/lean document, wordt verspreid over de hele organisatie/waardeketen/Agile release train/tribe. Omdat testen een cruciaal onderdeel is van

¹ Een blue-green deployment is een deployment strategie waarbij twee identieke maar gescheiden omgevingen worden gecreëerd. Eén omgeving (blue) draait de huidige applicatie-versie en één omgeving draait de nieuwe applicatie-versie.

het werk van Agile- en DevOps-teams en geen aparte organisatie is, moet het beleid bekend zijn bij iedereen die betrokken is en er invloed op heeft.

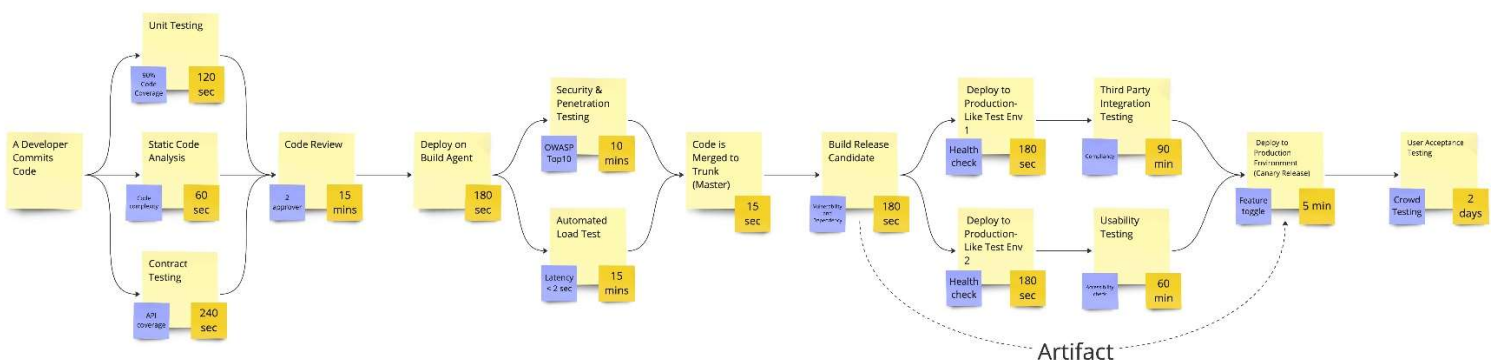
2.1.2 SG2 Opstellen Teststrategie

De teststrategie volgt het testbeleid en dient als uitgangspunt voor de testactiviteiten binnen projecten. Een teststrategie wordt doorgaans organisatie- of programmabreed gedefinieerd. Een typische teststrategie is gebaseerd op een productrisicobeoordeling op hoog niveau en bevat een beschrijving van de uit te voeren testtypen en testsoorten. Het is niet voldoende om bijvoorbeeld alleen te stellen dat binnen elk project integratietesten en acceptatietesten worden uitgevoerd. We moeten definiëren wat wordt bedoeld met unit- en acceptatietesten; hoe deze doorgaans worden uitgevoerd en wat hun belangrijkste doelstellingen zijn. De ervaring leert dat wanneer een teststrategie is gedefinieerd en gevolgd, er minder overlap tussen de verschillende testactiviteiten zal optreden, wat leidt tot een efficiënter testproces. Bovendien zullen er, doordat de testdoelstellingen en -aanpak van de verschillende testtypen en -soorten op elkaar zijn afgestemd, minder gaten overblijven, wat leidt tot een effectiever testproces en daarmee een hogere productkwaliteit.

Een teststrategie is een essentieel document binnen een DevOps-omgeving. Het definieert op hoofdniveau de tests die door de teams moeten worden uitgevoerd, voornamelijk binnen de CI/CD-strategie; welke testsoorten en testtypen worden uitgevoerd en op hoofdniveau hun aanpak. Voor elke testsoort en testtype worden de doelstellingen, verantwoordelijkheden, belangrijkste continue test taken en acceptatiecriteria en quality gates gedefinieerd. De teststrategie volgt doorgaans de Agile testkwadranten (zie afbeelding 2), maar is opgebouwd rond de CI/CD-pipeline om de stroom van activiteiten binnen de pipeline weer te geven. De pipeline moet niet alleen worden begrepen als de activiteiten die door een CI/CD-tool worden uitgevoerd, maar als alles wat de teams doen wanneer ze aan een backlogitem werken. Testactiviteiten zoals reviews en exploratory testen vinden plaats buiten de CI/CD-tools, maar als een integraal onderdeel van de stroom/pipeline/waardeketen. Let op: de pipeline kan niet alleen als technologie/tool worden gebruikt, maar ook als visuele representatie van de activiteiten, het proces en de mensen [zie afbeelding 3].



Afbeelding 2: De Agile testkwadranten



Afbeelding 3: Een teststrategie weergegeven als een pipeline

Een teststrategie is een essentieel document, ook bij het toepassen van een DevOps-werkwijze. DevOps is relatief nieuw voor veel organisaties. Een gemeenschappelijke, organisatie-brede teststrategie biedt daarom houvast voor alle betrokkenen en legt uit waar en hoe testen in de pipeline worden uitgevoerd. Het voorkomt ook dat elk project/pipeline het wiel opnieuw uitvindt.

De teststrategie moet uiteraard volledig aansluiten op de ontwikkelstrategie, aangezien testen een geïntegreerd onderdeel is van de ontwikkel pipeline. Enkele overwegingen (geordend per onderwerp) waarmee rekening moet worden gehouden bij het ontwikkelen van een DevOps-teststrategie:

Testsoorten

- Testsoorten worden voornamelijk weergegeven als testactiviteiten/taken in de CI/CD-pipeline.
- De teststrategie is vaak opgebouwd rond acceptatietesten van de ontwikkelde user stories, aangestuurd door Behavior Driven Development (BDD) toepassingen. Dit vereist continu testen in elke fase van de pipeline.
- De teststrategie definieert doorgaans een testsoort, zoals contract based testing, om de risico's van het gebruik van open API's te beperken en integratietesten te ondersteunen.

Low level testen

- Low-level ontwikkel- en test toepassingen zoals pull-requests, merges en code review moeten in overweging worden genomen.
- Statische testen en andere low-level testactiviteiten worden meestal verplicht gesteld door de CI/CD-tool en kunnen vaak niet eens worden vermeden (automatische controles en tests als onderdeel van de pull-request bijvoorbeeld).
- Gebaseerd op het shift-left paradigma wordt ook de nadruk gelegd op unit testing, het bereiken van een overeengekomen niveau van code dekking en integratietests.

Testautomatisering

- Continu testen is naadloos geïntegreerd in de softwareleveringspipeline en de DevOps-toolchain.
- Een testautomatiseringsstrategie is een cruciaal onderdeel van de DevOps-teststrategie.

Entry- and Exit criteria

- Entrycriteria worden weergegeven als Definition-of-Ready (DoR) voor de DevOps-teams. DoR vereist een hoogwaardige testbasis die als input dient voor testen, bijvoorbeeld acceptatiecriteria die gezamenlijk door het team worden gedefinieerd, meestal in de vorm van een Given-When-Then-statement. Het is belangrijk dat risico's en testdoelen goed worden gedefinieerd en overeengekomen voordat de Ready-status wordt bereikt, zodat de schatting ook de testbehoeften omvat.
- Exit criteria worden opgesplitst in de Definition-of-Done (DoD) voor DevOps-teams en de Quality Gate die is ingebouwd in elke testtaak in de CI/CD-pipeline. De DoD kan op verschillende niveaus (epic, feature, story) worden gedefinieerd en de voltooiing van de test en het bereikte vertrouwensniveau op verschillende niveaus/fasen vertegenwoordigen. Naast succesvolle acceptatietests omvat DoD doorgaans verwachte testactiviteiten om te voldoen aan niet-functionele vereisten, wat een beperking vormt voor alle backlogitems. In DevOps-teams wordt de status 'GEREED' pas bereikt wanneer het backlogitem al in productie is genomen en werkt op basis van monitoring van het systeem/de service.

Testtypen: Security testing

- Identificeer welke niet-functionele kenmerken, zoals beveiliging, prestaties, betrouwbaarheid of bruikbaarheid, cruciaal zijn voor succes en hun testaanpak.
- Gezien het belang van beveiliging kunnen Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), Software Composition Analysis

(SCA), penetratietesten en API security testen genoemd worden als (onderdeel van) de security testaanpak.

Testomgeving

- Testomgevingen worden weergegeven voor de verschillende testfasen in de CI/CD-pipeline (bijvoorbeeld ontwikkeling, staging, pre-productie), die elk een andere focus vertegenwoordigen. Ze zijn meestal ook afgestemd op de verschillende geïdentificeerde testsoorten (bijv. unit-, integratie-, systeem- en acceptatietesten). Omdat DevOps zich richt op verregaande automatisering van deployment- en configuratie werkzaamheden voor productieomgevingen, met behulp van toepassingen zoals configuratie-als-code, ligt het voor de hand om dezelfde toepassingen te gebruiken bij het ontwerpen en bouwen van testomgevingen. Bij gebruik van cloudgebaseerde omgevingen is het gebruikelijk om op aanvraag een testomgeving te creëren en deze na gebruik op te ruimen, wat resulteert in een gestandaardiseerde, gecontroleerde en stabiele testomgeving welke op aanvraag beschikbaar is wanneer deze benodigd is. Bij implementatiestrategieën zoals Blue-Green-implementatie kan de testomgeving van de nieuwste versie van ons product de productieomgeving worden na succesvolle voltooiing van de quality gates en DoD. In dit geval wordt de vorige productieomgeving hergebruikt als de volgende testomgeving (om te schakelen tussen de Blue- en Green-omgeving).

Het lean teststrategie document is vaak een goede oplossing voor DevOps-organisaties en een alternatief voor gedetailleerde (project)testplannen. De specifieke TMMi-doelstelling 'Een teststrategie opstellen', inclusief de specifieke toepassingen, is volledig toepasbaar op organisaties die DevOps werken. Net als bij het testbeleid is het belangrijk om ervoor te zorgen dat de teststrategie wordt verspreid over de gehele organisatie/waardeketen/Agile release train/tribe.

2.1.3 SG3 Opstellen Testprestatie-indicatoren

De bedrijfsdoelstellingen voor testverbetering, zoals gedefinieerd in het testbeleid, moeten worden vertaald in een set belangrijke testprestatie-indicatoren. Het testbeleid en de bijbehorende prestatie-indicatoren geven een duidelijke richting en een manier om verwachte en behaalde testprestatieniveaus te communiceren. De prestatie-indicatoren moeten de waarde van testen en testprocesverbetering aan de stakeholders aangeven. Omdat investeringen in procesverbetering langdurige managementondersteuning vereisen, is het cruciaal om de voordelen van een verbeterprogramma kwantitatief te meten om hen betrokken te houden. Houd er rekening mee dat deze specifieke TMMi-doelstelling gaat over het definiëren van een beperkt aantal (bijv. 2 of 3) testprestatie-indicatoren. Het gaat niet om het opzetten en implementeren van een volledig meetprogramma, maar om het definiëren van een kernset indicatoren die aangeven hoe de waarde van testen in de loop der tijd en binnen verschillende leveringsomgevingen verbeterd.

Net als bij Agile zal ook bij DevOps de focus meer op teamgericht en systeemdenken liggen. Dit kan resulteren in een overeenkomstige verbreding van de indicatoren naar prestatie-indicatoren voor ontwikkelings- en implementatieprocessen in plaats van zich uitsluitend te beperken tot de specifieke kenmerken van testen zelf. De belangrijkste DevOps-prestatie-indicatoren volgen doorgaans de DevOps Research and Assessment (DORA)-metrics. Hier valt op dat in plaats van de (test)dekking, het percentage mislukte wijzigingen wordt gemeten tijdens de implementatie naar productie (percentage mislukte implementaties).

Voorbeelden van DORA metrics:

- Prestatiegegevens voor softwarelevering: implementatiefrequentie, doorlooptijd voor wijzigingen, tijd voor herstel van de service en percentage mislukte wijzigingen
- Operationele prestaties: betrouwbaarheidsgegevens, bijvoorbeeld MTTR en MTTF=

Prestatie-indicatoren moeten een integraal onderdeel zijn van de CI/CD-activiteiten en de data moet vrijwel realtime beschikbaar zijn voor iedereen binnen de organisatie. Hierbij kunnen dashboards en monitoring een grote rol spelen. Naast de DORA-metrieken worden ook flow-metrieken gebruikt ter ondersteuning van Value Stream Management/-optimalisatie. Testactiviteiten kunnen worden gemeten in lijn met de bovenstaande metrieken, zodat hun bijdrage begrijpelijk is.

Voorbeelden van flowmetrieken voor Value Stream Management met betrekking tot testen:

- Prestatie-indicatoren moeten een integraal onderdeel zijn van de CI/CD-activiteiten en de data moet vrijwel realtime beschikbaar zijn voor iedereen binnen de organisatie. Hierbij kunnen dashboards en monitoring een grote rol spelen.
- Naast de DORA-metrieken worden ook flow-metrieken gebruikt ter ondersteuning van Value Stream Management/-optimalisatie. Testactiviteiten kunnen worden gemeten in lijn met de bovenstaande metrieken, zodat hun bijdrage begrijpelijk is.

De uitdaging is om de juiste mix van indicatoren te definiëren die betrekking hebben op de DevOps-aanpak en systeemdenken, en tegelijkertijd een goede indicatie te geven van de prestatieresultaten van een op TMMi gebaseerd testverbeteringsprogramma. De TMMi specifieke doelstelling (SG3) het vaststellen van testprestatie-indicatoren, inclusief de specifieke toepassingen, is volledig bruikbaar, hoewel de geselecteerde en toegepaste prestatie-indicatoren wellicht een bredere scope hebben en breder zijn dan alleen gerelateerd aan testen. Dit laatste maakt de analyse en interpretatie van prestatie-indicatoren uiteraard uitdagender. Sterker nog, de gebruikte prestatie-indicatoren worden mogelijk zelfs niet expliciet testprestatie-indicator genoemd. Dit is nog steeds acceptabel in de context van TMMi, zolang ze test gerelateerde elementen bevatten en worden gebruikt om de voortgang van testverbetering te evalueren.

2.2 Procesgebied 2.2 Test Planning

Het doel van het procesgebied Testplanning is het definiëren van een testaanpak op basis van de geïdentificeerde risico's en de gedefinieerde teststrategie. Daarnaast wordt een gedegen testplan opgesteld en onderhouden dat dient als basis voor het uitvoeren en het managen van de testactiviteiten.

Let op dat succesvol plannen afhangt van het vooraf nadenken als activiteit, niet in het definiëren van het test plan zelf ("het document").

In de DevOps context zijn er meestal twee soorten planningsactiviteiten: de releaseplanning en de sprintplanning. Het procesgebied Test Planning in TMMi level 2 focust op test-gerelateerde activiteiten van zowel de release- als de sprintplanning. Tijdens de releaseplanning wordt voorafgaand aan het project vooruitgekeken naar de release. Voor een releaseplanning is een goed gedefinieerde backlog een vereiste. Het kan nodig zijn om grotere User stories te verdelen in kleinere stories. Let er op dat DevOps en Agile meestal werken met vaste deadlines, maar met flexibele functionele inhoud. Het is belangrijk om te zorgen dat er alleen functionaliteit in de release zit die getest is volgens de gedefinieerde testaanpak en de DoD. Het resultaat van de releaseplanning is de basis voor de testaanpak en het testplan dat geldt voor alle sprints tot de releasedatum. Een releaseplan is een plan dat in grote lijnen de overkoepelende strategie of aanpak van het project beschrijft, en geeft geen gedetailleerde omschrijving van testtaken. Na de releaseplanning kan men beginnen met het plannen van de eerste sprint. In de planningsactiviteit van een enkele sprint kijkt men vooruit naar het gewenste resultaat, en wordt de backlog van de sprint ingedeeld volgens het plan.

2.2.1 SG1 Uitvoeren Productrisicoanalyse

Uitputtend testen is onmogelijk. Er zullen altijd keuzes gemaakt moeten worden om tot de juiste prioritering te komen. Deze TMMi doelstelling is daarom ook toepasbaar voor DevOps projecten. Voor DevOps projecten

zal een hoog-over risicoanalyse plaatsvinden tijdens de releaseplanning, op basis van documenten die de projectvisie of doelen beschrijven. Voor de individuele sprints zullen user stories of andere requirements behandeld worden in gedetailleerde risicoanalyse sessies, als deel van de sprintplanning. Eventueel kan de risicoanalyse van stories ook plaatsvinden tijdens de refinement sessies, waar de stories besproken worden en de risico's geëvalueerd. De productrisicoanalyse in DevOps zal een lichter karakter hebben dan de analyses die plaatsvinden in een traditioneel project met een watervalmodel. Voorbeelden van eenvoudige en lichtgewicht methodes voor het uitvoeren van productrisicoanalyses zijn risiscopoker [Veenendaal12] en ROAM (Resolved, Owned, Accepted, Mitigated) [Baah]. Tijdens de releaseplanning of de program increment planning (PI planning) wordt een hoog-over overzicht van de te ontwikkelen functionaliteit gepresenteerd door vertegenwoordigers van de business, die bekend zijn met de features. Zij bepalen ook de niet-functionele beperkingen, waarop het team, inclusief de tester(s), assisteert in het identificeren en inschatten van de risico's.

Tijdens de sprintplanning identificeert en analyseert het DevOps team de productrisico's op basis van de user stories die klaarstaan voor de komende sprint. Het liefst zijn alle leden van het Agile team aanwezig, en mogelijk ook andere stakeholders. Het resultaat van deze analyse is een geprioriteerde lijst met alle kritieke testgebieden. Inschatten hoeveel testinzet nodig is voor het afdekken van de risico's zal het planningsproces helpen. De testen en de testtaken zullen gepland worden in een volgorde die de efficiëntie en effectiviteit bevordert, waarbij taken die hogere risico's afdekken een hogere prioriteit krijgen en meer inzet vereisen. Taken met een lager risicoprofiel zullen later uitgevoerd worden en minder testinzet vragen.

Met het belang van data- en cybersecurity in veel organisaties kan het baten om threat modeling uit te voeren, buiten de bovengenoemde methoden van risicoanalyse. Threat modeling wordt gebruikt om veiligheidsbelangen, bedreigingen en potentiële zwakke plekken te identificeren. De bedreigingen voor de veiligheid en eventuele zwakheden worden gekwantificeerd, en zodoende worden maatregelen voor het mitigeren van deze risico's ook geprioriteerd. Deze kwantificatie gebeurt volgens richtlijnen en gebruikt de waarschijnlijkheid en impact van de bedreigingen om te komen tot een maat van ernst.

2.2.2 SG2 Opstellen Testaanpak

Een testaanpak wordt opgesteld om de geïdentificeerde en geprioriteerde risico's te mitigeren. Voor een specifieke sprint worden de uit te voeren testwerkzaamheden als items en features geïdentificeerd tijdens de sprintplanning. Deze activiteit wordt ook gebaseerd op de resultaten van de productrisicoanalyse. De items die getest moeten worden zijn meestal gerelateerd aan de te bouwen features in de sprint. De features relateren aan de kwaliteitskenmerken die getest moeten worden. Nieuwe productrisico's worden soms pas duidelijk tijdens de sprint, en vergen extra testwerk. Kwesties over de gevolgen hiervan worden meestal besproken tijdens de dagelijkse standup sessies.

In DevOps wordt de teststrategie gebaseerd op het continue testen in de CI/CD pipeline. Continu testen betekent vaak en vroeg testen, waarbij gebruik gemaakt wordt van automatisering met bruikbare feedback die past bij elk stadium van de delivery pipeline. Continuous Integration (CI) en Continuous Deployment (CD) software wordt meestal automatisch gestart en omvat activiteiten zoals bouwen, integratie, testen, security, compliance en opleveren van het product. De testactiviteiten in de CI/CD pipeline zijn meestal gebaseerd op de testsoorten die uit de Agile Testing Quadrants komen, en worden continu verbeterd door de teams.

De testaanpak die wordt opgesteld op sprintniveau of op backlog-item niveau omvat meestal stappen zoals extra reviews op user stories en acceptatiecriteria. De inzet is hierin afgeleid van het risico, en de toepasbare testtechnieken hangen af van het type en de grootte van het risico. Mogelijk worden ook testactiviteiten beschreven die pas na de deployment uitgevoerd worden, zoals crowd testing in productie om gebruikersfeedback te verzamelen, of performance testen met echt gebruikersverkeer tijdens een dark launch. In de context van DevOps zijn Behavior-Driven Development (BDD) en Acceptance Test-Driven

Development (ATDD) populaire technieken, die een basis geven voor het identificeren en ontwerpen van de testcases. De testaanpak op releaseniveau zal minder gedetailleerd zijn en zal afgeleid zijn van de testaanpak op programma- of organisatieniveau. De testaanpak wordt meestal opgeslagen en gedeeld op een wiki van het team of het project.

Een belangrijk en alomtegenwoordig risico in agile ontwikkeling is het risico op regressie. De testaanpak moet definiëren hoe het risico op regressie wordt verminderd. Meestal wordt hier een specifieke testset gemaakt die het liefst automatisch uitgevoerd kan worden. In deze context is het handig om de Test Automation Pyramide te raadplegen [Cohn]. Het toont hoe de waarde van geautomatiseerde regressietesten gemaximaliseerd kan worden. De piramide begint met unit testen onderaan, beschrijft dan testen op serviceniveau, en helemaal in het puntje de testen vanuit de gebruikersinterface (UI). Unit testen zijn snel en betrouwbaar. De servicelaag wordt gebruikt om business rules te testen op het API- of serviceniveau, waarin je niet gehinderd wordt door de UI. Hoe hoger de test in de piramide, hoe langzamer en fragieler de test wordt. Omdat automatisering een belangrijk deel is van DevOps, is ook het automatiseren van testen de primaire focus, en niet alleen voor regressietesten. Een andere manier om met regressie om te gaan is om het systeem zo observeerbaar mogelijk te maken. Zo kan de status continu gemonitord worden, en kunnen automatische mechanismen gebouwd worden om risico's te voorkomen, zoals automatische roll-backs naar een stabiele versie.

Entrycriteria (ook wel Definition-of-Ready genoemd) zijn normaal gesproken onderdeel van een gedefinieerde testaanpak (specifieke toepassing 2.3), maar worden in DevOps-ontwikkeling waarschijnlijk anders behandeld. Binnen DevOps-softwareontwikkeling is testen een integraal onderdeel van het teamproces en worden testen tijdens de sprint continu uitgevoerd. De checklist voor toelatingscriteria is verdeeld in de Definition-of-Ready, bijvoorbeeld het definiëren van acceptatiecriteria als testscenario's met Given-When-Then-statements, controles binnen de CI/CD-machine, controles binnen een pull-request, het uitvoeren van een smoketest op een recent geïmplementeerde testomgeving vóór de start van een grotere testuitvoeringsactiviteit, of het controleren van de implementatie van benodigde testgegevens. Daarom moet een specifieke checklist of gateway die vast stelt of er wel of niet gestart kan worden met testen, geïmplementeerd worden in de CI/CD workflow en de build laten falen o.b.v. non-conformiteit, net als bij ieder ander defect. Dit geldt ook voor een component die binnen een DevOps-team van de ene testfase (bijv. unit testing) naar de andere (bijv. acceptatietest) gaat.

Bepalen exit criteria (Specifieke toepassing 2.4) maakt deel uit van de zogenaamde Definition-of-Done (DoD), die sterk afhankelijk is van kwaliteitscontroles die binnen de CI/CD-pipeline worden bereikt. Het is belangrijk dat de DoD specifieke testgerelateerde criteria hanteert, bijvoorbeeld voor testdekking en productkwaliteit. De sprint moet resulteren in de implementatie van de overeengekomen set aan user stories en voldoen aan de (test)exit criteria zoals gedefinieerd in de DoD. In DevOps betekent "DONE" meestal dat de codewijzigingen met betrekking tot de backlogitem(s) in productie worden geïmplementeerd en alle kwaliteitscontroles binnen de CI/CD-pipeline zijn geslaagd. In het geval dat releases on-demand plaatsvinden en losgekoppeld zijn van de Deployment met bijvoorbeeld Feature Toggles, heeft elke release een eigen DoD die mogelijk over meerdere sprints loopt. Het doel van DevOps is echter om zo klein mogelijke releases met hoge frequentie te releasen. Op releaseniveau hanteert de DoD doorgaans criteria die verband houden met de dekking en productkwaliteit. Meestal ligt de nadruk hierbij op het valideren of de verwachte functionaliteit aan de gebruikers is geleverd op basis van testactiviteiten die in de productie plaatsvinden, zoals crowd testing of canary testing.

Specifieke toepassing 2.5 'Definieer opschortings- en hervattingscriteria' is waarschijnlijk niet relevant voor DevOps sprints. Omdat testen een integraal onderdeel is van het DevOps-softwareontwikkelingsproces, wordt het niet als een aparte en onafhankelijke activiteit buiten de sprint behandeld. Wanneer er blokkerende problemen zijn die kunnen worden beschouwd als potentiële of daadwerkelijke bedreigingen voor de voortgang van het testen, worden deze besproken tijdens de dagelijkse standup. Tijdens de dagelijkse standup bepaalt het team eventueel welke acties nodig zijn om de problemen op te lossen. Formele opschortings- en

hervattingscriteria zijn dus niet vereist en gedefinieerd; problemen worden behandeld als onderdeel van sprint. De sprint dient dus als een alternatief voor deze specifieke toepassing.

2.2.3 SG3 Opstellen Testbegroting

Gedetailleerde schattingen voor testactiviteiten worden gemaakt tijdens de sprintplanning. Schattingen op hoog niveau worden gemaakt tijdens de releaseplanning en mogelijk ook tijdens refinements. Alle schattingen worden gedaan op teamniveau, waarbij het team inschat hoeveel effort het gaat kosten om de user stories op te leveren. Het is belangrijk om ervoor te zorgen dat testactiviteiten daadwerkelijk mee worden genomen tijdens de refinements. Dit kan door testactiviteiten te identificeren als afzonderlijke taken en vervolgens elke individuele testtaak in te schatten, of door elke user story te schatten, waarbij de uit te voeren testactiviteiten expliciet worden gemaakt en dus worden meegenomen in de schatting van het team. Omdat in DevOps het gedachtengoed 'automate everything' bestaat, wordt automatisering van testgerelateerde activiteiten verwacht. Deze inzet moet ook worden geschat, inclusief testautomatisering, automatisering van de implementatie van testomgevingen, automatisering van de implementatie van testgegevens, evenals verbeteringen van de pipeline die in lijn zijn met de testaanpak voor de sprint. Tijdens de schatting zal een tester, die deel uitmaakt van het DevOps-team, deelnemen aan de refinements. Planning Poker, T-shirt sizing, swimlane sizing (bucket sizing) en dot voting zijn allemaal typische technieken die worden gebruikt in DevOps.

Werk dat in de volgende sprint moet worden gedaan, wordt doorgaans gedefinieerd door de user stories. Deze user stories moeten klein van omvang zijn om goed ingeschat te kunnen worden. Inschatbaarheid (Estimable) is een van de user story-criteria die INVEST definieert en is van toepassing op user stories die gepland zijn voor een sprint. Tijdens de sprintplanning identificeren en definiëren Agile teams doorgaans testgerelateerde taken. Testtaken worden samen met de andere ontwikkeltaken vastgelegd op een bord. Tijdens de retrospectieve kijken teams terug op hoe nauwkeurig de stories zijn ingeschat (indien dit als een probleem is aangemerkt) en verbeteren ze hun schattingsvermogen op basis van de werkelijke inzet van uitgevoerde taken.

Tijdens de releaseplanning worden user stories of epics doorgaans op een hoger niveau gedefinieerd en nog niet opgebroken in specifieke taken. Dit maakt de schattingen uiteraard moeilijker en minder nauwkeurig. Zoals gezegd zullen DevOps-projecten geen Work-Breakdown opstellen als basis voor schattingen, maar kan het op release-niveau wel baat hebben om informatie te verzamelen over de 'hoe en wat' van de Backlog Items, in ieder geval genoeg om de benodigde schattingen te doen en om de items te kunnen prioriteren. Omdat het doel bij DevOps is om de waarde-stroom te optimaliseren, moet de focus zijn op het ondersteunen van de prioritering, en juist niet op het creëren van een volledige projectplanning.

Hoewel een DevOps-team op een relatief informele manier schattingen maakt, moet de onderbouwing van de schattingen duidelijk zijn (d.w.z. welke factoren meespelen). Onderbouwde discussies bevorderen een hogere nauwkeurigheid van de schattingen. In DevOps-projecten is de schatting doorgaans gericht op omvang (met story points) of inspanning (met ideale werkdagen als eenheid voor de schatting). Kosten worden doorgaans niet besproken tijdens refinements in DevOps-projecten. Houd er rekening mee dat kosten voor infrastructuur doorgaans hoog zijn in DevOps-context; de kostenramingen worden meestal vastgesteld in specifieke schattingssessies, niet in team refinements. Nauwlettende monitoring en controle van infrastructuurkosten, bijvoorbeeld voor het gebruik van de publieke cloud, is noodzakelijk en kan worden geïntegreerd in de CI/CD-pipeline, met automatische rapportage. (Bijvoorbeeld elke uitgevoerde GitHub Action-taak weergegeven in vCPU-minuten).

Deze specifieke doelstelling en de bijbehorende specifieke toepassingen zijn volledig bruikbaar, met uitzondering van specifieke toepassing 3.2: Definiëren testfasering. Een van de basisprincipes van iteratieve en DevOps-softwareontwikkeling is werken in kleine stukjes. De taken op het sprintbord zijn doorgaans

gedetailleerd genoeg om als basis te dienen voor (test)schattingen. In de DevOps-context is het daarom nodig om ook een levenscyclus voor testtaken te definiëren als aanvullende basis voor schattingen.

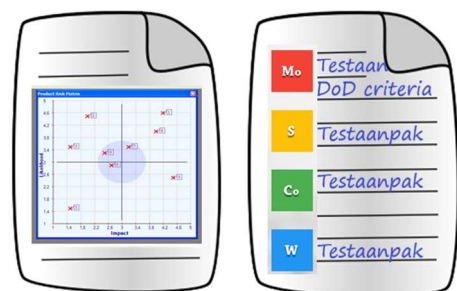
2.2.4 SG4 Opstellen Testplan

Herhaling van de opmerking dat testplanning draait om vooruitdenken ("de activiteit") en niet om het definiëren van het bijbehorende testplan ("het document"). In DevOps worden de meeste testplanningsactiviteiten, zoals gedefinieerd door deze specifieke TMMi doelstelling, uitgevoerd tijdens release- en sprintplanning, en tijdens sessies voor het refinieren van backlogitems. Het resultaat van deze activiteiten wordt echter doorgaans niet vastgelegd in een testplan, vooral niet bij een sprintplanning, waar ze op het takenbord worden weergegeven. Omdat testen een integraal onderdeel is van de release- en sprintplanning, zullen de resulterende inschattingen ook testactiviteiten omvatten. In plaats van een gedetailleerde planning, zoals ontwikkeld bij waterval projectmanagement, lijkt de planning binnen een DevOps-project veel meer op een volgorde van user stories (backlogitems) en taken die de release- en sprintprioriteiten weerspiegelt, bijvoorbeeld op basis van de gewenste levering van bedrijfswaarde. Houd er rekening mee dat testuitvoering voornamelijk plaatsvindt binnen de CI/CD-pipeline, getriggerd door codecommits. Aanvullende, handmatige tests of activiteiten die in productie worden uitgevoerd, kunnen worden gepland op het sprintbord. Mindmaps worden hierbij vaak gebruikt als ondersteunende techniek. Het takenbord weerspiegelt de sprintprioriteiten. Er wordt dus geen expliciete (test)planning vastgesteld; er wordt verwacht dat er duidelijke release- en sprintprioriteiten worden gedefinieerd voor de user stories, namelijk de uit te voeren taken, inclusief de testtaken.

Op projectniveau maakt de testbezetting deel uit van de teamopbouw; de behoefte aan resources, test gerelateerd of multidisciplinair, wordt vooraf vastgesteld. Terwijl projecten veranderen en groeien kan men makkelijk uit het oog verliezen waarom een persoon is geselecteerd voor een bepaald team of project. Vaak omvat de redenering een specifieke vaardigheid of ervaring die vitaal is voor het team of project. Dit biedt een goede reden om de benodigde vaardigheden van mensen te documenteren, wat kan dienen als back-up informatie over waarom bepaalde mensen zijn geselecteerd voor een specifiek team of project. Als de DevOps-teams eenmaal zijn besloten, dan zullen er zo min mogelijk personele wijzigingen zijn. Tijdens de sprintplanning kan besproken worden of er aanvullende testvereisten zijn, bijvoorbeeld voor het testen van niet-functionele kwaliteits-karakteristieken, om te zorgen dat benodigde vaardigheden, kennis en resources aanwezig zijn om de noodzakelijke testen uit te voeren.

Een initiële projectrisicosessie zou onderdeel moeten zijn van de release- en sprintplanning. Het identificeren (en het beheer) van verdere projectrisico's tijdens de sprint maakt deel uit van de dagelijkse stand-up en de Scrum-of-Scrums, en wordt doorgaans vastgelegd in een impediment log. Het is belangrijk dat testobstakels ook in het impediment log worden genoteerd. De obstakels moeten worden besproken tijdens de dagelijkse stand-up totdat ze zijn opgelost, of tijdens de Scrum-of-Scrums als het team niet de bevoegdheid heeft om ze op te lossen.

Hoewel er doorgaans geen specifiek en gedetailleerd testplan wordt ontwikkeld en gedocumenteerd, is specifieke doelstelling 4.5 'Opstellen testplan' nog steeds relevant binnen een DevOps-context. De resultaten van de discussies die plaatsvinden in de context van testplanning zullen echter waarschijnlijk worden vastgelegd in een eenvoudige vorm, mogelijk een mindmap of een document van één pagina (zie figuur 4), een Agile Testing Kwadrant op teamniveau en de testactiviteiten binnen de CI/CD-pipeline.



Afbeelding 4: One Page Test Plan

2.2.5 SG5 Verkrijgen Testplan Commitment

Binnen DevOps is het proces voor het ontwikkelen en vaststellen van een testaanpak en testplan een teamactiviteit, mogelijk onder leiding van een tester (één van de teamleden). Productkwaliteit is de verantwoordelijkheid van het team. Mits het team het juiste proces volgt en de overeengekomen testactiviteiten zijn ingebouwd in de CI/CD-pipeline, is commitment aan de (test)aanpak en het (test)plan dan ook al een impliciet resultaat van de release- en sprintplanning, aangezien ook deze activiteiten een teaminspanning zijn. Dit is natuurlijk een enorm verschil met de werkwijze in een traditionele waterval omgeving, waar de tester doorgaans het testplan voorbereidt en vervolgens expliciet commitment moet verkrijgen. De resources en de infrastructuur die nodig zijn voor de CI/CD-pipeline om het Testplan uit te voeren (in de vorm van geautomatiseerde tests en testomgevingen) moeten worden overwogen, en geleverd worden door de stakeholders.

In een DevOps-context moet het team (inclusief de product owner) de geprioriteerde lijst met productrisico's en de uit te voeren maatregelen voor het mitigeren van de risico's begrijpen en het hierover eens zijn. Overeenstemming en commitment kunnen bijvoorbeeld worden bereikt door middel van een korte presentatie, gevolgd door een discussie tijdens de release- of sprintplanning, waarin de productrisico's, de testaanpak en de onderbouwing ervan aan het team worden uitgelegd.

Tijdens refinements (zie SG 3 Opstellen testbegroting) wordt de workload geschat. De (test)resources voor een team liggen vast in de samenstelling van het DevOps-team. User stories die worden ingeschat en geselecteerd voor ontwikkeling, nemen de beschikbare resources als uitgangspunt. Het afstemmen van werk- en resource-niveaus is dus geen zinvolle activiteit, liever is er een continue focus op technische resources, zoals de beschikbaarheid van cloud computing, opslag en netwerk, en worden deze essentiële resources aan het team verstrekt. De specifieke toepassing 3.2 'Begroot de testinspanning en -kosten' is daarom een toepassing die doorgaans niet relevant is in een DevOps-context. De dagelijkse stand-up en de Scrum-Of-Scrums worden gebruikt om missende resources aan te pakken, door deze toe te wijzen of door een deliverable uit de sprint te halen en opnieuw te plannen in de releaseplanning.

2.3 Procesgebied 2.3 Testbewaking en -beheersing

Het doel van het procesgebied Testbewaking en -beheersing is goed inzicht te hebben in de voortgang van het testtraject en de kwaliteit van de opgeleverde producten. Hierdoor kunnen tijdig de juiste correctieve acties worden genomen als de voortgang van het testtraject significant afwijkt van de planning of de kwaliteit van de opgeleverde producten significant afwijkt van de verwachtingen

Er zijn een aantal zaken die fundamenteel verschillen tussen testbewaking en -beheersing in DevOps-context en traditionele waterval projecten. Hoewel testbewaking en -beheersing essentiële elementen zijn binnen DevOps, betekent dit niet dat het doel is om vast te houden aan een rigide plan. Vanuit het perspectief van testbewaking en -beheersing betekent dit dat we niet plan gedreven te werk gaan, maar dat we onze voortgang en testresultaten continu evalueren en ons plan en onze aanpak indien nodig aanpassen, bijvoorbeeld wanneer nieuwe productrisico's zich openbaren.

Het testplan, dat meestal verdeeld is over verfijnde acceptatiecriteria, backlogitems (inclusief hun risiconiveau) en taken, is een levend geheel en moet continu worden beoordeeld en bijgewerkt naarmate nieuwe informatie beschikbaar komt of feedback wordt gegeven. De doelstellingen voor testactiviteiten worden bepaald door de set acceptatiecriteria voor het betreffende backlogitem, de testaanpak van het team die voor het betreffende backlogitem wordt bijgewerkt, en op backlogniveau worden specifieke Definition-of-Done-criteria gedefinieerd. DevOps-projecten moeten ook 'the bigger picture' bewaken en beheersen, zowel op CI/CD-niveau als op sprintniveau.

Het is belangrijk om te onthouden dat testen in een DevOps-team een proces is dat volledig geïntegreerd is. Testbewaking en -beheersing zijn dan ook een integraal onderdeel van de algehele bewaking en beheersing van het DevOps-team. Hierdoor rapporteren testers niet aan een testmanager, zoals bij traditionele projecten, maar aan het team.

2.3.1 SG1 Bewaken testvoortgang t.o.v. het testplan

DevOps-teams gebruiken verschillende methoden en tools om de testvoortgang te bewaken en vast te leggen, zoals de voortgang van testtaken en -stories op het Agile-takenbord en burndown-grafieken. Deze kunnen vervolgens aan de rest van het team worden gecommuniceerd via media zoals wiki-dashboards en e-mails in dashboardstijl, evenals mondeling tijdens dagelijkse stand-up. Teams kunnen burndown-grafieken gebruiken om de voortgang over de gehele release en binnen elke sprint te volgen. Een burndown-grafiek toont doorgaans de voortgang ten opzichte van de verwachte snelheid en de achterstand in de te implementeren features (en toont dus de prestaties van het team). Daarnaast wordt er in DevOps continu bewaakt met online dashboards die de infrastructuur, de applicatie en het netwerk geautomatiseerd in de CI/CD-pipeline monitoren. De resources van de testomgeving worden continu gemonitord ten opzichte van de resources die zijn overeengekomen tijdens de sprintplanning en voor het gebruik van de CI/CD-pipeline. Een andere veelgebruikte methode is om problemen in de testomgeving bij te houden via het takenbord, bijvoorbeeld door stickers met de tekst 'geblokkeerde testomgeving' op de storycards te plakken of door een aparte kolom op het bord te creëren waar alle stories wachten die door omgevingen geblokkeerd zijn, tot ze weer vrijgegeven worden. Het doel hiervan is om de impact van een probleem met de testomgeving zichtbaar te maken en de aandacht van het team te vestigen op het oplossen van dit probleem, voordat er aan een andere taak wordt gewerkt.

Om direct een gedetailleerde visuele weergave te bieden van de huidige status van het team en de CI/CD pipeline, inclusief de teststatus, gebruiken teams doorgaans task boards. De storycards, ontwikkeltaken, testtaken en andere taken die tijdens de sprintplanning zijn gemaakt, worden vastgelegd op het bord. Tijdens de sprint wordt de voortgang beheerd door deze taken over het takenbord te verplaatsen naar kolommen zoals 'te doen', 'werk in uitvoering' en 'klaar'. Om de flow te behouden, worden er vaak limieten voor 'Werk in uitvoering' (WIP) toegepast op verschillende statussen op een Kanban-bord. Dit stuurt het gedrag van het team om taken (in uitvoering) af te ronden in plaats van te beginnen met nieuwe taken. DevOps-teams gebruiken doorgaans tools om hun storycards op takenborden bij te houden, waarmee dashboards en statusoverzichten kunnen worden geautomatiseerd. Sommige teams maken echter geen specifieke taken aan voor de individuele activiteiten, maar gebruiken alleen de storycard en annoteren opmerkingen in deze kaart voor tests, met verwijzingen naar tools of wiki's waar de tests mogelijk worden gedocumenteerd. Testtaken op het takenbord hebben doorgaans betrekking op de acceptatiecriteria die zijn gedefinieerd voor de user stories en de niet-functionele vereisten die een kader vormen voor elk backlogitem. Zodra testautomatiseringsscripts en exploratory testen voor een testtaak geslaagd zijn, wordt de taak verplaatst naar de kolom 'Gereed' op het takenbord.

De Definition-of-Done (DoD) dient als exit-criterium waaraan voortgang wordt gemeten. De DoD moet ook gerelateerd zijn aan testactiviteiten en toont alle criteria waaraan moet worden voldaan voordat de ontwikkeling en het testen van een user story 'Done' kunnen worden genoemd. Merk op dat testcriteria met betrekking tot de testtaken slechts een deel vormen van wat het team includeert in de Definition-of-Done. Definition-of-Done criteria worden doorgaans op meerdere niveaus toegepast, bijvoorbeeld op story- en epic-niveau. Het hele team bekijkt de status van het takenbord regelmatig, vaak tijdens de dagelijkse standup, om ervoor te waken dat taken in een acceptabel tempo worden uitgevoerd. Als taken (inclusief testtaken) niet of

te langzaam worden uitgevoerd, leidt dit tot een teamdiscussie waarin de mogelijke problemen worden geanalyseerd die de voortgang van die taken blokkeren.

Tijdens de dagelijkse standup worden alle leden van het DevOps-team, inclusief testers, betrokken. Tijdens deze meeting communiceren ze hun huidige status en voortgang met de rest van het team. Eventuele belemmeringen die de voortgang van de ontwikkeling of test kunnen blokkeren, worden tijdens de dagelijkse standup gecommuniceerd, zodat het hele team hiervan op de hoogte is en ernaar kan handelen. Op deze manier is projectrisicomanagement geïntegreerd in deze dagelijkse meetings. Het team kan regelmatig besluiten een ROAM-oefening uit te voeren ter ondersteuning van de monitoring van de status van projectrisico's [ROAM]. Elk projectrisico, inclusief die voor testen, zoals het gebrek aan beschikbaarheid van testomgevingen, kan tijdens de dagelijkse stand-up worden gecommuniceerd en aangepakt. (Let op: het bewaken van productrisico's maakt deel uit van het bewaken van productkwaliteit en wordt daarom hierna besproken met de specifieke doelstelling SP 2: 'Bewaken productkwaliteit ten opzichte van testplan en verwachtingen'). Een dagelijkse standup voor dagelijks taakbeheer en DevOps-team activiteiten met betrekking tot voortgangsbewaking zijn goede, bewezen technieken die voldoen aan de intentie van TMMi-specifieke toepassingen in het procesgebied Testbewaking en -beheersing.

Aan het einde van de sprint vindt een sprintreview plaats ten opzichte van de sprintdoelen. Het team demonstreert waaraan is gewerkt, bijvoorbeeld welke stories of epics er zijn afgerond. Integratie via de CI/CD-pipeline is er uiteraard continu. De resultaten van testen, bijvoorbeeld ten opzichte van de Definition-of-Done, maken deel uit van de sprintreview. Er worden demo's georganiseerd met stakeholders om de bedrijfswaarde en kwaliteit van het opgeleverde product te bespreken. Stakeholders worden door de product owner vertegenwoordigd in de sprintplanning, sprintreviews (demo's) en retrospectives. De product owner is betrokken in discussies over de product backlog en levert feedback en input voor de uitwerking van user stories en het ontwerp van testen gedurende de sprint. Andere stakeholders worden aan het einde van elke sprint betrokken tijdens de sprintreview. Er is geen specifieke bewaking van stakeholder betrokkenheid vereist, omdat de vertegenwoordiging van stakeholders door een Product Owner al ingebakken is in de Agile manier van werken.

2.3.2 SG2 Bewaken productkwaliteit t.o.v. het testplan en verwachtingen

Voor het bewaken van productkwaliteit worden grotendeels dezelfde mechanismen gebruikt als voor het bewaken van de testvoortgang (zie SG1 hierboven). Het bewaken van de productkwaliteit met behulp van een online dashboard wordt niet alleen binnen teams uitgevoerd tijdens het ontwikkelen en testen, maar ook door continu de productkwaliteit te meten in productie. 'Ops' rollen bewaken continu de kwaliteit van het product met behulp van observability, meestal gedefinieerd als Service Level Indicators (SLI's). Regressie of afwijking in SLI's triggert corrigerende maatregelen om de kwaliteit en gebruikerservaring te behouden. In DevOps ligt de focus bij het bewaken van productrisico's op regelmatige vergaderingen waarin lijsten met productrisico's worden gereviewd, inclusief beveiligingsrisico's/bedreigingen. Dit in plaats van het reviewen van gedetailleerde risicodocumentatie. Nieuw ontdekte of gewijzigde productrisico's, bijvoorbeeld als resultaat van exploratory testing, worden besproken om tot de vereiste testacties te komen. Indien nodig kan een extra ROAM-oefening worden uitgevoerd [ROAM]. De status van de verschillende productrisico's wordt doorgaans weergegeven met behulp van grafieken op het dashboard.

DevOps-teams gebruiken defect-gebaseerde statistieken die vergelijkbaar zijn met die van traditionele ontwikkelmethodes, zoals pass-/failpercentages, defect discovery rates, en het aantal gevonden en opgeloste defecten, om de productkwaliteit te bewaken en te verbeteren. Het aantal defecten dat tijdens een sprint

wordt gevonden en opgelost, het aantal defecten dat tijdens continuous integration wordt gevonden, en het aantal onopgeloste defecten die mogelijk op de backlog komen voor de volgende sprint, moeten worden gemonitord tijdens de dagelijkse standup. Om de algehele productkwaliteit te bewaken en te verbeteren gebruiken veel DevOps-teams ook klanttevredenheidsonderzoeken om feedback te krijgen over de vraag of het product aan de verwachtingen van de klant voldoet.

Exit criteria voor testen, bijvoorbeeld voor testdekking en productkwaliteit, maken deel uit van de Definition-of-Done (DoD). De naleving van overeengekomen exit criteria wordt doorgaans bewaakt via het takenbord, waarbij een story alleen als 'done' kan worden aangemerkt als deze voldoet aan de DoD. Deze criteria omvatten doorgaans de criteria dat het betreffende backlogitem in productie is geïmplementeerd (het is 'done' wanneer het is geïmplementeerd, en niet andersom). Gewoonlijk worden specifieke exit criteria gedefinieerd voor continuous testing. Defecten worden constant bewaakt als onderdeel van continuous testing binnen de CI/CD-pipeline. De naleving van de verwachte kwaliteitskenmerken van het product in productie wordt bewaakt met behulp van observability. Geautomatiseerde quality gates (gebaseerd op benchmarks) worden vaak in verschillende fasen van de DevOps-pipeline geïmplementeerd om de release te blokkeren wanneer de productkwaliteit niet aan de verwachtingen voldoet. Deze quality gates kunnen statistieken beoordelen, zoals het slagingspercentage van tests, performance, dekkingsgraad en resultaten van statische codeanalyse.

Performance en betrouwbaarheid zijn vaak belangrijke niet-functionele kwaliteitskenmerken en maken deel uit van de productkwaliteitsbewaking. Performance tests (bijv. load- en stresstests) kunnen in de pipeline worden geïntegreerd om de performance van het systeem te bewaken ten opzichte van verwachte benchmarks. Notificaties worden verstuurd wanneer de benchmarks worden overschreden, zodat teams problemen direct kunnen aanpakken. Ook (log) monitoringtools kunnen worden gebruikt om realtime foutpercentages, geheugenlekken en systeemcrashes in productie te volgen. Vroegtijdige identificatie van deze problemen kan ervoor zorgen dat het product voldoet aan de betrouwbaarheidsverwachtingen.

De dagelijkse standup is het mechanisme dat wordt gebruikt om vrijwel continu kwaliteitsbewaking uit te voeren. Validaties op de bedrijfswaarde en de productkwaliteit worden uitgevoerd tijdens demo's met stakeholders. De productkwaliteit wordt geverifieerd en gevalideerd aan de hand van de vastgestelde kwaliteitscriteria van de DoD.

Evenals het procesgebied Testplanning zijn specifieke toepassingen voor het bewaken van de entry criteria, opschorting en hervatting waarschijnlijk niet relevant. Raadpleeg voor meer informatie specifieke doelstelling 2 'Opstellen testaanpak' van het procesgebied Testplanning, waarin is uitgelegd waarom deze criteria doorgaans niet relevant zijn in een DevOps-omgeving.

2.3.3 SG3 Managen correctieve acties

DevOps-teams merken onverwachte afwijkingen van het plan snel op in een burn-down grafiek en/of het gebrek aan voortgang van (test)taken en stories op het takenbord. Een verslechterende softwarekwaliteit in productie wordt direct opgemerkt en leidt meestal tot correctieve acties, zoals een rollback naar een stabiele versie. Deze en andere problemen, bijvoorbeeld problemen die de testvoortgang blokkeren, worden gecommuniceerd tijdens de dagelijkse standup, zodat het hele team ervan op de hoogte is. Dit leidt vervolgens tot een teamdiscussie waarin de problemen worden geanalyseerd. Het team beslist samen met de product owner welke correctieve acties genomen moeten worden. Het beheren van deze acties in DevOps-projecten is vooral de verantwoordelijkheid van het zelforganiserende team. Het team kan passende maatregelen afspreken en implementeren, of problemen als 'impediment' escaleren naar de Scrum Master. Correctieve

©2025 TMMi Foundation

acties voor productkwaliteit worden soms geautomatiseerd, en de oorzaken worden gemonitord om herhaling te voorkomen. Dit maakt deel uit van de continue verbetercultuur van het DevOps-team. De Scrum Master is doorgaans verantwoordelijk voor het ondersteunen van het team bij het afhandelen van problemen. Typische momenten om correctieve acties te bespreken en te beheren zijn de dagelijkse standup en Retrospectives. Correctieve acties kunnen worden afgehandeld als 'taken' of 'backlogitems' via de productbacklog of (binnen de sprint) via het takenbord.

2.4 Procesgebied 2.4 Testontwerp en -uitvoering

Het doel van het procesgebied Testontwerp en -uitvoering is het structureren van de testactiviteiten tijdens de testontwerp en testuitvoeringsfase door testontwerpen op te stellen, gebruikmakend van testontwerptechnieken, de testuitvoeringsfase gestructureerd uit te voeren en het managen van de bevindingen tot deze zijn afgesloten.

Hoewel het uiteindelijke doel (“risico’s verkleinen en de software testen”) hetzelfde is bij zowel DevOps als traditionele projecten, verschilt de manier waarop er getest wordt vaak sterk. In DevOps zijn flexibiliteit en snel kunnen inspelen op veranderingen belangrijk. Testanalyse, testontwerp, uitvoering en voorbereiding vinden hier vaak tegelijk en herhaald plaats, ondersteund door snelle feedbackrondes.

Een ander verschil is hoeveel testdocumentatie er wordt gemaakt. In DevOps is die documentatie vaak onderdeel van de geautomatiseerde testcode. Bij handmatig testen worden meestal ervaringsgerichte en foutgerichte technieken gebruikt, als aanvulling op de automatische tests. Handmatig testen kan ook plaatsvinden via crowd testing. Hierbij wordt de software al in productie gebracht, maar alleen beschikbaar gesteld aan een geselecteerde groep gebruikers. Deze groep voert tests uit op basis van instructies of testopdrachten en geeft feedback aan het ontwikkelteam, bijvoorbeeld over gebruiksvriendelijkheid of toegankelijkheid. Ook specificatie-gebaseerde testtechnieken blijven nuttig. Vooral tijdens gezamenlijke sessies waarin gebruikersverhalen worden besproken, kunnen voorbeeld scenario's helpen om de acceptatiecriteria aan te scherpen. In DevOps ligt de nadruk ook meer op white-box technieken, zoals unit- en integratietesten. Daardoor worden technieken zoals statement testing en decision testing vaker gebruikt, net als contract testing voor API's. Een laatste belangrijk verschil is het hogere risico op fouten bij veel veranderingen in de software. Daarom is er in DevOps vaak meer behoefte aan regressietesten, op verschillende niveaus. Idealiter worden deze tests automatisch uitgevoerd binnen de CI/CD-pipeline, in meerdere fasen van het testproces. Er zijn dus veel verschillen in aanpak, maar uiteindelijk draait het bij Testontwerp en Uitvoering altijd om: tests maken, risico's beperken, tests uitvoeren en fouten opsporen.

2.4.1 SG1 Uitvoeren testanalyse en -ontwerp waarbij gebruik wordt gemaakt van testontwerptechnieken

In DevOps werken testen, het analyseren en ontwerpen van testen en het uitvoeren ervan allemaal samen. Deze activiteiten gebeuren meestal voortdurend en zijn onderdeel van de manier van werken in een Agile team. In DevOps-projecten maken testers deel uit van een team dat samen user stories maakt en refined. Tijdens het maken van deze user stories worden regelmatig informele reviews gedaan, ook op de acceptatiecriteria – dit zijn voorwaarden waaraan de user story moet voldoen. Deze voorwaarden worden samen bedacht door mensen uit de business, ontwikkelaars en testers.

Testers kijken met een andere blik naar de user stories. Zij zien vaak details die ontbreken of andere situaties die getest moeten worden. Daardoor kunnen de user stories beter en duidelijker worden gemaakt.

Testanalyse is geen aparte taak, maar iets wat testers automatisch doen tijdens het samenwerken aan de user stories. Op basis van de user stories worden testcondities opgesteld. Deze geven aan wat er getest moet worden. Als de acceptatiecriteria duidelijk genoeg zijn, kunnen die ook dienen als testcondities. In DevOps worden methodes zoals BDD (Behavior Driven Development) en ATDD (Acceptance Test Driven Development) gebruikt. Daarmee worden scenario's beschreven met Gherkin-taal – dit helpt om tests en condities op te stellen. Ook voor niet-functionele eisen worden vaak testcondities als acceptatiecriteria vastgelegd.

De testcondities worden daarna omgezet in testgevallen. Die testgevallen zorgen ervoor dat de afgesproken acceptatiecriteria worden behaald. Belangrijke test condities kunnen ook in testcharters worden gezet voor exploratory testing. Dit geldt niet alleen voor losse user stories, maar ook voor epics en features. Die hebben vaak meer algemene test condities die over meerdere user stories gaan of niet-functionele requirements beschrijven. Testontwerptechnieken helpen hierbij, maar in DevOps worden die meestal op een informele manier gebruikt – testers passen ze toe op basis van hun ervaring.

In DevOps wordt vaak gewerkt met het 'test-first'-principe: tests worden vooraf (of tegelijk met de code) bedacht en soms geautomatiseerd uitgevoerd. Dit kan al starten bij het refinieren van de backlog. Voor het uitvoeren van een geautomatiseerde unit test wordt vaak TDD (Test Driven Development) gebruikt. Voor testen op hoger niveau zijn Behavior-Driven Development (BDD) en Acceptance Test-Driven Development (ATDD) populair, die ook goed passen bij testautomatisering.

Bij handmatig testen worden de tests meestal bedacht en aangepast tijdens het testproces. Ze worden niet tot in detail opgeschreven, maar eerder als test ideeën gebruikt bij exploratory testing. In complexe of risicovolle gebieden kan een meer traditionele testaanpak nodig zijn, maar dan nog is de documentatie vaak beperkt. Tests krijgen prioriteit afhankelijk van het belang van de user story. De prioriteit wordt bepaald op basis van de businesswaarde. Het is belangrijk om tests te hebben voor zowel functionele als niet-functionele risico's. In DevOps is het extra belangrijk om gedegen regressie test uit te voeren. Daarom worden tests toegevoegd aan de CI/CD pipelines. De benodigde testdata wordt bepaald op basis van de testcondities. In tegenstelling tot traditionele projecten wordt de testdata in DevOps niet van tevoren volledig vastgelegd. De data wordt gemaakt zodra de tools of functies beschikbaar zijn, zodat snel gestart kan worden met testen. Bij geautomatiseerde tests moet de data vaak wel vooraf worden klaargezet. Testdata wordt op dezelfde manier beheerd als testomgevingen en kan wanneer nodig worden klaargezet. Er zijn ook tools die productiedata kunnen aanpassen (zoals anonimiseren of maskeren) om aan privacyregels (zoals AVG) te voldoen.

Het is belangrijk om de traceerbaarheid tussen de requirements, test condities en testgevallen te waarborgen. Zo kan het team aantonen dat alle user stories en acceptatiecriteria getest zijn. Hiervoor zijn tools nodig waarin je user stories kunt beheren en nummers kunt geven. Die nummers helpen om de link tussen de user stories en tests te maken. In veel tools kun je in realtime zien of alle tests zijn uitgevoerd. In de CI/CD-pipeline is dit ook zichtbaar via de teststatus en logboeken.

2.4.2 SG2 Uitvoeren testimplementatie

Testimplementatie gaat over alles klaarzetten wat nodig is om met het uitvoeren van tests te beginnen. In DevOps wordt er zo min mogelijk testdocumentatie gemaakt. In plaats daarvan worden geautomatiseerde (regressie) test scripts ontwikkeld en op volgorde van prioriteit gezet. Het voorbereiden van de regressietest begint zo vroeg mogelijk, tegelijk met andere testactiviteiten.

In een DevOps-omgeving worden gedetailleerde stappenplannen voor handmatig testen meestal niet gebruikt. Omdat het team veel kennis heeft, worden tests vaak op een hoog en eenvoudig niveau vastgelegd, bijvoorbeeld als test charters met exploratory testing. Dit is genoeg voor de mensen die de tests uitvoeren, want zij kennen het systeem en de techniek goed genoeg. Ze werken binnen het team en weten dus hoe de code werkt en of de functionaliteit geschikt is. Omdat er vaak veel verandert tussen sprints, zou het maken van uitgebreide testplannen veel werk zijn om bij te houden. Daarom wordt handmatig testen meestal gedaan door middel van exploratory testing. Daarbij maak je geen uitgebreid stappenplan, maar gebruik je test charters van één pagina met ideeën en richtlijnen voor de tester. De meeste DevOps-teams testen geautomatiseerd. Bekende methodes zijn TDD, BDD en ATDD. BDD is op zichzelf geen testautomatisering, maar het helpt wel bij het maken van automatische acceptatietests. De scenario's die je in BDD schrijft (vaak in Gherkin) worden gekoppeld aan test automation frameworks. Met behulp van deze methodes worden testscripts geschreven. Die scripts horen bij de testimplementatie-activiteit. De testscripts dienen tegelijk als testdocumentatie. Ze bevatten genoeg uitleg en volgen goede programmeerstandaarden. Vaak worden testscripts gegroepeerd in een test draaiboek. Deze draaiboeken worden gekoppeld aan test jobs in de CI/CD pipeline.

Specifieke testdata die eerder bij testanalyse en ontwerp zijn bepaald, wordt aangemaakt in een herbruikbaar formaat. Soms wordt test data opgezet als code, bijvoorbeeld via schema-bestanden en regels. Een intake test wordt opgezet in de CI/CD pipeline. Deze test, ook wel confidence of smoke test genoemd, controleert aan het begin van elke testuitvoering of het testobject klaar is voor uitgebreid testen. De smoke test geeft snel feedback aan het ontwikkelteam en andere betrokkenen als er iets mis is. Daarnaast wordt vaak een automatische checklist gemaakt voor de gezondheid van elke testomgeving of fase in de CI/CD pipeline. Taken zoals het voorbereiden van testdata, aanpassen van testomgevingen of het opzetten van testautomatisering worden ingeschat en gepland op het storyboard van het team tijdens de standup meetings.

Wat betreft de specifieke toepassing 2.4 Testontwerp en -uitvoering: Geautomatiseerde testen starten automatisch wanneer er een wijziging is in de code binnen de CI/CD pipeline. Trigger-scenario's worden ingesteld (bijvoorbeeld via web-hooks) om tests automatisch te starten bij veranderingen. Handmatige tests worden ingepland tijdens de sprintplanning, beheerd als taken op het storyboard en besproken in de dagelijkse stand-up meetings.

2.4.3 SG3 Uitvoeren testen

Testen worden uitgevoerd volgens de CI/CD pipeline jobs bij elke wijziging in de code, en voor nieuwe user stories die het team ontwikkelt volgens de planning van de sprint. De volgorde van test jobs is zo bepaald dat je snel feedback krijgt. Als er genoeg capaciteit is, kunnen meerdere test jobs tegelijk worden uitgevoerd, zodat feedback nog sneller komt. Daarbij zijn cloud kosten en duurzaamheid belangrijk om de CI/CD pipeline efficiënt te houden. Defects/incidenten worden gemeld en test logs worden automatisch gegenereerd. In een DevOps-project wordt software zo vaak mogelijk naar productie gebracht, meestal na een wijziging in de code. Testen is een vast onderdeel van dit proces en wordt samen met het team uitgevoerd om een goed product op te leveren. Smoke testen zijn een vast onderdeel van de test jobs in de CI/CD pipeline. Ze worden als eerste uitgevoerd. Als deze mislukken, wordt de pipeline gestopt en krijgt het ontwikkelteam direct feedback. Als de oorzaak een echte fout is, wordt dit gemeld als een defect.

Testen worden uitgevoerd volgens de prioriteiten die zijn vastgesteld tijdens de sprint planning. Sommige testen worden gedaan aan de hand van een gedocumenteerde testprocedure, maar de meeste worden gedaan met exploratory testing of session-based testing. Deze testvormen gebruiken vooraf gemaakte test

charters en zijn gelimiteerd in tijd. Soms worden testen ook in productie gedaan, voordat de software voor alle gebruikers beschikbaar is. Bij crowd testing geven geselecteerde testers (bijvoorbeeld beta-testers) feedback over de kwaliteit van het product. Hiervoor moeten functies aan- of uitgezet kunnen worden met een feature toggle, afhankelijk van voorwaarden zoals gebruikers of locatie.

Bij DevOps is het belangrijk om regressietesten goed te organiseren. Vooral bij unit testen en integratie testen hoort regressietesten bij continuous integration. Dit is een automatisch proces van “build en test” dat start bij elke codewijziging. Het helpt om fouten snel te vinden. Continuous integration zorgt ervoor dat regressietesten regelmatig automatisch draaien en het team snel feedback krijgt over de kwaliteit van de code en hoeveel daarvan is getest (code coverage). Testen worden dus uitgevoerd via de CI/CD pipeline, bij veranderingen in de code. De automatische test scripts leveren resultaten en logs op. Deze logs worden samengevat en laten bijvoorbeeld zien wat de status is per epic, user story, test case, test suite, test job, en voor de hele pipeline.

Verschillen tussen de verwachte en echte resultaten worden als defect gemeld. Defectmanagement is meestal onderdeel van het ontwikkelsysteem, en levert volledige herleidbaarheid naar testuitvoering en code wijzigingen. Testlogs worden automatisch opgeslagen om een overzicht te geven van wat er precies gebeurde tijdens het testen.

Tijdens het testen van nieuwe user stories wordt er in Agile teams vaak besproken of elke gevonden fout moet worden gelogd. Zeker als het team samen op één locatie werkt, praten testers en developers direct met elkaar. Fouten die meteen worden opgelost en in de volgende build zitten, hoeven soms niet te worden gelogd. Sommige teams loggen alleen fouten die later zijn ontdekt, anderen alleen fouten die niet vandaag kunnen worden opgelost, of alleen hoge prioriteit. Als niet alles wordt gelogd, moeten er duidelijke regels zijn over wat wel en niet gelogd moet worden.

Ook in DevOps-teams wordt het als een goede gewoonte beschouwd om data te loggen tijdens de testuitvoering om te bepalen of de geteste items voldoen aan de gedefinieerde acceptatiecriteria en daadwerkelijk als "Done" kunnen worden bestempeld. Testdata logging dient mogelijk te zijn bij de gebruikte automatisering. Dit geldt ook bij het toepassen van ervaringsgerichte technieken zoals exploratory testing. Voorbeelden van nuttige informatie om te documenteren zijn: testdekking (hoeveel is er gedekt en hoeveel moet er nog getest worden), observaties tijdens het testen (bijvoorbeeld: lijken het te testen systeem en de user story stabiel), risicolijst (welke risico's zijn gedekt en welke behoren nog tot de belangrijkste), gevonden defecten en andere problemen, en mogelijke openstaande vragen. De geregistreerde informatie dient te worden vastgelegd en/of samengevat in een vorm van statusbeheertool (bijvoorbeeld testmanagementtools, taakbeheertools, takenbord), op een manier die het voor het team en stakeholders gemakkelijk maakt om de huidige status van alle uitgevoerde tests te begrijpen. Natuurlijk is het niet alleen de status van afzonderlijke testtaken waar het team inzicht in nodig heeft, maar moet het ook de algehele status van de user story kennen. Een backlogitem is voltooid wanneer het voldoet aan de Definition-of-Done. Wanneer een DevOps-team Continuous Deployment als werkwijze hanteert, omvat de Definition-of-Done meestal ook een succesvolle deployment als criterium.

2.4.4 SG4 Managen testbevindingen

De term "incident" in DevOps wordt meestal gebruikt voor een echt probleem in de productie, waarbij gebruikers last hebben. Daarom gebruiken we in de rest van deze paragraaf het woord "defect" om te praten over een testprobleem in de context van TMMi. Omdat DevOps-teams ook verantwoordelijk zijn voor

Operations, moeten ze snel reageren. Incident management binnen DevOps is een proces uit IT Service Management (ITIL). Het betekent: "Het zo snel mogelijk herstellen van de normale werking van de dienst om negatieve gevolgen van incidenten te beperken."

In DevOps is bijna alles geautomatiseerd en in code vastgelegd. Dus elk defect dat gevonden wordt, is in feite een probleem in software, als het om een probleem in de omgeving gaat, is het een fout in de Infra as Code van die omgeving, als het om verkeerde testdata gaat, is het een fout in de test automation code, als het gaat om onbetrouwbare uitvoering, kan het een fout in de pipeline code zijn. Zoals eerder gezegd: in DevOps-omgevingen worden niet alle gevonden defecten geregistreerd. Deze doelstelling geldt alleen voor de defecten die wél worden geregistreerd en dus moeten worden opgevolgd tot ze zijn opgelost. In principe is het beheren van incidenten in DevOps simpel. Als een incident wordt geregistreerd op het task board, wordt het zichtbaar als een defect dat een story en de bijbehorende taken tegenhoudt. De standaard werkwijze in DevOps wordt dan gevolgd, net zoals bij andere blokkades of taken. Het defect wordt besproken tijdens stand-up meetings en krijgt een verantwoordelijke binnen het team die het moet oplossen.

Als besloten wordt om een gevonden defect uit te stellen naar een volgende sprint, dan wordt het gewoon een wens of wijziging voor het product. Zet het op de backlog en geef het een prioriteit. Als de prioriteit hoog genoeg is, pakt het team het op in de volgende sprint.

Het is een goede gewoonte om bij te houden hoeveel defecten er per sprint zijn gevonden en opgelost, en hoeveel er nog openstaan en mogelijk op de backlog van de volgende sprint terechtkomen. Bespreek dit tijdens de dagelijkse stand-up meetings en bij de retrospective meetings.

2.5 Procesgebied 2.5 Testomgeving

Het doel van het procesgebied Testomgeving is het inrichten en onderhouden van een adequate testomgeving, inclusief testdata, waarin het mogelijk is om de testgevallen op een beheersbare en reproduceerbare wijze uit te voeren.

Met het procesgebied Testomgeving wordt er gezorgd voor een goede testomgeving, inclusief algemene testdata, waarin je de tests op een beheersbare en herhaalbare manier kunt uitvoeren. Ook in een DevOps software ontwikkelproject is een goede testomgeving heel belangrijk. Door gebruik te maken van cloud technologieën in elke stap van de CI/CD pipeline, kun je een testomgeving op aanvraag opzetten in een gecontroleerde toestand. Omdat de sprints snel na elkaar komen, moet de testomgeving uitermate stabiel en beschikbaar zijn. Problemen in de testomgeving, bijvoorbeeld in de CI/CD pipeline, hebben direct invloed op de voortgang en resultaten van de sprint. Daarom is het ook heel belangrijk om de configuratie en veranderingen van de testomgeving en de testdata goed te beheren.

2.5.1 SG1 Opstellen testomgevingseisen

Het specificeren van wat er nodig is voor een testomgeving gebeurt vroeg in het project, en blijft daarna doorgaan naarmate we meer leren. Het is belangrijk om goed te begrijpen hoe het product tot stand komt (de value stream), hoe de productieomgeving eruit ziet, welke technische systemen er zijn, en welke ontwikkeltools gebruikt worden. Dit helpt om te bepalen wat voor soort omgeving nodig is, en om eisen op te stellen waarmee de omgeving gebouwd kan worden. Deze requirements worden nagekeken om te zorgen dat ze kloppen, haalbaar zijn, geschikt zijn, en een goede afspiegeling zijn van hoe het er in het echt aan toe gaat. Door vroeg te beginnen met het opstellen van eisen, is er meer tijd om de juiste testomgeving te maken of te regelen. Hierbij wordt vaak gebruikgemaakt van technieken zoals virtualization, containerization en andere

cloud native oplossingen, die vaak bij DevOps worden gebruikt. Omgevingen die in de build pipeline (de CI-kant van de pipeline) worden gebruikt, zijn meestal klein en eenvoudig. Ze worden steeds verder ontwikkeld naarmate er meer eisen komen op het gebied van snelheid en betrouwbaarheid. In de release pipeline (de CD-kant van de pipeline) is soms een volledige test van een bedrijfsproces nodig. Dan is er ook een veel complexere testomgeving nodig.

Wat een testomgeving nodig heeft, wordt besproken en vastgelegd tijdens refinement sessies. Dit gebeurt volgens infrastructure-as-code en configuration-as-code standaarden, zoals beschreven door CI/CD playbooks of een platform team. Hoe belangrijk de eisen zijn voor de testomgeving en data, hangt af van hoe belangrijk het bijbehorende item op de backlog is. Soms is er een apart technisch onderzoek (technical spike) nodig om ingewikkelde eisen te kunnen bepalen. De eisen worden meestal geschreven in programmeertalen die geschikt zijn voor infrastructure/configuration-as-code, en volgen de gewoontes en taal van het CI/CD proces van de organisatie. Deze eisen worden gekoppeld aan delen van de value stream of stappen in de CI/CD pipeline. De ontwikkeltools moeten passen bij de systemen waarin het draait, zodat alles wat nodig is ook echt aanwezig is.

2.5.2 SG2 Implementeren testomgeving

Het managen en beheren van de test- en productieomgevingen gebeurt tijdens het hele DevOps-project. Deze omgevingen worden goed beheerd zodat testen zonder onderbreking kunnen doorgaan. Hiervoor wordt gebruikgemaakt van beschikbare tools om omgevingen en data aan te maken, in te stellen, toe te wijzen en te gebruiken.

Het beheren van systemen in testomgevingen helpt bij het beschikbaar maken, instellen, gebruiken en uiteindelijk weer afsluiten van deze omgevingen. Dit gebeurt met de volgende werkwijzen:

- beheren van infrastructuur met behulp van code, automatisering en monitoring
- toegang regelen tot de testomgeving door inloggegevens te geven
- technische ondersteuning bieden bij problemen die de voortgang van het testen verstoren
- monitoring en telemetrie aanbieden, die kunnen helpen bij het analyseren van testresultaten en het vinden van de oorzaak van fouten

Ook het aanmaken, toewijzen, aanpassen en gebruiken van testdata wordt zo veel mogelijk beheerd en geautomatiseerd, om het testproces te ondersteunen.

Het aanmaken en toewijzen van testomgevingen gebeurt automatisch, goed afgestemd en wordt gemeten, zodat deze altijd beschikbaar zijn en efficiënt gebruikt worden. Zelfbediening of on-demand provisioning van omgevingen (aanzetten/uitzetten) en het monitoren van deze omgevingen – vooral bij omgevingen die langere tijd blijven bestaan – worden toegepast. Problemen die ontstaan bij het aanmaken of gebruiken van testomgevingen worden geregistreerd en opgelost door de DevOps-teams. Incidenten in de testomgeving worden vastgelegd zodra een probleem wordt opgemerkt – dit kan automatisch gebeuren bij mislukte tests of handmatig door gebruikers van de omgeving.

De belangrijkste conclusie voor het onderdeel testomgevingen is dat de specifieke doelstellingen en toepassingen nog steeds geldig zijn en in de basis niet veranderen. Het opzetten van de deployment pipeline en het maken van de ontwikkel- en testomgeving moet zo snel mogelijk beginnen. Zo is er al een eerste versie van de omgeving beschikbaar wanneer de eerste sprint van het project start. Testomgevingen lijken op de echte productieomgeving maar bevatten algemene testdata om de juiste soorten tests uit te voeren binnen de verschillende CI/CD teststappen. Slimme DevOps teams gebruiken dezelfde automatische processen voor de testomgevingen als voor de productieomgevingen. Ze zetten de omgevingen op wanneer dat nodig is (on

©2025 TMMi Foundation

demand). Alle taken die met het maken van de omgeving te maken hebben, moeten op de backlog komen, net als andere user-stories.

Bronnen

- [Baah] A. Baah (2017), Agile quality assurance, Bookbaby
- [Beyer] B. Beyer, C. Jones, J. Petoff and N.R. Murphy (2016). Site Reliability Engineering: How Google Runs Production Systems, 1st. O'Reilly Media, Inc.
- [Cohn] M. Cohn (2009), Succeeding with Agile: Software Development using Scrum, Addison-Wesley
- [DORA16] A. Brown, N. Forsgren, J. Humble, N. Kersten and G. Kim (2016), 2016 State of DevOps Report, Puppet Labs, DORA (DevOps Research & Assessment]
- [DORA24] DORA [2024], DORA's software delivery metrics: the four keys, <https://dora.dev/guides/dora-metrics-four-keys/>, DORA (DevOps Research & Assessment]
- [Veenendaal12] E. van Veenendaal (2012), PRISMA: Product Risk Assessment for Agile Projects, in: Testing Experience, Issue 04/12, December 2012
- [Veenendaal24] Erik van Veenendaal, Rex Black, and Dorothy Graham (2024), Foundations of Software Testing - ISTQB Certification (5th edition), Cengage
- [Wake] B. Wake (2003), INVEST in Good Stories, and SMART Tasks, xp123.com/articles/invest-in-good-stories-and-smart-tasks