

Kwaliteitsplan

**{Projectnaam}**

Rubriceringsniveau {Rubriceringsniveau}

Versie {Versienummer}, {Datum}



Inhoudsopgave

##### Colofon

###### Rubricering

Rubricering conform [VIRBI 2013, art. 4](https://wetten.overheid.nl/BWBR0033507/2013-06-01#Artikel4).

{Verwijder eventueel deze paragraaf en de rubricering op de titelpagina indien rubricering niet van toepassing is}

|  |  |  |
| --- | --- | --- |
| Rubriceringsniveau | Rubriceringsduur | Vaststeller |
| {Rubriceringsniveau} | {Rubriceringsduur} | {Vaststeller van de rubricering: minister, staatssecretaris, secretaris-generaal of een door de secretaris-generaal aangewezen rubriceringsambtenaar} |

###### Revisiehistorie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versie | Auteur | Datum | Status | Opmerkingen |
| {versie} | {naam} | {datum} | {concept/definitief} | {opmerkingen} |

###### Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Functie/rol | Naam | Datum | Versie |
| Kwaliteitsmanager {opdrachtgever} | {naam} | {datum} | {versie} |
| Kwaliteitsmanager {beheerpartij} | {naam} | {datum} | {versie} |
| Kwaliteitsmanager ICTU | {naam} | {datum} | {versie} |

###### Vereiste goedkeuringen

|  |  |  |  |
| --- | --- | --- | --- |
| Functie/rol | Naam | Datum | Versie |
| Projectleider {opdrachtgever} | {naam} | {datum} | {versie} |
| Projectleider {beheerpartij} | {naam} | {datum} | {versie} |
| Projectleider ICTU | {naam} | {datum} | {versie} |
| Product owner | {naam} | {datum} | {versie} |

###### Verzendlijst huidige versie

|  |  |  |
| --- | --- | --- |
| Naam | Organisatie | Functie/rol |
| {naam} | {opdrachtgever} | Projectleider |
| {naam} | {opdrachtgever} | Product owner |
| {naam} | {beheerpartij} | Projectleider |
| {naam} | ICTU | Projectleider |
| {naam} | ICTU | Software delivery manager |

###### Template versie

Versie 4.0.0-dev, 06-06-2023

Verbeterpunten t.a.v. deze template graag melden via [GitHub](https://github.com/ICTU/Kwaliteitsaanpak/issues).

# Managementsamenvatting

ICTU hanteert voor het ontwikkelen van maatwerksoftware de ICTU Kwaliteitsaanpak Softwareontwikkeling. Deze Kwaliteitsaanpak houdt in dat ICTU voor elk softwareproject een aantal standaard maatregelen toepast, min of meer onafhankelijk van de precieze eisen die de opdrachtgever stelt aan de software. Dit kwaliteitsplan geeft een overzicht van deze standaard kwaliteitsmaatregelen. Voor de realisatiefase zijn de belangrijkste maatregelen:

* Het hanteren van expliciete entry-en exitcriteria voor het oppakken en afronden van werk
* Het toepassen van het ICTU-kwaliteitssysteem (Quality-time)
* Het bewaken van de kwaliteit van de broncode op aspecten als complexiteit en omvang
* Het uitvoeren van verschillende testvormen zoals unit testen, systeemtesten, performancetesten, securitytesten, toegankelijkheidstesten, en usabilitytesten
* Het zichtbaar maken en managen van technische schuld
* {Bij toepassing van DevOps:} Het bewaken van de kwaliteit van de applicatie in de operatie op aspecten als performance en resource gebruik

Omdat de opdrachtgever specifieke eisen stelt aan de kwaliteitsaspecten {kwaliteitsaspecten} zullen de volgende extra kwaliteitsmaatregelen worden getroffen om deze eisen te borgen:

* {Projectspecifieke kwaliteitsmaatregel 1}
* {Projectspecifieke kwaliteitsmaatregel 2}

# Inleiding

## Over dit document

Kwaliteitsmanagement is gericht op het borgen dat het projectresultaat voldoet aan de kwaliteitseisen van de opdrachtgever en andere belanghebbenden en aan de kwaliteitsnormen van ICTU. Tegelijkertijd bevordert het de efficiëntie en effectiviteit, en daarmee de doorlooptijd, van het project. Alle betrokkenen in en bij het project dragen bij aan de kwaliteit van producten en processen. De ICTU-projectleider is verantwoordelijk voor het opleveren van een eindresultaat dat voldoet aan de eisen en normen.

De ICTU Kwaliteitsaanpak Softwareontwikkeling (zie bijlagen) ligt ten grondslag aan dit kwaliteitsplan en vereist dat projecten continu voldoen aan de kwaliteitsnormen:

**M02: Het project bewaakt continu dat het product aan de kwaliteitsnormen voldoet**

Projecten bewaken zo snel mogelijk vanaf de start de door het project en ICTU vastgestelde kwaliteitsnormen en voldoen daar zo snel en goed mogelijk aan. De kwaliteit van producten, die nog niet zijn afgerond of nog niet aan de normen voldoen, wordt door het project bewaakt. Het voldoen aan de kwaliteitsnormen is onderdeel van de Definition of Done en herstel van de kwaliteit wordt planmatig opgepakt.

Dit kwaliteitsplan beschrijft in nader detail hoe invulling wordt gegeven aan de maatregelen uit de Kwaliteitsaanpak en benoemt eventuele afwijkingen. Als er bijzondere of hoge niet-functionele eisen zijn, beschrijft het kwaliteitsplan ook de extra projectspecifieke kwaliteitsmaatregelen die het project treft om deze eisen te realiseren. Het kwaliteitsplan wordt gedurende de voorfase van een softwareontwikkelproject opgesteld en heeft betrekking op zowel de voorfase als realisatiefase van het project.

De opdrachtgever en de ICTU-projectleider accorderen dit kwaliteitsplan. Omdat de maatregelen invloed kunnen hebben op de backlog, zal de product owner het kwaliteitsplan ook accorderen.

## Doelgroep

Doelgroep van dit kwaliteitsplan zijn opdrachtgever en product owner van {opdrachtgevende organisatie}, beheerders van {beheerorganisatie} en ontwikkelaars en projectmanagement van ICTU.

## Kaders

De volgende kaders zijn van toepassing op het realisatieproces van het project. Merk op dat kaders die van toepassing zijn op het te realiseren *product* opgenomen zijn in PSA, NFE en/of backlog.

|  |  |
| --- | --- |
| Volgnummer | Kader |
| K01 | NEN-ISO/IEC 27001:2017 en NEN-ISO/IEC 27002:2017 |
| K02 | VIR 2007, VIRBI 2013 en BIO voor het inrichten en beheren van informatiebeveiliging in brede zin. |
| K03 | {Indien van toepassing:} NEN 7510:2017 - Informatiebeveiliging in de zorg. |
| K04 | {Indien van toepassing:} Wbni 2018 - Wet Beveiliging Netwerk- en Informatiesystemen. |
| K05 | ICTU Kwaliteitsaanpak Softwareontwikkeling 4.0.0-dev |

## Uitgangspunten

De volgende uitgangspunten zijn van toepassing op dit document:

|  |  |
| --- | --- |
| Volgnummer | Uitgangspunt |
| U01 | De ontwikkeling van de software wordt uitgevoerd conform de ICTU Kwaliteitsaanpak Softwareontwikkeling. |
| U02 | {uitgangspunt} |
| U03 | {uitgangspunt} |
| {volgnummer} | {uitgangspunt} |

## Relatie met andere documenten

Input voor dit kwaliteitsplan, en dan met name paragraaf 5.16, zijn de volgende documenten:

|  |  |
| --- | --- |
| Titel | Versie |
| Projectstartarchitectuur {projectnaam} | {versie} |
| Niet-functionele eisen {productnaam} | {versie} |
| {titel} | {versie} |

## Leeswijzer

Hoofdstuk 3 beschrijft de kwaliteitsmaatregelen die ICTU treft tijdens alles fases van de uitvoering van dit project. Hoofdstuk 4, 5 en 6 beschrijven de maatregelen voor respectievelijk de voorfase, de realisatiefase en de projectafsluiting. De meeste maatregelen zijn generiek en van toepassing op alle softwarerealisatieprojecten die ICTU uitvoert. Projectspecifieke kwaliteitsmaatregelen die voortvloeien uit de documenten genoemd in paragraaf 1.5 staan beschreven in paragraaf 5.16. {Vul aan en pas verwijzingen aan indien nodig}

Bijlage A bevat afkortingen en termen die voorkomen in de ICTU Kwaliteitsaanpak Softwareontwikkeling en bijbehorende templates. Bijlage B verwijst naar regelmatig gebruikte bronnen. Bijlage C bevat een beknopte samenvatting van de ICTU Kwaliteitsaanpak Softwareontwikkeling.

Bijlage D beschrijft hoe ICTU Jira toepast binnen softwareontwikkelprojecten. Bijlage E bevat een lijst met handmatige controles die de kwaliteitsmanager tijdens het project periodiek uitvoert om de uitvoering van kwaliteitsmaatregelen te waarborgen.

# Kwaliteitsmaatregelen tijdens het project

De onderstaande kwaliteitsmaatregelen zijn van toepassing gedurende het gehele project, waaronder de voorfase en de realisatiefase. De navolgende hoofdstukken beschrijven aanvullende kwaliteitsmaatregelen voor respectievelijk de voorfase, de realisatiefase en de projectafsluiting. De M-nummers verwijzen naar maatregelen uit de ICTU Kwaliteitsaanpak Softwareontwikkeling, zie <https://www.ictu.nl/kwaliteitsaanpak>.

## Projectmanagement

### Wekelijks projectoverleg

Het project houdt een wekelijks intern projectoverleg (IPO) waarin planning, voortgang, kwaliteitsrapportage, acties, risico's en overige relevante zaken worden besproken (M10: Het project kent een wekelijks projectoverleg). Het doel is om informatie uit te wisselen binnen het project, belemmeringen voor de voortgang die het Scrumteam niet zelf kan oplossen te melden en eventueel te besluiten over escalaties buiten het project.

De IPO-agenda bevat minimaal: mededelingen, kwaliteit, voortgang, risico's, acties en rondvraag. Aanwezig zijn in ieder geval de projectleider, de software delivery manager, de kwaliteitsmanager, de Scrummaster en een vertegenwoordiger uit elk Scrumteam.

De kwaliteitsmanager controleert of deze overleggen plaatsvinden.

### Actie- en besluitenlijst

Het project houdt een actie- en besluitenlijst bij. De actie- en besluitenlijst wordt digitaal bijgehouden door de software delivery manager en wordt tijdens het wekelijks projectoverleg besproken. Quality-time, het kwaliteitssysteem van ICTU, bewaakt de actualiteit van de actie- en besluitenlijst.

### Risicomanagement

Het project houdt een risicolog bij. De risicolog wordt digitaal bijgehouden door de software delivery manager en periodiek besproken tijdens het projectoverleg. Iedere projectmedewerker kan risico's melden bij de software delivery manager. De omvang van risico's wordt ingeschat en indien nodig voorzien van een of meer maatregelen. Quality-time bewaakt de actualiteit van het risicolog.

### Werkwijze

Voor de realisatiefase wordt de agile ontwikkelmethode Scrum gebruikt (M05: Het project hanteert een iteratief en incrementeel ontwikkelproces). Voor de beheerfase wordt gebruik gemaakt van de DevOps-principes. Afwijkingen hierop worden alleen toegestaan met goedkeuring van de software delivery manager en kwaliteitsmanager.

De volgende rapportage/escalatielijnen worden gehanteerd indien kwaliteitsnormen niet tijdig worden behaald:

1. De kwaliteitsmanager bespreekt de situatie met de software delivery manager;
2. Indien 1. niet tot resultaat leidt, escaleert de kwaliteitsmanager de situatie naar de ICTU-projectleider;
3. Indien 2. niet tot resultaat leidt, escaleert de kwaliteitsmanager de situatie naar het hoofd van de afdeling ICTU Software Expertise (ISE).

Als ontdekte kwaliteitsproblemen daartoe aanleiding geven, worden het kwaliteitsplan en/of Quality-time uitgebreid met nieuwe maatregelen en metrieken om de problemen in de toekomst te signaleren en te voorkomen. Dat gebeurt ook proactief, bijvoorbeeld naar aanleiding van ervaringen in andere projecten of als er nieuwe tools beschikbaar komen. De projectleiders van opdrachtgever, de beheerpartij en ICTU zorgen er gezamenlijk voor dat de gewenste uitbreidingen worden gerealiseerd.

## Projectdocumenten

### Versiebeheer documenten

Alle documenten die een deliverable van het project zijn, zoals architectuurdocumenten, functioneel ontwerp en installatiehandleidingen, worden in de digitale samenwerkruimte van het project geplaatst. De opgeleverde documenten worden in pdf-formaat opgeslagen en bevatten een versienummer in de naam. De versies van de bronbestanden van deliverables worden opgeslagen in een online werkomgeving en moeten beschikbaar zijn om later aangepast te kunnen worden. Dit project gebruikt {Git, Sharepoint, Samenwerkingsruimte}.

Bij elk formele release moeten de documenten geactualiseerd zijn en formeel opgeleverd worden.

### Documentreview

Alle deliverables worden gereviewd. De auteur van een deliverable zorgt, in overleg met de software delivery manager, dat de juiste reviewers benoemd zijn; hiertoe behoort in ieder geval de kwaliteitsmanager. De auteur van het document zorgt voor een correct versiebeheer van het document.

Reviewers worden uitgenodigd door de auteur van het document en ontvangen de juiste versie van de auteur. De werkwijze wordt in overleg door de software delivery manager en kwaliteitsmanager bepaald.

De opstellers verwerken het commentaar. Vervolgens sturen de opstellers een toelichting op de wijze waarop het commentaar al dan niet is verwerkt naar de reviewers.

# Kwaliteitsmaatregelen voorfase

Het doel van de voorfase is tweeledig: het voorbereiden van de realisatiefase, zodat ICTU verantwoord een projectovereenkomst kan opstellen voor de realisatiefase, en het identificeren van risico’s die van toepassing zijn op de realisatiefase en het verdere verloop van het project.

De opdrachtgever zorgt dat het project bij de start van de voorfase inzicht heeft in de informatie die typisch wordt vastgelegd in een projectstartarchitectuur, business impact analysis en privacy impact assessment. Waar nodig werkt de opdrachtgever de informatie bij tijdens de voorfase en realisatiefase.

Dit kwaliteitsplan wordt opgesteld tijdens de voorfase, maar is tevens al deels van toepassing, in ieder geval aan het eind van de voorfase. Voor de voorfase gelden de onderstaande kwaliteitsmaatregelen.

## Belanghebbenden

De kwaliteit van de deliverables wordt mede bepaald door de verwachtingen van de belanghebbenden. Het is van belang dat alle belanghebbenden zijn geïdentificeerd en hun verwachtingen zijn vastgelegd, geanalyseerd en vertaald naar de eisen voor het te implementeren systeem. De belanghebbenden worden geïdentificeerd in het projectvoorstel voor de voorfase. De eisen aan het te ontwikkelen systeem worden vastgelegd in Backlog en NFE-document.

De tijdens de voorfase geïdentificeerde eisen vormen het startpunt van de opdrachtgever en kunnen gedurende de vervolgfases in overeenstemming met de opdrachtnemer aangepast worden. De product owner vertegenwoordigt gedurende de vervolgfasen de geïdentificeerde belanghebbenden.

## Verwerking eisen

### Functionele eisen

Het programma van eisen en de projectstartarchitectuur, beide opgesteld door de opdrachtgever, zijn de basis voor de op te leveren ICTU-documenten, zoals architectuur en ontwerpdocumenten.

### Niet-Functionele eisen

Niet-functionele eisen aan het te ontwikkelen systeem worden vastgelegd op basis van de projectstartarchitectuur en aan de hand van de NEN-ISO/IEC 25010-standaard in een NFE-document. De ISO-25010-kwaliteitsattributen worden door de belanghebbenden geprioriteerd in een of meer workshops (PRA – zie ook testen). Voor de kwaliteitsattributen worden SMART-eisen geformuleerd.

Niet-functionele eisen voor onderstaande kwaliteitsattributen worden als volgt verwerkt:

* De informatiebeveiligingseisen worden in een afzonderlijk informatiebeveiligingsplan vastgelegd. De software zal zodanig worden voortgebracht en {in geval van DevOps:} beheerd dat deze de BIO-compliance van de opdrachtgever niet zal hinderen.
* Gebruikskwaliteit (usability) is ingebed in de standaard werkwijze van ICTU voor de realisatie van maatwerksoftware. Dit aspect wordt geborgd door opname in het plan van aanpak, het ontwerp en de testplannen.
* Toegankelijkheid is een wettelijke verplichting voor webgebaseerde en mobiele applicaties, zie de EN 301 549 en de WCAG 2.1, niveau A en AA. Toegankelijkheid wordt geborgd via toegankelijkheidstesten, zie de kwaliteitsmaatregelen in paragraaf Toegankelijkheidstesten.
* Performance- en securityeisen worden via performance- en securitytests geborgd, zie de kwaliteitsmaatregelen in paragraaf Testen. Voor de borging van andere niet-functionele eisen moeten projectspecifieke maatregelen getroffen worden. Deze worden in dit kwaliteitsplan opgenomen.
* {In het geval van DevOps:} Eisen aan het operationeel beheer worden vastgelegd als beheerafspraken in het plan van aanpak voor de realisatiefase en geborgd door te rapporteren over de software tijdens het gebruik en over de uitgevoerde beheeractiviteiten.

### Compleetheid deliverables voorfase

In aanvulling op de maatregelen met betrekking tot reviews, zie paragraaf Documentreview, reviewt de kwaliteitsmanager de deliverables van de voorfase op compleetheid. Hiervoor vormt de ICTU Kwaliteitsaanpak Softwareontwikkeling het referentiekader.

### Tracering eisen

{Als de eisen traceerbaar moeten zijn vanuit de Backlog/NFE-document/Informatiebeveiligingsplan via GFO en SAD naar broncode, beschrijf dan hier de wijze waarop de eisen uniek identificeerbaar zijn gemaakt, hoe de relaties tussen eisen en ontwerp(beslissingen) worden bijgehouden en hoe de relaties tussen ontwerp(beslissingen) en code worden bijgehouden. Beschrijf hier ook wie deze relaties op welke momenten verifieert en hoe de verificatie wordt gedocumenteerd.}

# Kwaliteitsmaatregelen realisatiefase

De onderstaande kwaliteitsmaatregelen zijn van toepassing gedurende de realisatiefase.

## Entry- en exitcriteria

ICTU volgt een agile aanpak bij het realiseren van software, waarbij gewerkt wordt in korte iteraties (sprints) waarin user stories worden omgezet in werkende software. De resultaten van één of meer sprints worden gebundeld in (major) releases. Hierbij worden entry- en exitcriteria gebruikt om te bepalen of werk kan worden opgepakt of gereed is: Definition of Ready (entry criteria) en Definition of Done (exit criteria) voor user stories en het vrijgaveadvies voor releases (exit criteria).

### Definition of Ready

Het project definieert en hanteert een Definition of Ready (DoR) voor user stories. Een user story kan pas in een sprint worden opgepakt als deze aan de DoR voldoet. De kwaliteitsmanager controleert of er een DoR is gedefinieerd en controleert periodiek of deze wordt gehanteerd.

De Definition Of Ready van het project bevat de volgende criteria ({vul aan en pas aan}):

1. De beschrijving voldoet aan het user story formaat “als <rol> wil ik <actie> zodat <reden>”;
2. De annotaties en screenshots/prototype zijn gereed;
3. Afhankelijkheden met derden zijn in kaart (bijvoorbeeld: een koppelvlakbeschrijving is beschikbaar);
4. De story is ingeschat door het Scrumteam;
5. De story is goedgekeurd door de product owner;
6. De story is goedgekeurd door de reviewer binnen het Scrumteam;
7. Het verwachte aantal logische testgevallen is ingevuld;
8. Er is ingeschat of de user story mogelijk impact heeft op performance, beveiliging, infrastructuur of andere niet-functionele aspecten. {Zie de bijlage "Gebruik van Jira" voor meer informatie.}

### Definition of Done

Het project definieert en hanteert een Definition of Done (DoD) voor user stories. De kwaliteitsmanager controleert of er een DoD is gedefinieerd en controleert periodiek of deze wordt gehanteerd.

De Definition Of Done van het project bevat de volgende criteria ({vul aan en pas aan}):

1. De broncode voldoet aan de codeerstandaard en is gereviewed,
2. Nieuwe testgevallen zijn gereviewd en geaccordeerd door reviewer,
3. Functionele testen zijn succesvol afgerond,
4. De dekkingsgraad van de functionele testen is minstens 80% (maar meer indien mogelijk),
5. Integratietesten zijn succesvol afgerond,
6. Performancetesten zijn succesvol afgerond,
7. Toegankelijkheidstesten zijn succesvol afgerond,
8. Relevante documentatie, zoals GFO en deploymentdocument, is bijgewerkt,
9. De demo van de user story is voorbereid.

### Vrijgaveadvies

Voor elke major release stelt het project (bijvoorbeeld de testmanager) een vrijgaveadvies op. Eventuele afwijkingen in de kwaliteitsrapportage en hieruit voortvloeiende risico's ("rode metrieken") worden in het vrijgaveadvies toegelicht. De kwaliteitsmanager reviewt het vrijgaveadvies.

Het vrijgaveadvies beschrijft ({vul aan en pas aan}):

1. Welke functionele en niet-functionele eisen in deze release gerealiseerd zijn en hoe dit getoetst en/of getest is,
2. Openstaande bugs,
3. Afwijkingen kwaliteitsrapportage,
4. Overige risico's,
5. Het advies van ICTU wat betreft ingebruikname van de release.

### Release notes

Voor elke release stelt het project release notes op, een overzicht van de wijzigingen in de release. De release notes worden opgesteld door {rol} namens ICTU.

## ICTU-kwaliteitssysteem

Gedurende de realisatiefase gebruikt ICTU Quality-time, het kwaliteitssysteem van ICTU, om automatisch metrieken te meten. Deze metrieken hebben zowel betrekking op het proces als op het product (broncode). De metrieken worden meerdere keer per uur bijgewerkt, zodat er altijd een goed inzicht is in de softwarekwaliteit. Zie het [overzicht van mogelijke metrieken](https://quality-time.readthedocs.io/en/latest/reference.html#metrics) en de daarbij horende standaard normen; de kwaliteitsmanager zal in samenwerking met het realisatieteam de te gebruiken metrieken en normen instellen.

{In het geval van DevOps:} Het project gebruikt {monitoringapplicatie} om het gedrag en de kwaliteit van de applicatie in de operationale situatie te bewaken.

De actuele kwaliteitsrapportage wordt dagelijks besproken tijdens de daily scrum en wekelijks in het intern projectoverleg (IPO). De kwaliteitsmanager onderhoudt en bewaakt de kwaliteitsrapportage.

Ondanks dat het de voorkeur heeft zoveel mogelijk kwaliteitsaspecten van de software en het softwareproces geautomatiseerd te meten is dit niet altijd mogelijk of kosteneffectief. Daarom voert het project self-assessments uit en doet de kwaliteitsmanager periodiek ook handmatige controles. Zie bijlage E.

## Kwaliteit van de broncode

### Codeerstandaard

Het project hanteert de volgende codeerstandaarden:

|  |  |  |
| --- | --- | --- |
| Programmeertaal | Codeerstandaard | Controle |
| {programmeertaal A} | {codeerstandaard A} | {broncodereview} |
| {programmeertaal B} | {codeerstandaard B} | {tool B} |

De keuze van programmeertalen en andere technologie staat beschreven in het SAD.

### Linters, formatters, checkers

Ontwikkelaars in het project gebruiken de volgende tools in hun IDE (Integrated Development Environment) {vul aan/pas aan}:

|  |  |  |
| --- | --- | --- |
| Programmeertaal | Tool | Soort |
| DotNet | ReSharper | Linter |
| DotNet | StyleCop | Linter |
| DotNet | Dotnet-format | Formatter |
| Java | Checkstyle | Linter |
| Java | ErrorProne | Bug checker |
| Java | Google Java Format | Formatter |
| JavaScript | ESLint | Linter |
| JavaScript | JSLint | Linter |
| JavaScript | Prettier | Formatter |
| Python | Pylint | Linter |
| Python | Ruff | Linter |
| Python | Black | Formatter |
| Python | Mypy | Type checker |
| Diverse | SonarLint | Linter |
| {Programmeertaal} | {Tool} | {Soort} |

De configuratie van de tools wordt {wel/niet} gedeeld in de broncode-repository.

### Broncodereviews

Het project hanteert de volgende werkwijze voor broncodereviews:

* Elke pull request wordt door minimaal {X} ontwikkelaar(s) gereviewed.
* Er zijn {Y} goedkeuringen nodig voordat een pull request mag worden gemerged.
* Na goedkeuring wordt de pull request gemerged door de {ontwikkelaar/reviewer}.

Quality-time bewaakt of de reviews hebben plaatsgevonden.

### Complexiteit van broncode limiteren

Om de onderhoudbaarheid van de software te bevorderen dienen methoden en klassen niet te complex te zijn. Het uitgangspunt is dat de cyclomatische complexiteit van individuele methoden, zoals berekend door SonarQube, niet groter is dan 10. De norm is dat 0% van de methoden een te hoge cyclomatische complexiteit hebben. Quality-time bewaakt de complexiteit van de broncode.

### Duplicatie van broncode beperken

Om de onderhoudbaarheid van de software te bevorderen dient er zo min mogelijk duplicatie van broncode aanwezig te zijn. Het uitgangspunt is dat SonarQube met de standaard instellingen 0% duplicatie rapporteert. Quality-time bewaakt de duplicatie in broncode.

### Omvang van het systeem beperken

Om de onderhoudbaarheid van de software te bevorderen dient de totale omvang beperkt te blijven, gemeten in mensjaren herbouwtijd. De SIG/TüVIT-standaard geeft een maximale herbouwtijd en vertaalt deze in een maximum omvang van de software gemeten in regels code. Die maximale omvang verschilt per programmeertaal. Quality-time bewaakt de totale omvang van het systeem.

Voor dit project gelden de volgende normen:

|  |  |
| --- | --- |
| Programmeertaal | Maximale omvang in regels code |
| {programmeertaal A} | {maximale omvang} |
| {programmeertaal B} | {maximale omvang} |

### Omvang van methoden beperken

Om de onderhoudbaarheid van de software te bevorderen dienen methoden niet te groot te zijn. ICTU stelt de norm op maximaal 20 non-comment source statements (NCSS) per methode, zoals gemeten door SonarQube. In het systeem als geheel mag maximaal 0% (afgerond) van de methoden deze norm overschrijden. Quality-time bewaakt de omvang van methoden.

### Omvang van unit interfaces beperken

Om de onderhoudbaarheid van de software te bevorderen, dienen methoden niet te veel parameters te hebben. ICTU stelt de norm op maximaal vijf parameters per methode, zoals gemeten door SonarQube. In het systeem als geheel mag maximaal 0% (afgerond) van de methoden deze norm overschrijden. Quality-time bewaakt de omvang van unit interfaces.

## Versiebeheer broncode

Broncode, zowel van productiesoftware als van geautomatiseerde tests, wordt in een versiebeheersysteem geplaatst. Alleen projectmedewerkers hebben toegang tot de broncode. Quality-time bewaakt of branches worden gemerged met de default branch (meestal *main* of *develop* genoemd).

## Testen

Gedurende de realisatiefase worden de use cases in samenwerking met de product owner vertaald naar user stories. Per user story worden één of meer logische testgevallen opgesteld; per logisch testgeval worden één of meer fysieke testgevallen opgesteld. De user stories, logische testgevallen en fysieke testgevallen zijn expliciet aan elkaar gekoppeld. User stories en logische testgevallen worden vastgelegd in Jira; zie de bijlagen voor de te gebruiken typen en relaties. Afwijkingen op deze standaard aanpak zijn alleen toegestaan met goedkeuring van de kwaliteitsmanager.

Het mastertestplan (MTP) beschrijft welke testsoorten met welke intensiteit voor de realisatie- en beheerfase worden uitgevoerd. Het MTP wordt gebaseerd op in ieder geval PSA, NFE, SAD en GFO. Daarnaast is een product risicoanalyse (PRA) is uitgevoerd als basis voor de vaststelling van de diepgang waarmee de gedefinieerde testsoorten uitgevoerd moeten worden.

## Unit tests

Om de correcte werking van de software te borgen, schrijven ontwikkelaars unit tests. De unit tests draaien als onderdeel van de geautomatiseerde pijplijn en de broncodedekking van de unit tests wordt gemeten. Quality-time rapporteert over het aantal unit tests, het al dan niet slagen van de unit tests en broncodedekking van de unit tests.

De default normen voor broncodedekking met unit tests zijn 90% line coverage en 80% branch coverage. De norm geldt niet voor gegenereerde code, code van derde-partijen die ICTU zelf niet onderhoudt en triviale code zoals getters/setters/framework boilerplate-code.

## Systeemtesten

Om de correcte werking van de software te borgen maken de ontwikkelaars systeemtesten. Deze bestaan uit een combinatie van handmatige eenmalige testen, handmatige regressietesten en automatische regressietesten. De automatische regressietesten draaien als onderdeel van de geautomatiseerde pijplijn en de broncode dekking van de automatische regressietesten wordt gemeten. Quality-time rapporteert over het aantal handmatige regressietesten, de hoeveelheid tijd die het kost de handmatige regressietesten uit te voeren en de laatste datum dat ze zijn uitgevoerd. Quality-time rapporteert ook over het aantal automatische regressietesten, of ze al dan niet falen en de broncodedekking van de geautomatiseerde regressietesten.

De default normen voor broncodedekking met geautomatiseerde regressietesten zijn 80% line coverage en 70% branch coverage. De norm geldt niet voor gegenereerde code, code van derde-partijen die ICTU zelf niet onderhoudt en triviale code zoals getters/setters/framework boiler plate code.

## Unit tests en systeemtesten gecombineerd

Quality-time kan ook de geaggregeerde broncodedekking van unit tests en automatische regressietesten samen rapporteren. In dat geval zijn de default normen voor geaggregeerde broncodedekking 90% line coverage en 85% branch coverage.

Het is, ook als de geaggregeerde dekking gemeten wordt, nog steeds handig te weten welke code de automatische regressietesten (ART) en unit tests elk voor zich raken. Het is aan het project te beslissen welke norm nuttig is ART-dekking en unittest-dekking en of daar überhaupt normen voor nodig zijn.

Het uitgangspunt is dat zoveel mogelijk van de software die ICTU maakt, geautomatiseerd getest wordt en dat daarbij bewuste keuzes zijn gemaakt over de code die niet geautomatiseerd wordt getest. Om die keuzes goed te kunnen maken is het belangrijk dat het deel van de code, dat niet wordt geraakt door geautomatiseerde testen, relatief klein is; daardoor blijft de benodigde hoeveelheid handmatig testwerk beperkt en de risico's van handmatig testen beperkt.

## Performancetesten

Om de performance van de software te borgen voert het project performancetesten uit. Het MTP beschrijft de gekozen aanpak; de performance-eisen zijn vastgelegd in het NFE-document.

ICTU voert drie soorten performancetesten uit, die inzicht geven in de volgende facetten:

1. snelheid van handelingen en voldoen aan eisen (loadtest)
2. stabiliteit op langere termijn (duurtest)
3. schaalbaarheid en maximale belastbaarheid (stresstest).

Deze performancetesten worden uitgevoerd in de {performancetestomgeving}. De loadtest draait dagelijks. De duurtest en stresstest draaien wekelijks.

Quality-time rapporteert over de geautomatiseerde performancetesten. Als de verantwoordelijke tester performancerisico's ontdekt die ook aanwezig zijn in een versie van de software die reeds is opgeleverd, rapporteert de tester deze risico's aan het Scrumteam. Issues die voortkomen uit performancetesten worden opgenomen in Jira met het label "performance\_bevinding".

{Als het project geen DevOps doet:} De testen van ICTU kunnen geen uitsluitsel geven over de uiteindelijke performance in de productie-omgeving: ze geven niet meer dan een relatief resultaat ten opzichte van eerdere testen in dezelfde testomgeving. Toch hanteert ICTU ze als een standaard kwaliteitsmaatregel, vóór de oplevering van een nieuwe versie van de software. Want ze geven het inzicht of de performance voor wat betreft de software geen achteruitgang betekent ten opzichte van de bestaande situatie. De uiteindelijke performance in de productieomgeving dient de opdrachtgever zelf te laten testen.

## Security-testen

De eisen aan de beveiliging worden in de documenten projectstartarchitectuur en niet-functionele eisen gedefinieerd. De in te richten testen dienen aan te tonen dat aan de gestelde beveiligingseisen wordt voldaan.

De geautomatiseerde broncodereviews en rapportages uit Quality-time bevatten diverse metrieken voor beveiligingsaspecten, zoals de OWASP Top-10-criteria. De applicatie wordt gescand met behulp van SonarQube, OWASP dependency checker, OWASP ZAP en OpenVAS.

Om de beveiliging van de software te testen kan deze met enige regelmaat getest worden door een externe partij. Het MTP beschrijft de gekozen aanpak.

Elke beveiligingstest resulteert in een beveiligingstestrapportage met daarin de aangetroffen beveiligingsissues. Issues die voortkomen uit deze testen worden opgenomen in Jira met het label "security\_bevinding". Voor beveiligingsissues gelden de volgende oplostermijnen:

|  |  |  |
| --- | --- | --- |
| Kwalificatie beveiligingsissue | Oplostermijn bij software die wel in productie is | Oplostermijn bij software die niet in productie is |
| Hoog (hoog risico en/of hoge impact) | Zo snel mogelijk, resulteert in een bug fix release | Voor de eerste productierelease |
| Midden (gemiddeld risico en/of gemiddelde impact) | Voor de eerstvolgende major productierelease | Voor de eerste productierelease |
| Laag (laag risico en lage impact) | In overleg met de product owner | In overleg met de product owner |

Quality-time rapporteert of gevonden beveiligingsissues niet te lang open staan.

{Als het project geen DevOps doet:} Merk op: de beveiliging van de software in de acceptatie- en productieomgeving kan niet door ICTU getest worden. Deze test moet de opdrachtgever of de beheerpartij uitvoeren.

## Toegankelijkheidstesten

Om de toegankelijkheid van webapplicaties te testen gebruikt ICTU Axe; Axe is ingericht in de pijplijn. Quality-time rapporteert over de toegankelijkheidsrapportage van Axe. Daarnaast worden handmatige toegankelijkheidstesten uitgevoerd, door deskundige leden van het Scrumteam {of door externe deskundigen}. Het MTP beschrijft de gekozen aanpak voor het testen van WCAG-2.1-richtlijnen die niet geautomatiseerd kunnen worden getest.

## Usability-testen

{Verwijder deze paragraaf indien usability-testen niet van toepassing zijn}

Om de usability van de software te testen worden usability-testen uitgevoerd, door deskundige leden van het Scrumteam {of door externe deskundigen}. Het MTP beschrijft de gekozen aanpak.

Issues die voortkomen uit usability-testen worden opgenomen in Jira met het label "usability\_bevinding".

## Technische schuld

Technische schuld zijn eigenschappen van de software die de lange-termijninzetbaarheid en onderhoudbaarheid van de software bedreigen; denk hierbij aan hoge complexiteit, lage testdekking, ontbrekende testsoorten en ontbrekende documentatie.

Als het Scrumteam of de kwaliteitsmanager constateert dat er technische schuld is, markeert de kwaliteitsmanager deze technische schuld in Quality-time als zodanig om te voorkomen dat de technische schuld ongemerkt verder toeneemt. Vervolgens vraagt de kwaliteitsmanager het Scrumteam, in overleg met de software delivery manager, om de omvang van de technische schuld in te schatten in user-storypunten. Vervolgens wordt een plan gemaakt om de technische schuld in een beheerst tempo - de ontwikkeling/onderhoud van de software moet wel doorgang vinden - weg te werken. Uitgangspunt is ongeveer 10% van de user-storypunten die het Scrumteam normaal in een sprint realiseert; dit kan in principe zonder overleg met de opdrachtgever, omdat het leveren van kwaliteit onderdeel van het werk is.

## Beheer

Ten behoeve van de beheerfase wordt gedurende de realisatiefase een implementatie- en beheerplan opgesteld. Hierin worden de kaders aangegeven op welke wijze het beheer ingericht, geïmplementeerd en uitgevoerd wordt. Hierbij gaat de voorkeur uit naar beheer op basis van de principes van de DevOps-werkwijze.

{Werk dit kwaliteitsplan bij zodra er meer informatie beschikbaar is over de werkwijze ten aanzien van implementatie en beheer}

## Externe testen, toetsen en certificeringen

De volgende externe testen, toetsen en certificeringen zijn gepland:

|  |  |  |
| --- | --- | --- |
| Aspect | Opdrachtnemer | Planning |
| Penetratietest | {Leverancier uit de ICTU-mantel IT-audits } | {Sprint/Kwartaal} |
| Onderhoudbaarheidstoets | {Leverancier uit de ICTU-mantel IT-audits } | {Sprint/Kwartaal} |
| Toegankelijkheidstoets | {Leverancier} | {Sprint/Kwartaal} |
| {Certificering} | {Leverancier} | {Sprint/Kwartaal} |

Een certificeringenplan wordt opgesteld indien het op te leveren systeem aan specifieke certificeringseisen moet voldoen. Dit plan bevat de activiteiten op welke wijze de certificatie wordt uitgevoerd. Eisen voor te behalen certificaten moeten in het PvE en/of NFE-document benoemd zijn; bijvoorbeeld NEN-ISO/IEC 27001:2017 compliancy.

## Projectspecifieke maatregelen

{Verwijder deze paragraaf indien er geen projectspecifieke kwaliteitsmaatregelen nodig zijn}

Deze paragraaf bevat maatregelen voor functionele en niet-functionele eisen, die via de standaardmaatregelen uit de ICTU-kwaliteitsaanpak en in dit kwaliteitsplan niet voldoende worden geborgd.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eis | Omschrijving | Extra maatregelen | Hoe controleren | Verificatie |
| {Nummer} | {voorbeeld: onderdeel A moet kunnen worden vervangen door onderdeel B} | {voorbeeld: scenariotest opnemen in MTP} | {voorbeeld: review MTP} | {voorbeeld: dd/mm/jjjj review MTP, test opgenomen in hoofdstuk x} |
| {Nummer} | {Omschrijving} | {Maatregelen} | {Controle} | {Verificatie} |

# Kwaliteitsmaatregelen projectafsluiting

De kwaliteitsmanager controleert of de projectafsluiting conform de afspraken daarover in het plan van aanpak van de voorfase en/of de realisatiefase is uitgevoerd.

Bijlagen

1. Terminologie en afkortingen

De onderstaande tabel bevat afkortingen en termen die voorkomen in de ICTU Kwaliteitsaanpak Softwareontwikkeling en bijbehorende templates.

|  |  |
| --- | --- |
| Term/afkorting | Toelichting |
| **actor** | een persoon die, of een extern informatiesysteem dat, een handeling verricht op het **informatiesysteem** |
| **API** | application programming interface |
| **ART** | automatische **regressietest** |
| **auditing** | Vastlegging van de door een actor verrichtte handelingen. |
| **authenticatie** | het vaststellen van de identiteit van een **actor** |
| **autorisatie** | aan een **actor** toegekende rechten |
| **BIA** | business impact analysis |
| **BIO** | Baseline Informatiebeveiliging Overheid |
| **broncode** | **software** in een vorm die leesbaar is voor mensen en de intentie van een programmeur uitdrukt |
| **deployment** | installatie van **software** op een systeem waardoor de software beschikbaar wordt gemaakt voor gebruik door **actor**en |
| **developers** | Developers zijn de mensen in het **Scrumteam** die iedere sprint gecommitteerd zijn aan het maken van elk aspect van een bruikbaar increment [Scrumgids] |
| **DevOps** | een praktijk die tot doel heeft **softwareontwikkeling** en **operationeel beheer** samen te brengen |
| **DoD** | definition of done |
| **DoR** | definition of ready |
| **gebruikskwaliteit** | mate waarin een systeem, product of dienst kan worden gebruikt door gespecificeerde gebruikers, voor het bereiken van gespecificeerde doelen, met effectiviteit, efficiëntie en tevredenheid in een gespecificeerde gebruikscontext |
| **GFO** | globaal functioneel ontwerp |
| **IB-plan** | informatiebeveiligingsplan |
| **informatiesysteem** | een samenhangend geheel van gegevensverzamelingen en de daarbij behorende personen, procedures, processen en **programmatuur** alsmede de voor het informatiesysteem getroffen voorzieningen voor opslag, verwerking en communicatie [VIR 2007, NORA] |
| **IPO** | intern projectoverleg |
| **ISD** | ICTU Software Diensten, afdeling van ICTU die **softwareontwikkelprojecten** ondersteunt met ontwikkel- en testomgevingen, tools en diensten |
| **ISE** | ICTU Software Expertise, afdeling van ICTU die **softwareontwikkelprojecten** ondersteunt met expertise op het gebied van **softwareontwikkeling** en die de ICTU Kwaliteitsaanpak Softwareontwikkeling onderhoudt |
| **ISO** | International Organization for Standardization |
| **Jira** | tool om **use cases**, user stories, logische testgevallen en issues vast te leggen |
| **klantreis** | alle directe en indirecte interactie van een klant of gebruiker met een product of dienst |
| **KPI** | key performance indicator |
| **kwaliteitsmanager** | controleert en borgt de kwaliteit van **software** conform de vastgestelde eisen en de Kwaliteitsaanpak en rapporteert aan de **projectleider** |
| **minimum viable product** | de eerste versie van een product of dienst, die zo vroeg mogelijk wordt uitgerold naar de gebruikers; het bevat net voldoende functionaliteit om het gestelde doel te behalen, en niet meer dan dat |
| **MTP** | master testplan |
| **MVP** | **minimum viable product** |
| **NFE** | niet-functionele eis(en) |
| **NORA** | Nederlandse Overheidsreferentie-architectuur |
| **NPR** | Nederlandse Praktijkrichtlijn |
| **ontwikkelaars** | Ontwikkelaars (*developers* in de Scrumgids) zijn de mensen in het **Scrumteam** die iedere sprint gecommitteerd zijn aan het maken van elk aspect van een bruikbaar increment [Scrumgids] |
| **operationeel beheer** | activiteiten die zorgen dat software operationeel is en blijft, zoals het oplossen van incidenten, het uitvoeren van onderhoud, het implementeren van upgrades en patches, het beheren van configuraties, en het monitoren van prestaties en beschikbaarheid |
| **OTAP** | ontwikkel, test, acceptatie, productie; gebruikt om verschillende soorten omgevingen aan te duiden |
| **persona** | een min of meer realistische beschrijving van een fictief persoon, veelal met naam, persoonskenmerken, drijfveren en behoeften, die een groep gebruikers representeert en gebruikt wordt om te redeneren over de gewenste functionele en niet-functionele eigenschappen van de **software** |
| **PIA** | privacy impact assessment |
| **PKI** | public key infrastructure |
| **PRA** | productrisicoanalyse |
| **Product owner** | De product owner is verantwoordelijk voor het maximaliseren van de waarde van het product, dat het resultaat is van het werk van het **Scrumteam** [Scrumgids] |
| **programmatuur** | zie **software** |
| **project** | een tijdelijke organisatie voor het realiseren van een resultaat - bij ICTU bestaat een **softwareontwikkelproject** uit medewerkers van ICTU, opdrachtgever, beheerorganisatie en eventueel andere partijen |
| **projectleider** | medewerker eindverantwoordelijk voor het projectresultaat - bij ICTU-softwareontwikkelprojecten is de projectleider een medewerker van ICTU |
| **PSA** | projectstartarchitectuur |
| **PvE** | programma van eisen |
| **Quality-time** | een door ICTU ontwikkeld, open source, geautomatiseerd kwaliteitssysteem |
| **realisatiefase** | fase van een **softwareontwikkelproject** waarin de **software** daadwerkelijk wordt gebouwd en onderhouden, en bij een **DevOps** werkwijze ook operationeel wordt beheerd |
| **regressietest** | test die na een wijziging controleert of niet-gewijzigde delen van een systeem nog steeds correct functioneren |
| **release notes** | een overzicht van de wijzigingen in een **release** |
| **release** | een voor gebruik vrijgegeven versie van de **software** |
| **SAD** | software-architectuurdocument |
| **Scrum** | Scrum is een lichtgewicht raamwerk dat mensen, teams en organisaties helpt om waarde te creёren door middel van adaptieve oplossingen voor complexe problemen [Scrumgids] |
| **Scrummaster** | De Scrummaster is verantwoordelijk voor het opzetten van **Scrum**, zoals staat beschreven in de Scrumgids [Scrumgids] |
| **Scrumteam** | Een Scrumteam bestaat uit één **Scrummaster**, één **product owner** en **ontwikkelaars** (*developers* in de Scrumgids) [Scrumgids]. |
| **software delivery manager** | organiseert het ontwikkelen en opleveren van **software** conform de vastgestelde eisen en de Kwaliteitsaanpak en rapporteert aan de **projectleider** |
| **software** | software is de verzameling instructies die bepalen wat een computer uitvoert en is uiteindelijk wat de gebruiker ziet, ervaart en waarmee hij interacteert. |
| **softwareontwikkeling** | een activiteit die nieuwe **software** maakt en/of bestaande software aanpast |
| **softwareontwikkelproject** | een **project** dat de oplevering van **software** als enige of voornaamste projectresultaat heeft |
| **technische schuld** | eigenschappen van de **software** die de lange-termijninzetbaarheid en onderhoudbaarheid bedreigen |
| **TVA** | threat and vulnerability assessment |
| **usability** | gebruiksvriendelijkheid |
| **use case** | een afgebakende eenheid van interactie tussen een **actor** en het systeem |
| **UX** | user experience |
| **VIR** | Voorschrift Informatiebeveiliging Rijksdienst |
| **VIRBI** | Voorschrift Informatiebeveiliging Rijksdienst Bijzondere Informatie |
| **VM** | virtual machine, virtuele machine |
| **voorfase** | fase van een **softwareontwikkelproject**, voorafgaande aan de **realisatiefase**, waarin de uitgangspunten, risico's en randvoorwaarden voor de realisatiefase worden bepaald en waarin wordt gezorgd dat aan de randvoorwaarden wordt voldaan en dat voor zoveel mogelijk risico's maatregelen getroffen zijn |
| **vrijgaveadvies** | advies om een **release** vrij te geven, met een testverslag dat tenminste alle nog openstaande testbevindingen en geconstateerde beveiligingsbevindingen bevat |

1. Bronnen

De onderstaande tabel verwijst naar regelmatig gebruikte bronnen.

|  |  |
| --- | --- |
| Bron | Toelichting |
| [BIO](https://bio-overheid.nl/media/1572/bio-versie-104zv_def.pdf) | Baseline Informatiebeveiliging Overheid. |
| [ISO 9241-210:2019](https://www.iso.org/standard/77520.html) | Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems. |
| [NCSC ICT-beveiligingsrichtlijnen voor webapplicaties](https://www.ncsc.nl/documenten/publicaties/2019/mei/01/ict-beveiligingsrichtlijnen-voor-webapplicaties) | De ICT-beveiligingsrichtlijnen voor webapplicaties geven een leidraad voor veiliger ontwikkelen, beheren en aanbieden van webapplicaties en bijbehorende infrastructuur. |
| [NEN-ISO/IEC 25010:2011](https://www.nen.nl/nen-iso-iec-25010-2011-en-157265) | Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. |
| [NEN-ISO/IEC 27001:2017](https://www.nen.nl/nen-en-iso-iec-27001-2017-a11-2020-nl-265545) | Informatietechnologie - Beveiligingstechnieken - Managementsystemen voor informatiebeveiliging - Eisen |
| [NEN-ISO/IEC 27002:2017](https://www.nen.nl/nen-en-iso-iec-27002-2017-nl-245390) | Informatietechnologie - Beveiligingstechnieken - Praktijkrichtlijn met beheersmaatregelen op het gebied van informatiebeveiliging |
| [NEN 7510:2017](https://www.nen.nl/nen-7510-1-2017-a1-2020-nl-267179) | Informatiebeveiliging in de zorg. |
| [NEN NPR 5325:2017](https://www.nen.nl/npr-5325-2017-nl-238298) | Praktijkrichtlijn voor het overdragen van software. |
| [NEN NPR 5326:2019](https://www.nen.nl/npr-5326-2019-nl-262885) | Praktijkrichtlijn voor risicobeheersing bij softwareontwikkeling. |
| [NORA](https://www.noraonline.nl) | Referentiearchitectuur voor de Nederlandse Overheid. |
| [OWASP Top-10](https://owasp.org/www-project-top-ten/) | De OWASP Top-10 is een op consensus gebaseerd overzicht van de meest kritische beveiligingsrisico's voor webapplicaties. |
| [Scrumgids](https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Dutch.pdf) | De Scrum Gids - De Definitieve Gids voor Scrum: De Regels van het Spel. |
| [VIR 2007](https://wetten.overheid.nl/BWBR0022141/2007-07-01) | Besluit Voorschrift Informatiebeveiliging Rijksdienst 2007. |
| [VIRBI 2013](https://wetten.overheid.nl/BWBR0033507/2013-06-01) | Besluit Voorschrift Informatiebeveiliging Rijksdienst Bijzondere Informatie 2013. |
| [Wbni 2018](https://wetten.overheid.nl/BWBR0041515/2020-07-15) | Wet Beveiliging Netwerk- en Informatiesystemen. Beschrijft de meldplicht en de zorgplicht die van toepassing zijn op organisaties die vitaal zijn én op digitale dienstverleners. |

1. De ICTU Kwaliteitsaanpak Softwareontwikkeling

De overheid is in hoge mate afhankelijk van informatiesystemen voor de uitvoering van haar taken. Veel van die informatiesystemen zijn dusdanig specifiek dat de benodigde software “op maat” gemaakt moet worden. De totstandkoming van op maat gemaakte software is meestal een complex proces, waarin vele belangen en behoeften worden afgewogen en afgezet tegen de mogelijkheden die technologie biedt. Eenmaal operationeel zal een informatiesysteem verantwoord onderhouden moeten worden; behoeften en technologie veranderen in de loop van de tijd.

Overheidsprojecten waarin software wordt ontwikkeld of onderhouden kampen nog vaak met vertraging, budgetoverschrijding of een eindresultaat met te lage kwaliteit. Zo concludeerde de commissie-Elias in haar [eindrapport](https://www.tweedekamer.nl/sites/default/files/field_uploads/33326-5-Eindrapport_tcm181-239826.pdf): "De Rijksoverheid heeft haar ICT (Informatie- en communicatietechnologie)-projecten niet onder controle". Eén van de fundamentele problemen is dat de risico's, die inherent zijn aan softwareontwikkeling, door organisaties nog onvoldoende worden herkend, erkend en gemitigeerd. Dit terwijl de risico's bij de ontwikkeling van software, binnen het ICT-domein, algemeen bekend zijn en er ook voor veel risico's passende maatregelen bestaan.

ICTU heeft jarenlange ervaring met het realiseren van software en past de opgedane ervaring toe bij de ontwikkeling van nieuwe software. Die ervaring is vastgelegd in een werkwijze, deze “ICTU Kwaliteitsaanpak Softwareontwikkeling”, die telkens wordt aangepast en aangevuld op basis van de praktijk.

ICTU is ervan overtuigd dat het bouwen van duurzame software, die goed aansluit bij de behoeften van gebruikers en andere belanghebbenden, bijdraagt aan betere informatiesystemen en een betere dienstverlening door de overheid. Dienstverlening die betrouwbaar moet zijn voor burgers, bedrijven en ambtenaren. Om samen met opdrachtgevers passende oplossingen te realiseren ontwikkelt ICTU daarom software volgens een agile proces. En om de duurzaamheid en betrouwbaarheid te bevorderen besteedt ICTU standaard aandacht aan beveiliging, privacy, performance, gebruikskwaliteit en toegankelijkheid. De Kwaliteitsaanpak dient daarvoor als leidraad, maar de aanpak voorziet ook in mogelijkheden om het project en het eindproduct aan te passen aan de specifieke situatie.

Om projecten, die software realiseren volgens de Kwaliteitsaanpak, efficiënt en effectief te ondersteunen, heeft ICTU twee gespecialiseerde afdelingen in het leven geroepen. Deze afdelingen staan projecten bij door middel van kennis, menskracht en technische hulpmiddelen. Zo profiteren projecten van schaalgrootte en hergebruik van inzichten.

Met behulp van de ICTU Kwaliteitsaanpak Softwareontwikkeling heeft ICTU samen met andere overheden inmiddels enige tientallen projecten succesvol uitgevoerd. ICTU wil deze aanpak graag aanvullen met de ervaringen en geleerde lessen van andere organisaties en deze overdraagbaar maken en breder uitdragen. Om die reden stelt ICTU deze Kwaliteitsaanpak aan iedereen beschikbaar via <https://www.ictu.nl/kwaliteitsaanpak> en heeft zij, samen met normalisatie-instituut NEN en partijen uit overheid en markt, een praktijkrichtlijn “Risicobeheersing bij ontwikkeling en onderhoud van maatwerksoftware” [NEN NPR 5326:2019] gepubliceerd, die mede is gebaseerd op de ICTU Kwaliteitsaanpak Softwareontwikkeling.

De ICTU Kwaliteitsaanpak Softwareontwikkeling heeft drie doelstellingen:

1. Opdrachtgevers helpen bekende risico's bij softwareontwikkeling, zoals technische schuld, vertraging en defecten, zo veel mogelijk te voorkomen.
2. ICTU helpen om software te ontwikkelen die de missie van ICTU, namelijk bijdragen aan een betere digitale overheid, ondersteunt.
3. De overheid als geheel helpen bij het zo goed mogelijk ontwikkelen van software.

De Kwaliteitsaanpak zelf is geformuleerd in de vorm van maatregelen die elke software-ontwikkelende organisatie kan treffen om risico's van softwareontwikkeling te mitigeren en de kans op succesvolle softwareontwikkelprojecten te vergroten. De maatregelen zijn gebaseerd op geleerde lessen uit de praktijk van ICTU.

De Kwaliteitsaanpak is een evoluerende aanpak, gebaseerd op de ervaringen die ICTU continu opdoet in de projecten waarin ICTU samen met opdrachtgevers maatwerksoftware ontwikkelt en onderhoudt. ICTU hanteert daarbij de vuistregel dat als tenminste 80% van de projecten minstens 80% van de tijd een bepaalde werkwijze hanteren, voor die werkwijze een maatregel in de Kwaliteitsaanpak wordt opgenomen. Maar het kan ook voorkomen dat maatregelen om andere redenen landen in de Kwaliteitsaanpak; denk aan het toegankelijk maken van software dat wettelijk verplicht is.

De maatregelen vormen het startpunt voor de aanpak van ieder ICTU-softwareproject, waarbij ruimte wordt geboden voor variatie of alternatieve invulling. Bijvoorbeeld stelt de Kwaliteitsaanpak: software wordt minimaal bij iedere grote release of tenminste twee keer per jaar onderworpen aan een beveiligingstest door beveiligingsexperts die ICTU daarvoor inhuurt (zie M26: Het project laat de beveiliging van het ontwikkelde product periodiek beoordelen). Een alternatief is dat de opdrachtgever de verantwoordelijkheid neemt voor het laten uitvoeren van beveiligingstests. Hierover maakt de projectleider nadere afspraken met de opdrachtgever.

De Kwaliteitsaanpak is dus zowel voorschrijvend als beschrijvend. Voorschrijvend omdat ICTU verwacht dat projecten die maatwerksoftware ontwikkelen en onderhouden de aanpak toepassen, en alleen aanpassen als daar een goede reden voor is, en mits dat wettelijk is toegestaan. Tegelijkertijd is de aanpak beschrijvend omdat de meeste maatregelen voortkomen uit de bestaande werkwijzen van de projecten. Zoals blijkt uit de self-assessment die ICTU regelmatig uitvoert op de toepassing van de Kwaliteitsaanpak.

1. Gebruik van Jira

Gedurende de realisatiefase wordt voor de vastlegging van diverse objecten, zoals user stories en logische testgevallen, gebruik gemaakt van Jira. Scrumteams registreren specifieke informatie gedurende de gehele projectduur. Deze bijlage beschrijft welke informatie vastgelegd wordt in Jira, welke Jira-issuetypen hierbij gebruikt worden en wat de onderlinge relaties zijn. Het project richt hiervoor haar eigen Jira-project in.

1. Jira-typen

De gebruikte Jira-typen ondersteunen verschillende inzichten in het project:

**Systeembeschrijving**: De te gebruiken typen geven gezamenlijk de actuele situatie weer van de te ontwikkelen of aan te passen applicatie. Informatie wordt vastgelegd over de use cases en logische testgevallen in respectievelijk de Jira-typen *Use Case* en *Logical Test Case*. Zie nadere toelichting hieronder.

**Realisatie-activiteiten**: De te gebruiken typen geven gezamenlijk inzicht in de realisatie-activiteiten: te plannen, onderhanden en/of afgeronde activiteiten. Informatie wordt vastgelegd over epics, stories en onderhanden taken in respectievelijk de Jira-typen *Epic*, *Story* en *Technical Task*. Zie nadere toelichting hieronder.

**Bugs**: Bugs gevonden na oplevering in de acceptatieomgeving worden geregistreerd. Dit kunnen ook bugs zijn uit de productieomgeving die opgelost moeten worden in de applicatie. De informatie over bugs wordt vastgelegd met het Jira-type *Bug*. Zie nadere toelichting hieronder.

1. Workflow

Elk Jira-type maakt gebruik van een standaard workflow. Omdat de kwaliteitsrapportage gebruik maakt van deze statussen, is het afwijken van deze stadia ongewenst. Binnen het project worden afspraken gemaakt op welke wijze deze Jira workflow in het realisatie proces wordt geïntegreerd. Mogelijke statussen zijn:

* Status Open: Deze status wordt bereikt na een *Create*- of *Stop Progress*-actie;
* Status In Progress: wordt bereikt na de actie *Start Progress*;
* Status Resolved: wordt bereikt na de actie *Resolve Issue*;
* Status Closed: wordt bereikt na het afsluiten van een issue (*Close Issue*);
* Status Reopened: wordt bereikt na de actie *Reopen Issue*.

1. Systeembeschrijving

Het op te leveren systeem is beschreven in documenten zoals globaal functioneel ontwerp (GFO) en software-architectuurdocument (SAD). De actuele situatie van het systeem kan anders zijn in situatie waarin gefaseerd het gewenste systeem wordt opgeleverd of waarin het systeem door de agile manier van werken reeds vooruitloopt op het GFO. Om de actuele situatie te beschrijven wordt gebruik gemaakt van use cases. Logische testgevallen beschrijven hoe de use cases worden getest.

Use cases

De te realiseren functionaliteit wordt met behulp van de user stories geïmplementeerd. De delen van de systeemfuncties die gerealiseerd zijn, worden met use cases door het Scrumteam vastgelegd (meestal een functioneel ontwerper) door issues aan te maken van het type *Use Case*. De verzameling use cases beschrijft de functionele as-is situatie van het systeem.

Logische testgevallen

Logische testgevallen worden door de Scrumteams (normaal gesproken de tester) vastgelegd in Jira door een issue te maken van het type *Logical Test Case* (LTC). Het logisch testgeval wordt middels een Jira link van het type *Tests* gekoppeld aan zowel de use case als de user story die hiermee getest wordt.

Het logische testgeval zelf wordt beschreven volgens het Given/When/Then formaat. De Jira-issue heeft hiervoor drie velden:

* De *Given* van een logisch testgeval beschrijft welke niet-triviale informatie aanwezig wordt verondersteld of in welke context een gebruiker zich bevindt. Bijvoorbeeld: "Gegeven een afgesloten inspectierapport" of "Gegeven een medewerker die zich net heeft geregistreerd". Context die vanzelfsprekend is, bijvoorbeeld dat een gebruiker is ingelogd, hoeft niet expliciet te worden opgeschreven.
* De *When* van een logisch testgeval beschrijft welke actie de gebruiker doet. Bijvoorbeeld: "Als de inspecteur het afgesloten inspectierapport heropent" of "Als de medewerker zijn registratie bekijkt". Passief taalgebruik ("een rapport wordt geopend") is niet toegestaan, omdat dan niet duidelijk is wie de actie doet. Let ook op dat het testgeval logisch is, dat wil zeggen, geen user interface elementen beschrijft. Dus niet "Als de gemeentemedewerker op het dropdown menu klikt", maar "Als de gemeentemedewerker een type kinderopvang kiest".
* De *Then* van een logisch testgeval beschrijft hoe het systeem reageert op de actie van de gebruiker, met een focus op datgene wat het testgeval beoogt te testen. Bijvoorbeeld: "Dan toont het systeem het inspectierapport met als startdatum de datum van vandaag" of "Dan toont het systeem de registratie van de gebruiker en dat aantal inlogpogingen 0 is".

Logische testgevallen worden als geautomatiseerd (Automated), handmatig (Manual) of eenmalig te testen (Will not execute) gemarkeerd. Geautomatiseerd betekent dat fysieke testgevallen worden opgenomen in de automatische regressietest (ART) van het project. Handmatig betekent dat het logische testgeval elke sprint handmatig zal worden getest door de testers. Eenmalig betekent dat de tester eenmalig handmatig het logische testgeval zal uitvoeren. In principe dienen alle logische testgevallen te worden geautomatiseerd, tenzij er goede redenen zijn om dat niet te doen, bijvoorbeeld omdat het technisch niet mogelijk is het testgeval te automatiseren. Eenmalige testen doen we bij triviale wijzigingen zoals het aanpassen van een label of de layout van een scherm.

1. Realisatie-activiteiten

Gedurende de realisatie sprints worden user stories uitgevoerd. Omdat stories slechts een klein deel van gewenste functionaliteiten bevatten (om story punten niet te hoog te laten zijn), worden epics als containers gedefinieerd. Indien een story wordt opgepakt door het Scrumteam worden sub-taken gedefinieerd door middel van het Jira type Technical Task. Wanneer gedurende de ontwikkeling issues vastgelegd moeten worden, kan dit met behulp van het type *Custom Issue*.

Epic

Epics zijn ‘brokken’ functionaliteit die door de user stories worden geïmplementeerd. Ze worden door de Scrumteams (meestal de functioneel ontwerper) vastgelegd in Jira door een issue te maken van het type *Epic*. Een epic wordt middels een Jira link van het type *Realizes* aan de use case gekoppeld waarvoor functionaliteiten worden geïmplementeerd.

User story

User stories worden door de Scrumteams (meestal de product owner of een functioneel ontwerper) vastgelegd in Jira door een issue te maken van het type *Story*. Een story wordt middels een Jira link van het type Changes aan de use case gekoppeld waarvoor functionaliteiten worden geïmplementeerd, en middels het veld Epic link aan de epic die gerealiseerd wordt.

De user story zelf wordt beschreven in het formaat: "Als <rol> wil ik <actie> zodat <rationale die duidelijk maakt wat de business waarde is>". Voorbeelden zijn: "Als medewerker van ICTU wil ik een parkeerplaats voor een bezoeker kunnen reserveren zodat deze niet op zoek hoeft naar een parkeerplaats" of "Als aankomend medewerker in de kinderopvang wil ik mijn VOG registreren in het register voor medewerkers in de kinderopvang omdat ik anders niet mag werken in de kinderopvang". De tekst van de user story dienst in het description veld van het issue te worden vastgelegd zodat de user story tekst goed in de rapportages komt.

Bij elke user story kan worden vastgelegd wat het risico van de verandering is op verschillende aspecten zoals planning, performance en security. Dit maakt het mogelijk om user stories te filteren op, bijvoorbeeld, hoog risico voor security en die lijst als input voor een securitytest te gebruiken.

Technical tasks

Gedurende de realisatie van een user story worden door het Scrumteam diverse activiteiten uitgevoerd. Om de sprint voortgang eenvoudiger te kunnen monitoren, wordt gebruik gemaakt van sub-taken binnen een user story. Deze kunnen automatisch aangemaakt worden (aanvragen via Jira), of handmatig door het Scrumteam vastgelegd worden door *Create Sub-task* type *Technical Task*. Een technical task heeft het formaat <werkwoord> <onderwerp>. Voorbeelden zijn: Opstellen logische testgevallen, Review testgevallen, Ontwikkelen <module>, Uitvoeren handmatige testen, Ontwikkelen ART, Controleren kwaliteitsrapportages, Bijwerken use cases, Check code kwaliteit en testcoverage, etc.

Bugs

Bugs zijn afwijkingen tussen verwacht gedrag en actuele situatie die is gedetecteerd. Het kunnen bugs zijn die gevonden worden tijdens acceptatietesten of productieverstoringen. De bugs worden vastgelegd door Jira issues aan te maken van het type *Bug*.

De bug moet bij registreren alle informatie bevatten die nodig is om de geconstateerde afwijking, gebruikte omgeving en situatie te beschrijven, de prioriteit i.r.t. impact en urgentie, specifieke labels die gebruikt worden in de kwaliteitsrapportage om type bugs te kunnen onderkennen (bijvoorbeeld Security, Performance), en de referentie naar het gebruikte testgeval i.g.v. testen.

* Blocker: applicatie of bedrijfskritische functies/processen kunnen niet gebruikt worden;
* Critical: bedrijfskritische functies/processen worden negatief beïnvloed en er is geen workaround mogelijk;
* Major: critical, maar workaround mogelijk;
* Minor/Trivial: raakt geen bedrijfskritische functies/processen.

Indien de bug opgelost gaat worden, zal er een Jira link van het type *Is realized by* gelegd worden naar die betreffende user story waarin de bug wordt opgepakt; middels comments wordt informatie toegevoegd over de analyse, de alternatieven en de uiteindelijk gekozen oplossing.

1. Periodieke handmatige controles

Ondanks dat het de voorkeur heeft zoveel mogelijk kwaliteitsaspecten van de software en het softwareproces geautomatiseerd te meten is dit niet altijd mogelijk of kosteneffectief. Daarom voert de kwaliteitsmanager periodiek onderstaande controles handmatig uit.

{Pas onderstaande lijst van controles aan waar nodig. Voeg indien relevant kolommen toe voor frequentie, uitvoerder, datum laatste controle, status, vervolgacties, datum volgende controle, etc.}

|  |  |  |
| --- | --- | --- |
| Onderwerp | Referentie | Controle |
| Verwerking informatiebeveiligingsplan | M01 | Actuele beveiligingsmaatregelen zijn verwerkt in SAD, GFO en kwaliteitsplan |
| Traceerbaarheid functionele eisen | M03 | Functionele functionele eisen zijn traceerbaar naar logische en fysieke testgevallen |
| Traceerbaarheid niet-functionele eisen | M03 | Niet-functionele eisen zijn traceerbaar naar SAD, MTP en detailtestplannen |
| Bewijs realisatie niet-functionele eisen | Niet-functionele eisen | Het bewijs dat aan ICTU toegewezen niet-functionele eisen zijn gerealiseerd is actueel |
| Implementatie nieuwe versie Kwaliteitsaanpak | M28 | De meest recente versie van de Kwaliteitsaanpak is geïmplementeerd in het project |
| Gebruik tools | M16 | Het project gebruikt de geadviseerde tools |
| Actualiteit tools | M16 | Het project gebruikt actuele versies van tools |
| Uitvoering self-assessment | M28 | Het project heeft recent een self-assessment uitgevoerd waarin de verschillende maatregelen ook inhoudelijk zijn beoordeeld |
| Actualiteit self-assessment | M28 | De kwaliteitsrapportage bevat een metriek voor de actualiteit van de self-assessment |
| Versiebeheer van documenten | Kwaliteitsplan §3.2.1 | Documenten zijn met de juiste meta-informatie op de afgesproken plaats en wijze opgeslagen |
| Review van documenten | Kwaliteitsplan §3.3.2 | Documenten zijn op de afgesproken wijze gereviewd |
| Definition of Ready | Kwaliteitsplan §5.1.1 | Scrumteams hebben en gebruiken een DoR |
| Definition of Done | Kwaliteitsplan §5.1.2 | Scrumteams hebben en gebruiken een DoD, waarin het voldoen aan de kwaliteitsnormen is opgenomen |
| Vrijgaveadvies | Kwaliteitsplan §5.1.3 | Het vrijgaveadvies bevat de afgesproken onderdelen |
| Kwaliteit broncode | Kwaliteitsplan §5.3 | Onderdrukte SonarQube-waarschuwingen zijn afgestemd en (nog steeds) terecht |
| Kwaliteit broncode | Kwaliteitsplan §5.3 | De kwaliteit van de broncode voldoet bij een steekproefsgewijze, handmatige inspectie aan de eisen |
| Beveiliging broncode | Kwaliteitsplan §5.10 | Onderdrukte beveiligingsbevindingen uit de OWASP Dependency Check en SonarQube zijn afgestemd en (nog steeds) terecht |
| Beveiligingstesten | Kwaliteitsplan §5.10 | Beveiligingsissues uit securitytesten zijn opgenomen in Jira en volgens de oplostermijnen opgelost |
| Projectspecifieke kwaliteitsmaatregelen | Kwaliteitsplan §5.16 | Projectspecifieke maatregelen zijn uitgevoerd zoals afgesproken |
| Projectafsluiting | Kwaliteitsplan §6 | Projectafsluiting is conform de afspraken uitgevoerd |
| Gebruik van Jira | Kwaliteitsplan bijlage D | Jira wordt gebruikt zoals afgesproken |
| Actualiteit kwaliteitsplan |  | De kwaliteitsrapportage bevat een metriek voor de actualiteit van het kwaliteitsplan |
| Nieuwe teamleden |  | Nieuwe teamleden zijn op de hoogte van de Kwaliteitsaanpak, het kwaliteitssysteem (Quality-time) en het kwaliteitsplan |
| Vertrokken teamleden |  | Alle rechten in tools (GitLab, GitHub, Trello, Sharepoint, VPN, Jira, Signal, Slack, etc.) van vertrokken teamleden zijn ingetrokken |