**Deelproduct 5e**

**Integratietest**

**Beroepsproduct 4**

*Blok 4*

*Adam Oubelkas (2150912)*

Avans Academie Associate degree Informatica

**1. Inleiding**

Het eindproduct van een ICT-project moet zorgvuldig mogelijk worden getest. Dit kan pas als het eindproduct getoetst wordt op wensen en eisen van zowel de klant als de stakeholders. Dit gebeurt in de vorm van een toets matrijs waarin het eindproduct wordt beoordeeld op criteria over de tussen de klant en ontwikkelaar opgestelde Requirements [2].

Het eindproduct moet dan als het ware, via diverse testen, worden gevalideerd op haar functionaliteiten, functies en entiteiten om de klant/stakeholders inzicht te geven op de werkelijk status van het eindproduct. Met welke tools verschilt per programmeertaal, maar de algemene richtlijnen van elke soort test, dat in dit document wordt uitgelegd, is altijd van toepassing. Het is in ieder geval aangeraden om, tijdens een van in dit document genoemde testen, gebruik te maken van de bijbehorende sjabloon die te vinden is in hoofdstuk 4.

**2. Testsoorten**

“Realistisch gezien, is testen een afweging tussen budget, tijd en kwaliteit. Het wordt aangestuurd door winstgerichte modellen." - Yang, M.C.K.; Chao, A. (1995)

Testen van het eindproduct in deze context is dus niets meer dan fouten/ongewenst gedrag opsporen en verhelpen in de broncode of programma, waardoor de klant/stakeholders financiële/tijdsgebonden/kwaliteitsgerichte winst van kunnen boeken. Toch zijn er verschillende manieren waarop het eindproduct getest kan worden.

De soorten en volgorde van testen is als volgt:

1. Regressietest
2. Unittest
3. Integratietest
4. Systeemtest

*Integratietest*

Componenten/Integratietest is het testen op problemen tijdens het samenvoegen van units.

De integratietest gebeurt achteraf de unittest maar vóóraf de systeemtest.

Deze volgorde is van groot belang omdat in een integratietest units eerst gevalideerd moeten zijn ná het doorvoeren hiervan in een unittest. Vervolgens worden deze units samengevoegd in grotere gehelen/verzamelingen die volgens een integratietestplan getest moeten worden. Zodra deze verzamelingen langs de integratietest zijn gegaan en gevalideerd zijn, dan pas zijn deze klaar voor gebruik in een systeemtest.

Integratietests omvat normaalgesproken heel veel programmeercode, met als gevolg dat er meer nakijkwerk is dan in unittests. Dit heeft invloed op de moeilijkheidsgraad van het opsporen van bugs wanneer een integratietest niet goed is uitgevoerd. Om dit probleem te verhelpen, moet het aantal te testen regels code verdeeld zijn in kleine brokstukken zodat bugs makkelijker opgespoord kan worden.

**3. Testgrenzen**

*Grenzen van integratietest*

De database, indien van toepassing, moet getest worden op veranderingen in waarde en restricties (nullable, unique of andere voorwaarde) per veld van ieder tabel tijdens de integratietest.

De applicatie moet getest worden op stabiliteit, flexibiliteit en functionaliteit per softwaremodule tijdens de integratietest.

Integratietest moet worden uitgevoerd binnen het kader van de onderstaande kwaliteitseigenschappen, waarin de te testen software hier niet van mag afwijkt:

* Inpasbaarheid - (De programmeercode van) ieder softwaremodule moet een ander softwaremodule kunnen ondersteunen of respecteren zodat er geen conflicten ontstaan tussen en binnen de modulen met als gevolg dat de samenvoeging van modules ten goede komt.
* Herbruikbaarheid - De geteste software moet grondig zijn gevalideerd op hergebruik van softwaremodules binnen een verzameling van een andere softwaremodules ongeacht de veranderingen in de betreffende broncode.
* Onderhoudbaarheid - Geteste softwaremodules moeten kunnen blijven functioneren ongeacht de veranderingen in de broncode van de gehele software.
* Connectiviteit - Geteste softwaremodules kunnen zonder slag of stoot met elkaar communiceren via gebruikersinterfaces (Front-end) en/of applicatie-interfaces (Backend).
* Testbaarheid - De scoop en factoren van ieder geteste softwaremodule is stapsgewijs (in heldere details) te meten.

**4. Testsjablonen**

Zodra een integratietest in de praktijk wordt uitgevoerd, beveel ik de onderstaande sjablonen aan. Dit omdat er zo een duidelijk overzicht gemaakt kan worden waar alle testgegevens op één plek staan en makkelijk terug te vinden is.

Echter moet de gebruikte sjabloon verder nog worden ingevuld, omdat het aantal rijen en gevulde waardes verschillen van context per software/applicatie, zoals:

* Themanaam en eventueel subnaam
* De in samenwerking tussen de klant en ontwikkelaar opgestelde Requirements als testcriteria
* Exacte waardes voor maximale- en behaalde score per test
* Eventueel meer kwaliteitseigenschappen met bijhorende toelichtingen als testcriteria

*Beoordelingscriteria voor integratietest*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testnaam | Integratietest op Androidapplicatie voor de ECI-Cultuurfabriek | | | |
| Themanaam | Evenementen organiseren binnen het ECI-Cultuurfabriek | | | |
| Subnaam |  | | |  |
|  | Testdoelen | Testcriteria | Maximale score | Behaalde score |
| 1 | Software/applicatie testen op Requirements |  | 18 ≈ 41,9% | 15 ≈ 34,9% |
| a. |  | Cursuscategoriën kunnen weergegeven, toegevoegd, bewerkt en/of verwijderd worden. | 5 | 5 |
| b. |  | Docenten kunnen weergegeven, toegevoegd, bewerkt en/of verwijderd worden. | 5 | 5 |
| c. |  | Cursussen kunnen weergegeven, toegevoegd, bewerkt en/of verwijderd worden. | 5 | 5 |
| d. |  | Er kan een inschrijving worden gedaan voor een cursus. | 2 | 0 |
| e. |  | Er kan een aanmelding worden gedaan voor een proefles van een bepaald cursus. | 1 | 0 |
|  | Software/applicatie testen op algemene kwaliteitseigenschappen |  | 25 ≈ 58,1% | 22 ≈ 51,2% |
| a. |  | Inpasbaarheid - (De programmeercode van) ieder softwaremodule moet een ander softwaremodule kunnen ondersteunen of respecteren zodat er geen conflicten ontstaan tussen en binnen de modulen met als gevolg dat de samenvoeging van modules ten goede komt. | 5 | 4 |
| b. |  | Herbruikbaarheid - De geteste software moet grondig zijn gevalideerd op hergebruik van softwaremodules binnen een verzameling van een andere softwaremodules ongeacht de veranderingen in de betreffende broncode. | 5 | 4 |
| c. |  | Onderhoudbaarheid - Geteste softwaremodules moeten kunnen blijven functioneren ongeacht de veranderingen in de broncode van de gehele software. | 5 | 5 |
| d. |  | Connectiviteit - Geteste softwaremodules kunnen zonder slag of stoot met elkaar communiceren via gebruikersinterfaces (Front-end) en/of applicatie-interfaces (Backend). | 5 | 5 |
| e. |  | Testbaarheid - De scoop en factoren van ieder geteste softwaremodule is stapsgewijs (in heldere details) te meten. | 5 | 4 |
| Totaal |  |  | 43 =  100% | 37 = 86% |

|  |  |
| --- | --- |
| **Integratie: Cursus** | **Testuitslag** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Verwacht resultaat** | Software/applicatie zal werken naar behoren volgens de in beoordelingscriteria beschreven Requirements en kwaliteitseigenschappen. | | | | | |
| **Relevante entiteiten** | Cursus, Docent, Cursuscategorie | | | | | |
| **Werkelijk** **resultaat** | Software/applicatie blijft werken naar behoren volgens de in beoordelingscriteria beschreven Requirements en kwaliteitseigenschappen. | | | | | |
| **Uitvoering** | **Uren** | 1 | **Prioriteit[[1]](#endnote-1)** | Hoog | **Door** | **Adam Oubelkas** |

**5. Literatuurlijst**

[1] Microsoft Word 2019 - Microsoft Corporation - <https://products.office.com/nl-nl/word>

[2] Swart, N. (2010) - Handboek Requirements - Brug tussen Business en ICT - Uitgeverij: Eburon Business.

Kaner, Cem (November 17, 2006). Exploratory Testing (PDF). Quality Assurance Institute Worldwide Annual Software Testing Conference. Orlando, FL.

Pan, Jiantao (Spring 1999). [*"Software Testing"*](http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/) (coursework). Carnegie Mellon University*. Retrieved November 21, 2017*.

Basu, Anirban (2015). [*Software Quality Assurance, Testing and Metrics*](https://books.google.de/books?id=aNTiCQAAQBAJ&pg=PA150). PHI Learning. [*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier)) [*978-81-203-5068-7*](https://en.wikipedia.org/wiki/Special:BookSources/978-81-203-5068-7).

<https://github.com/junit-team/junit4/wiki/Parameterized-tests>

<https://junit.org/junit5/>

[Martyn A Ould & Charles Unwin (ed), *Testing in Software Development*, BCS (1986), p71](https://books.google.com/books?id=utFCImZOTEIC&pg=PA73&dq=integration+test&hl=en&sa=X&ei=4EpTVOvJMayu7Aak5YCIDA&ved=0CDwQ6AEwAg#v=onepage&q=integration%20test&f=false). Accessed 31 Oct 2014

Xuan, Jifeng; Monperrus, Martin (2014). "Test case purification for improving fault localization". Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014: 52–63. [*arXiv*](https://en.wikipedia.org/wiki/ArXiv_(identifier)):[*1409.3176*](https://arxiv.org/abs/1409.3176). [*Bibcode*](https://en.wikipedia.org/wiki/Bibcode_(identifier)):[*2014arXiv1409.3176X*](https://ui.adsabs.harvard.edu/abs/2014arXiv1409.3176X). [*doi*](https://en.wikipedia.org/wiki/Doi_(identifier)):[*10.1145/2635868.2635906*](https://doi.org/10.1145%2F2635868.2635906). [*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier)) [*9781450330565*](https://en.wikipedia.org/wiki/Special:BookSources/9781450330565).

Martin, Robert C. (2000) - Design Principles and Design Patterns (PDF)

<https://en.wikipedia.org/wiki/System_integration_testing>

1. Toelichting voor de waardes van een prioriteit:  
   1 = Laag  
   2 = Middel  
   3 = Hoog [↑](#endnote-ref-1)