**Deelproduct 5e**

**Systeemtest**

**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

*Systeemtest*

Een systeemtest is het meten van reactievermogen, verwerkingssnelheid, betrouwbaarheid en of/ schaalbaarheid van een systeem binnen een bepaalde werklast. Hierin wordt het gehele systeem van de klant/stakeholders, bestaande uit vele verschillende hardware- en software modulen, getest op criteria die de klant/stakeholders hebben gesteld over het systeem. In deze criteria staan o.a. de gerelateerde Requirements en verwachtingen van de klant/stakeholders.

**3. Testgrenzen**

*Grenzen van systeemtest*

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 systeemtest.

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

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

* Beheerbaarheid - Gebruiker van de geteste software en systeem kan zelf deze software op normale wijze (zoals geschreven in de Requirements) aansturen in dit systeem.
* Beveiliging - Geteste software levert geen conflicten op voor andere software en hardware van het systeem, zodanig dat de software geen gevaar vormt voor de particuliere werkzaamheden binnen het systeem van de gebruiker.
* Portabiliteit - De geteste software kan op ieder systeem ongehinderd functioneren, waarvan besturingssysteem door de software ondersteund wordt, ongeacht de hardware- en softwaremodules.
* Continuïteit - De geteste software moet binnen een door de klant bepaalde termijn kunnen functioneren zonder hulp, toevoegingen en bijwerkingen vanuit de ontwikkelaar, ongeacht verandering in de hardware modules van het geteste systeem.
* Gebruikersvriendelijkheid - Software moet ongeacht de hardware-modules van het geteste systeem kunnen functioneren m.b.t. verschillende handelingen die de gebruiker kan uitvoeren. Als gevolg hiervan zal de gebruiker zich niet ergeren aan het gebruik van de software.

**4. Testsjablonen**

Zodra een systeemtest 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 systeemtest*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testnaam | Systeemtest 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 |  | 20 ≈ 44,4% | \*punten\* ≈ \*percentage afgerond op één decimaal\*% |
| a. |  | Binnen één venster van de applicatie wil de klant graag een venster, waarin een overzicht van evenementen en registratieformulier te zien is. Op dit venster kan een evenement geselecteerd worden en hiervan de bijbehorende gegevens in het formulier getoond worden. | 10 | n.t.b. |
| b. |  | Daarnaast moet, binnen dezelfde venster van de applicatie, het voor consumenten mogelijk zijn om zich aan te melden voor een evenement en hieruit, indien van toepassing, een ticket/registratiebewijs van te downloaden. | 10 | n.t.b. |
|  | Software/applicatie testen op algemene kwaliteitseigenschappen |  | 25 ≈ 55,6% | \*punten\* ≈ \*percentage afgerond op één decimaal\*% |
| a. |  | Beheerbaarheid - Gebruiker van de geteste software en systeem kan zelf deze software op normale wijze (zoals geschreven in de Requirements) aansturen in dit systeem. | 5 | n.t.b. |
| b. |  | Beveiliging - Geteste software levert geen conflicten op voor andere software en hardware van het systeem, zodanig dat de software geen gevaar vormt voor de particuliere werkzaamheden binnen het systeem van de gebruiker. | 5 | n.t.b. |
| c. |  | Portabiliteit - De geteste software kan op ieder systeem ongehinderd functioneren, waarvan besturingssysteem door de software ondersteund wordt, ongeacht de hardware- en softwaremodules. | 5 | n.t.b. |
| d. |  | Continuïteit - De geteste software moet binnen een door de klant bepaalde termijn kunnen functioneren zonder hulp, toevoegingen en bijwerkingen vanuit de ontwikkelaar, ongeacht verandering in de hardware modules van het geteste systeem. | 5 | n.t.b. |
| e. |  | Gebruikersvriendelijkheid - Software moet ongeacht de hardware-modules van het geteste systeem kunnen functioneren m.b.t. verschillende handelingen die de gebruiker kan uitvoeren. Als gevolg hiervan zal de gebruiker zich niet ergeren aan het gebruik van de software. | 5 | n.t.b. |
| Totaal |  |  | 45 =  100% | \*totaal behaalde punten\* = \*Gelijke percentage\*% |

|  |  |
| --- | --- |
| **Systeem: PC ECI-Cultuurfabriek** | **Testuitslag** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Verwacht resultaat** |  | | | | | |
| **Relevante entiteiten** |  | | | | | |
| **Werkelijk** **resultaat** |  | | | | | |
| **Aanpassingen** |  | | | | | |
| **Uitvoering** | **Uren** | 1 | **Prioriteit[[1]](#endnote-1)** | Hoog/middel/laag | **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)