PENGGUNAAN METODE ACCEPTENCE TEST DRIVEN DEVELOPMENT PADA PROSES ACCEPTENCE TEST MENGGUNAKAN CODECEPTION

[1]Sugiarto, [2]Budi Nugroho, [3]Azalia Dwi Putri
[123]Teknik Informatika Fakultas Ilmu Komputer
Universitas Pembangunan Nasional "Veteran" Jawa Timur
Jl. Raya Rungkut Madya, Gunung Anyar, Surabaya, Jawa Timur

Email: [1]sugiarto.if@upnjatim.ac.id

Abstrak. *Acceptence test merupakan salah satu jenis pengujian pada pengembangan perangkat lunak, acceptance test tidak hanya dilakukan satu kali melainkan berkalli-kali jika program banyak mengalami perubahan, dalam accpetence test, tester memposisikan diri sebagai pengguna aplikasi. Perilaku pengguna dibentukan dalam sebuah scenario ujicoba, dengan mengadpatasi metode Acceptence Test Driven Development (ATDD), scenario pengujian dan pembuatan script pengujian dilakukan di awal pengembangan perangkat lunak. Pengujian dilakukan untuk menguji aktivitas tambah, ubah, dan hapus pada beberapa form sistem atau aplikasi yang sudah ada. Penelitian ini membuat sebuah proses Automated Acceptence Testing yang biasanya dilakukan secara manual. Pembuatan proses Automated Acceptence Test menggunakan tools codeception. Hasil dari pengujian semua scenario tersebut adalah Passed (Lolos). Namun hasil pengujian sangat bergantung dengan perangkat keras yang digunakan untuk pengujian.*

Kata Kunci: Acceptence Test, Acceptence Test Driven Development, Codeception.

Dalam pembangunan perangkat lunak sistem informasi terdapat proses yang kompleks, tidak hanya melibatkan proses coding tetapi juga proses testing menjadi salah satu kunci keberhasilan dalam mencapai tujuan sistem informasi tersebut. Proses testing sebagai salah satu cara untuk menjaga agar sistem informasi dikembangkan sesuai dengan tujuan awal dan tidak melebar ke permasalahan lain dan juga untuk memastikan alur bisnis berjalan sesuai dengan skenario yang sudah disepakati dengan klien sehingga testing menjadi sarana penting untuk menjaga kepercayaan klien. (Al-Fatta, 2007).

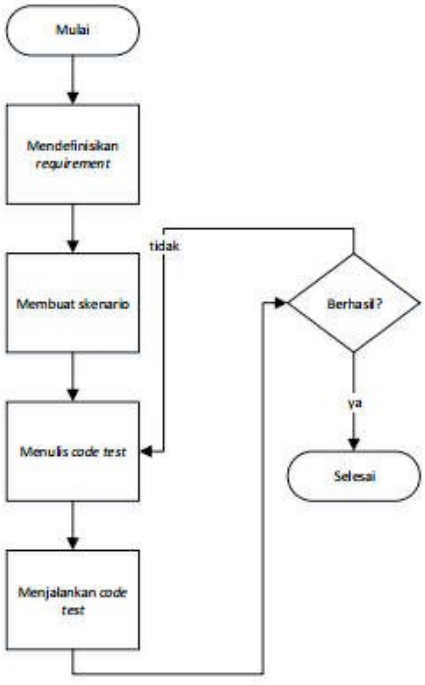
Acceptance test adalah salah satu jenis pengujian pada perangkat lunak. Acceptance test dilakukan untuk menyesuaikan perangkat lunak yang dibangun sesuai dengan requirement yang ada atau kontrak yang telah disepakati (Koudelia & Nikolai, 2011), dimana proses tersebut akan menentukan diterima atau tidaknya perangkat lunak yang sedang atau telah dibangun. Dalam acceptance test, tester memposisikan diri sebagai pengguna aplikasi. Perilaku pengguna diterjemahkan dalam bentuk skenario. Acceptance test berkaitan dengan antarmuka (interface) aplikasi yang terdiri dari form-form inputan maupun tombol-tombol. Pada proses pengembangan perangkat lunak, acceptance test tidak hanya dilakukan satu kali melainkan berkali-kali jika program banyak mengalami perubahan. Salah satu metode pengembangan perangkat lunak yang terbilang baru dan menjadi sorotan selama beberapa tahun terakhir ini adalah Acceptance Test Driven Development (ATDD) (Kamalrudin, Sidek, Moketar, & Robinson, 2013). ATDD merupakan bagian dari metode agile software development, secara garis besar ATDD melibatkan proses acceptance testing sebelum implementasi kode program yang dapat meningkatkan produktivitas waktu pengembangan perangkat lunak serta kesesuaian requirement dari perangkat lunak yang dibangun (Pugh, 2010). Biasanya, pengembang aplikasi melakukan proses acceptance testing pada metode ATDD secara manual yaitu menggunakan interaksi manusia dan ini menguras waktu dalam pengembangan aplikasi tersebut. Proses pengujian yang manual ini dapat diotomatisasi dengan menggunakan tools Codeception.

Codeception merupakan *automated framework testing* yang menyederhanakan penulisan test script yang dapat mempercepat dan meningkatkan keefektifan pada proses acceptance testing.

Sebagai contoh, terdapat beberapa penelitian yang membahas mengenai implementasi metode TDD dalam pembuatan aplikasi. Salah satu penelitian tersebut adalah penelitian yang dilakukan oleh [7] yang melakukan implementasi TDD untuk pengembangan sisten informasi rawat jalan. Kemudian implementasi TDD juga dilakukan dalam pembuatan perangkat lunak pengajuan tugas akhir [8]. Hasil yang diperoleh menunjukan bahwa dengan menerapakan TDD pengecekan error lebih mudah dilakukan. Selain itu pengembang dapat bekerja lebih cepat karena simulasi manual untuk pengujian fungsi perangkat lunak menjadi berkurang.

I. Metodologi

Pada proses pengujian *acceptance test* menggunakan *codeception*, *tester* melakukan koordinasi dengan *developer* baik pada saat awal pengembangan perangkat lunak sampai tahap akhir yaitu pengujian. Alur dari pembuatan *script* pengujian sampai dengan menjalankan pengujian dapat digambarkan seperti di bawah ini :



Gambar 1. Tahapan Pengembangan Sistem Informasi

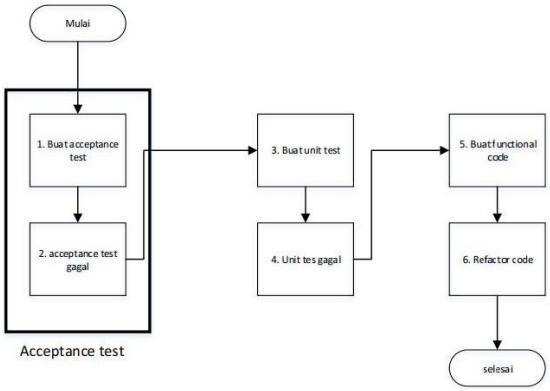
Langkah pertama yang dilakukan adalah mendefinisikan requirement yang didapat dari klien. Pada tahap ini sistem analis melakukan pendefinisian guna untuk memastikan requirement yang diminta dan juga menentukan ruang lingkup dari perangkat lunak yang akan dikembangkan. Kemudian, tester mulai menyusun skenario pengujian beserta hasil yang diharapkan dan mencocokan kesesuaian dengan ruang lingkup yang sudah didefinisikan sebelumya. Selanjutnya, dilakukan penulisan script test sesuai dengan skenario. Lalu script test tersebut dijalankan, jika hasil pengujian sesuai dengan hasil yang diharapkan maka script test dinyatakan berhasil (*passed*), namun jika tidak sesuai dengan hasil yang diharapkan maka koreksi lagi pada script test apakah ada masalah yang menyebabkan script test tersebut gagal (*failed*).

Metode penelitian ini menggunakan *Acceptance Test Driven Development* (ATDD). ATDD merupakan perkembangan dari metode *Test Driven Development* (TDD). TDD bisa dikatakan melakukan pengujian sebelum pengkodean, bertujuan untuk mengevaluasi hasil tes tersebut untuk menentukan kecacatan atau kesalahan yang mungkin terjadi pada perangkat lunak kepada tim pengembang perangkat lunak tersebut (Hendrickson, 2008). Serupa dengan TDD, ATDD juga melakukan pengujian sebelum implementasi kode, dimana hasil tes tersebut mempresentasikan espektasi dari perangkat lunak yang harus dimiliki (Hendrickson, 2008), namun ATDD memperluas aplikasi dari teknik TDD untuk tingkatan acceptance testing, menggunakan user-level black box tests sebagai pengganti unit tests (Sauvé, Abath Neto, & Cirne, 2006).

Dalam pemilihan tools untuk automated testing pada proses acceptance test terdapat beberapa pilihan, untuk perangkat lunak yang berbasis website yang menggunakan bahasa pemrograman HTML contoh tools yang dapat digunakan adalah Selenium dan Cucumber, pada bahasa pemrograman PHP dapat menggunakan PHPUnit, Codeception, Behat, SimpleTEst, pada bahasa pemrograman Java dapat menggunakan Carina, untuk bahasa pemrograman Phyton dapat menggunakan RobotFramework, untuk bahasa

pemrograman Ruby dapat menggunakan Watir, untuk bahasa pemrograman mobile seperti android dan IOS dapat menggunakan Appium dan Google EarlGrey.

Dengan ATDD, tester diharuskan menulis acceptance test terlebih dahulu sebelum dibuat code utamanya oleh developer. Siklus ini berulang sampai tujuan akhir dari acceptance test tercapai. Dalam penelitian ini yang dikerjakan oleh tester hanya proses acceptance test-nya saja, yaitu pada langkah 1 dan 2. Implementasinya adalah sebagai berikut :



Gambar 2. Implementasi ATDD

Dari gambar 2 diatas dijelaskan alur atau langkah pengerjaan dari ATDD sebagai berikut :

1. Tuliskan Acceptance Test.
   Pastikan tes dibuat berdasarkan use case yang diterima dari klien, dan tuliskan semua kemungkinan yang dapat diterima untuk input dan output-nya agar semua harapan klien terpenuhi sesuai standar.
2. Pastikan Acceptance Test Gagal.
   Karena belum ada kode apapun yang membuat tes tersebut lolos (passed).
3. Tuliskan Unit Test.
   Penting untuk menulis unit test sebelum kode fungsional. Ini membantu memastikan bahwa setiap baris kode dapat diuji.
4. Pastikan Uunit Test Gagal.
   Sama seperti acceptance test, karena kode fungsional belum siap, unit test harus gagal.
5. Tuliskan Kode Fungsional.
   Terapkan tujuan dari kode fungsional dan pastikan bahwa unit test dan acceptance test lolos (passed).
6. Refactor Code.
   Lakukan perubahan kode jika merasa kode yang sudah ditulis tidak rapi. Selama unit test tidak ada yang gagal berarti tidak masalah terhadap kode yang di-refactor tersebut.

II. Hasil dan Pembahasan

Pada bagian ini akan dipaparkan hasil dari penelitian ini. Hasil yang akan ditampilkan adalah bagaimana langkah-langkah pembuatan script test dan hasil pengujian dari script test tersebut.

1. Pembuatan skenario pengujian
   Dalam penelitian ini akan ditampilkan contoh skenario login dengan data valid dan tidak valid.

Tabel 1. Skenario Login dengan Data Valid

Skenario	Langkah Tes	Jenis Inputan	Hasil yang diharapkan
Login dengan data valid	1. Ke login page		Masuk ke home
	2. Masukkan username = tu@sma.com	textfield	
	3. Masukkan password = 1234	textfield	
	4. Pilih tahun ajaran = 2014/2015 Genap	combo box	
	5. Klik Sign In / Tekan Enter	button	

Tabel 2. Skenario Login dengan Data Tidak Valid

Skenario	Langkah Tes	Jenis Inputan	Hasil yang diharapkan
Login TU dengan salah satu data tidak valid	1. Ke login page		Munculkan pesan error 'Kombinasi username/password salah. Silahkan coba lagi.'
	2. Masukkan username = tu@sma.com	textfield	
	3. Masukkan password = assdfghjkuhegghb	textfield	
	4. Pilih tahun ajaran = 2015/2016 Genap	combo box	
	5. Klik Sign In / Tekan Enter	button	

2. Implementasi Codeception
   a. Instalasi Codeception
   Install dependency Codeception yang paling stabil via Composer, caranya yaitu buka command prompt, lalu masuk ke direktori dimana file acceptance test berada, ketikkan perintah berikut :

```
composer require
"codeception/codeception"  --dev
```

   b. Membuat Bootstrap File
   Eksekusi perintah untuk membuat bootstrap-nya, ketika perintah ini dijalankan maka codeception akan meng-generate file-file yang dibutuhkan untuk proses testing. Hal ini akan membuat file codeception.yml dan direktori tests dan default tes.

```
"codecept bootstrap"
```

   c. Konfigurasi Web Driver
   Pastikan bahwa aplikasi yang akan diuji berjalan dengan baik. Buka file tests/acceptance.suite.yml, lalu lakukan beberapa konfigurasi seperti URL Aplikasi yang akan diuji dan WebDriver. Webdriver yang digunakan untuk penelitian ini adalah chromedriver yang dapat mengeksekusi test script pada real browser google chrome layaknya pengujian secara manual, namun goooogle chrome akan dikontrol oleh sebuah servis.

```
"actor: AcceptanceTester
modules:
  enabled:
  - WebDriver:
  url:
  http://localhost/eschoolface-
  lift/publicwindow_size: false
  port: 9515
  browser: chrome"
```

   d. Membuat File Accpetence Test

```
"codecept generate:cest acceptance
LoginValid"

"codecept generate:cest acceptance
LoginTidakValid"
```

   e. Membuat Script Pengujian Pengujian Login Valid

```
"class LoginValidCest
{
public function
_before(AcceptanceTester  $I){
}
// tests
public function
tryToTest(AcceptanceTester  $I)
{
$I->wantTo('perform actions and
see result');
$I->amOnPage('/');
$I-
>fillField('email','tu@sma.com');
$I->wait(2);
$I->fillField('password','1234');

$I>selectOption('semester_id','201
4/2015 Ganjil');

$I>click('//select[@name="semester
_id"]');

$I>click('//select[@name="semester
_id"]/option[@value="20152"]');

$I->wait(2);

$I->click('SIGN IN');

$I->see('dashboard');

$I->wait(3);"
```

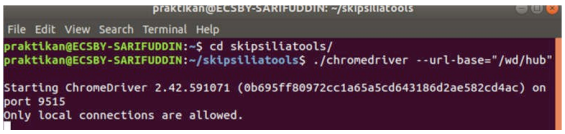
   f. Membuat Script Pengujian Login Tidak Valid

```
"Public function
tryToTest(AcceptanceTester  $I)
{
$I->wantTo('perform  actions  and
see result');
$I->amOnPage('/');
$I->fillField('email','tu');
```

```
$I->wait(2);
$I>fillField('password','assdfghjk
uheghb');
$I->click('SIGN  IN');
$I->see('Kombinasi
username/password  salah.  Silahkan
coba lagi');
$I->wait(2);
}"
```

g. Menjalakan Pengujian
Dalam penelitian ini hanya akan ditampilkan skenario tes login valid dan tidak valid dan hasil pengujian skenario tes login valid dan tidak valid karena keterbatasan aturan pembuatan dokumen jurnal penelitian, hal ini dikarenakan pengujian ini menggunakan chromedriver, maka sebelum menjalankan pengujian pastikan bahwa servis chromedriver telah jalan. Untuk menjalankan servis chromedriver dapat dilakukan dengan cara seperti berikut :

```
Chromedriver  -url-base="\wd\hub"
```

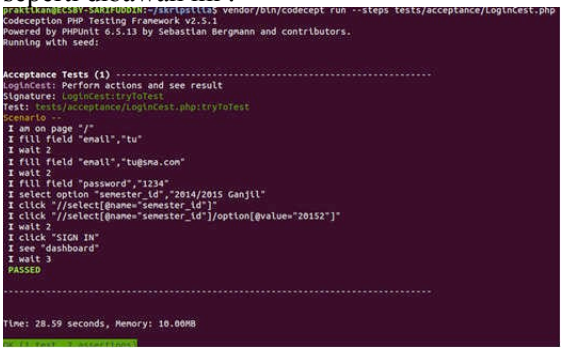


Gambar 3. Hasil Menjalankan Chromedriver

Untuk menjalankan pengujian login valid, masukkan perintah berikut

```
"codecept  run  --steps
LoginValidCest.php"
```

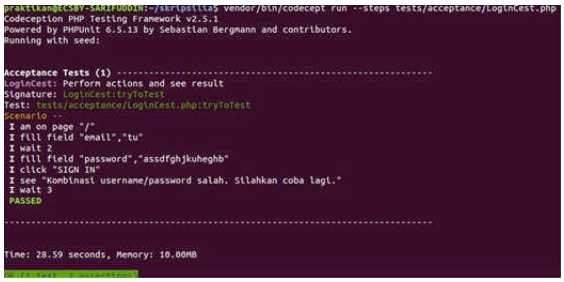
Dari test script diatas mendapatkan hasil seperti dibawah ini :



Gambar 4. Hasil Pengujian Login Valid

Dari hasil pengujian gambar 4 diatas, dimana login dengan data valid sesuai dengan Tabel 1 Skenario Login dengan Data Valid, terlihat bahwa antara hasil yang diharapkan dengan hasil pengujian sesuai ekspektasi yaitu berhasil (*passed*).
Untuk menjalankan pengujian login tidak valid, masukkan perintah berikut.

```
"codecept  run  --steps
LoginTidakValidCest.php"
```



Gambar 5. Hasil Pengujian Login Tidak Valid

Dari hasil pengujian login dengan data valid sesuai dengan Tabel 2 Skenario Login dengan Data Tidak Valid, terlihat bahwa antara hasil yang diharapkan dengan hasil pengujian sesuai ekspektasi yaitu berhasil (passed).

III. Simpulan

Dalam penelitian Penggunaan Metode Acceptance Test Driven Development Pada Proses Acceptance Test Menggunakan Codeception, dapat ditarik beberapa kesimpulan sebagai berikut :

1. Penggunaan codedeption pada proses acceptance testing terbilang lebih cepat dari segi waktu dan lebih tinggi tingkat keefektifan setiap skenario tes. Bagaimanapun juga penggunaan automated framework test memerlukan usaha yang lebih dibanding pengujian secara manual. Namun jika developer dan tester sudah terlatih dan terbiasa menggunakan codeception maka dampak positif penggunaan automated framework test akan lebih terasa.
2. Hasil pengujian script test dengan codeception sangat bergantung dengan perangkat keras yang digunakan ketika pengujian. Dikarenakan proses tunggu (wait) tidak berefek sama pada semua perangkat.
3. Dibalik hal positif dari penggunaan automated framework testing yang sudah disebutkan, automated framework testing juga memicu munculnya defect tersendiri, yaitu defect yang disebabkan penulisan script test. Penyebab munculnya kesalahan pada script test disebabkan

oleh human error, kesalahan dalam menerjemahkan skenario tes kedalam format script test, maupun ketidaksesuaian script test dengan sumber kode yang telah ditulis.

4. Menggabungkan manual acceptance test dan automated acceptance test merupakan salah satu upaya mengoptimalkan proses acceptance testing pada ATDD.

IV. Daftar Pustaka

[1] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. Finland: VTT. Publication 478.

[2] Al-Fatta, H. (2007). Analisis & Perancangan Sistem Informasi Untuk Keunggulan Bersaing Perusahaan & Organisasi Modern. Yogyakarta: ANDI.

[3] Alliance, A. (2018). Agile Alliance. Retrieved from Agile Alliance: https://www.agilealliance.org

[4] Anggarwal, V., & Singh, M. (2014). Acceptance Test Driven Development. Journal of Advanced Computing and Communication Technologies (ISSN: 2347 - 2804), 1-4.

[5] Beck, K. (2003). Test-Driven Development By Example. Addison-Wesley Professional. Codeception, T. (2018).

[6] Codeception. Retrieved from Codeception: http://codeception.com/

[7] Rachmadi, G. 2015. "Implementasi Test Driven Development Dalam Studi Kasus Pengembangan Sistem Informasi Rawat Jalan Di Rumah Sakit Hewan Universitas Airlangga". Skipsi. Universitas Airlangga, Surabaya

[8] Firmansyah, R. 2016. "Pengembangan Perangkat Lunak Pengajuan Sidang Tugas Akhir Menggunakan Meodologi Test Driven Development". Skripsi. Universitas Pasundan, Bandung.