IMPLEMENTASI METODE *WHITE BOX TESTING* PADA PROSES *QUALITY ASSURANCE* PERANGKAT LUNAK BERBASIS *WEB* DAN *MOBILE COLLECTION SYSTEM*

Roy Mubarak
Program Studi Teknik Informatika STMIK Eresha, Jl. Raya Puspitek No.10, Tangerang Selatan

E-mail: roy.dosen@gmail.com

ABSTRAK

Pengujian perangkat lunak merupakan salah satu tahap dalam pengembangan perangkat lunak. Pengujian perangkat lunak dilakukan untuk mencari kesalahan perangkat lunak yang akan dikembangkan[1]. Selain akan dibahas beberapa teknis yang digunakan untuk menguji perangkat lunak yang sudah dibuat sekaligus melakukan proses *Quality Assurance* sebelum perangkat lunak tersebut diimplementasikan. Perangkat lunak yang akan di uji terbagi menjadi 2, yaitu perangkat lunak berbasis *web* dan perangkat lunak berbasis *mobile*. Pokok dari bahasan ini memberikan tinjauan literatur tentang teknik pengujian perangkat lunak yang sudah dibuat mempergunakan *Teknik White Box Testing*.

Kata Kunci *Testing*, Perangkat Lunak, *Software, White-Box, Web Application, Mobile Application*

1. PENDAHULUAN

Dalam kebanyakan software pengembangan, program pengujian fungsi sebagai akhir *"quality gate"* untuk sebuah aplikasi, memungkinkan atau mencegah bergerak dari kenyamanan lingkungan software-engineering ke dalam dunia nyata. Hal ini dapat langsung diterapkan oleh personel software untuk meningkatkan produk mereka dan menghindari kesalahan dan kelalaian. [2].

Testing yang sukses adalah testing yang dapat mengungkap semua kesalahan yang belum pernah terjadi sebelumnya. Dan test case yang baik adalah test case yang memiliki kemungkinan besar untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya. Sehebat apapun testing yang dilakukan, tidak ada software yang bebas dari kesalahan. [3]

Software testing sangat penting dalam aktivitas dalam pengembangan perangkat lunak. Ini adalah salah satu aktivitas penting yang membutuhkan banyak waktu dan tenaga kerja. Teknik pengujian yang berbeda digunakan untuk menemukan bug pada perangkat lunak. Pengujian dilibatkan pada berbagai tahap pengembangan perangkat lunak seperti *Unit Test, Integration Test, System Testing, dan Acceptance Test*. Teknik pengujian yang berbeda yaitu *Dynamic Test, Functional Testing* dan *Structural Testing* digunakan untuk menguji perangkat lunak.[4]

1.1 Latar Belakang

Ide pengujian program ini sebagai komputer program mendapat lebih besar dan lebih besar sejak awal mereka pada tahun 1960, kebutuhan untuk menghilangkan kesalahan dari mereka secara sistematis mendapat perhatian lebih. Kedua komunitas riset dan praktisi menjadi lebih dalam terlibat dalam pengujian perangkat lunak. Dengan demikian, pada 1970-an, bidang penelitian baru yang disebut teori pengujian muncul. [5]

Secara umum, metodologi yang paling tidak efektif adalah input acak Pengujian-proses pengujian program dengan memilih, secara acak, Beberapa subset dari semua nilai masukan yang

mungkin. Dalam hal kemungkinan Mendeteksi kesalahan yang paling banyak, kumpulan uji kasus yang dipilih secara acak Memiliki sedikit kesempatan untuk menjadi subset optimal, atau mendekati optimal.[6]

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah memberikan tinjauan literatur tentang teknik pengujian perangkat lunak yang sudah dibuat. Kemudian, membandingkan beberapa teknik pengujian yang sudah ada agar dapat digunakan sesuai dengan aplikasi yang dibuat.

2. LANDASAN TEORI

Untuk mengetahui beberapa teknik testing software yang sudah ada dan banyak dipakai oleh dunia industri pembuatan software aplikasi.

2.1 *Test-Case Design*

Pengujian, betapapun kreatif dan nampaknya lengkap, tidak bisa menjamin tidak adanya semua kesalahan. Test-Case Design sangat penting karena pengujian yang lengkap tidak mungkin. Uji terhadap setiap program harus selalu tidak lengkap. Strategi yang jelas, kemudian, adalah mencoba melakukan tes Selengkap mungkin. Mengingat kendala waktu dan biaya, menjadi masalah utama pengujian.[6]

2.2 White-Box

White box testing merupakan teknik pengujian yang menggunakan struktur dan perancangan prosedural untuk memperoleh kasus uji. White box testing dapat mengungkap kesalahan penerapan dengan menganalisa kerja internal dan struktur sebuah software. Pada pengujian ini tester perlu melihat kode suatu program untuk mengetahui bahwa unit dari kode berperilaku tidak tepat. Beberapa metode yang terdapat pada pengujian ini adalah basis path testing, control flow testing , branch testing dan lainnya.[7]

2. 3 Module (Unit) Testing

Module testing (or unit testing) adalah proses pengujian subprogram, subrutin, atau prosedur individual dalam sebuah program. Artinya, daripada pada awalnya menguji keseluruhan program, pengujian pertama-tama difokuskan pada building blocks yang lebih kecil dari program.

Tujuan pengujian modul adalah membandingkan fungsi modul dengan beberapa spesifikasi fungsional atau antarmuka yang menentukan modul. Untuk menekankan kembali tujuan semua proses pengujian, tujuannya di sini bukan untuk menunjukkan bahwa modul tersebut memenuhi spesifikasi, namun untuk menunjukkan bahwa modul tersebut bertentangan dengan spesifikasi.[6]

2.4 Dasar Arsitektur Berbasis Web dan Mobile

Tier 1, Web server mewakili tingkat pertama dalam arsitektur sistem berbasis *Web* dan *Mobile, Web Server* berfungsi untuk menampung *Situs Web*. Tampilan dan nuansa aplikasi Internet berasal dari tingkat pertama. Dengan demikian, istilah lain untuk tier ini adalah Presentation tier atau layer, yang dijuluki karena memberikan konten visual kepada pengguna akhir. *Server Web* dapat menggunakan skrip statis *Text Markup Language (HTML)* statis atau skrip Common Gateway Interface (CGI) untuk membuat HTML dinamis, namun kemungkinan besar menggunakan kombinasi halaman statis dan dinamis.

Tier 2, atau lapisan bisnis menampung server aplikasi. Di sini Anda menjalankan perangkat lunak yang memodelkan proses bisnis Anda. Berikut adalah daftar beberapa fungsi yang terkait dengan lapisan bisnis:

- Proses transaksi
- Otentikasi pengguna
- Validasi data
- Aplikasi logging

Tingkat ketiga berfokus pada penyimpanan dan pengambilan data dari sumber data,

Relational Database Management System (RDBMS). Istilah lain untuk Tier 3 adalah lapisan Data. Tingkat ini terdiri dari infrastruktur database untuk berkomunikasi dengan tingkat kedua. Antarmuka ke lapisan Data ditentukan oleh model data, yang menjelaskan bagaimana Anda ingin menyimpan data

Testing Challenges

Anda akan menghadapi banyak tantangan saat merancang dan menguji aplikasi berbasis *Web* dan *Mobile* karena banyaknya elemen yang tidak dapat Anda kendalikan dan jumlah komponen yang saling terkait.

Aplikasi berbasis *Web* dan *Mobile* memiliki banyak poin kegagalan yang harus di pertimbangkan saat merancang pendekatan pengujian. Daftar berikut memberikan beberapa contoh tantangan yang terkait dengan pengujian aplikasi berbasis *Web* dan *Mobile*:

- Dasar pengguna yang besar dan beragam
- Lingkungan Bisnis: aplikasi terletak pada Cloud System, jadi proses testing juga harus memperhatikan aspek bisnis, contohnya pada papper ini yang membahas masalah *Collection System*
- Menguji lingkungan: Untuk menguji aplikasi Anda dengan benar, Anda perlu menduplikat lingkungan produksi. Ini berarti Anda harus menggunakan server Web, server aplikasi, dan server database yang identik dengan peralatan produksi. Untuk hasil pengujian yang paling akurat, infrastruktur jaringan harus diduplikasi juga. Ini termasuk router, switch, dan firewall.
- Keamanan: Karena situs Anda terbuka untuk dunia, Anda harus melindunginya dari peretas. Mereka dapat membawa Situs Anda ke penghentian dengan serangan denial-of-service (DoS) atau merobek informasi kartu kredit pelanggan Anda.

Pengujian Otorisasi dan Lingkungan Pengguna Aplikasi

Menjadi lebih rumit saat aplikasi Anda sangat bergantung pada pemrosesan skrip sisi klien. Setiap browser memiliki mesin scripting atau mesin virtual yang berbeda untuk menjalankan skrip dan kode pada komputer klien. Perhatikan masalah kompatibilitas browser jika Anda menggunakan salah satu dari berikut ini:

• ActiveX controls
• JavaScript
• VBScript
• Java applets

Pengujian Lapisan Bisnis Proses

Pengujian lapisan bisnis berfokus pada menemukan kesalahan dalam logika bisnis proses dari aplikasi. Ada beberapa karakteristik aplikasi yang harus selalu diuji yang meliputi:

- *Performance*. Test untuk melihat apakah aplikasi memenuhi spesifikasi kinerja terdokumentasi (umumnya ditentukan dalam waktu respon dan tingkat throughput).
- *Data Validity*. Test untuk mendeteksi kesalahan dalam data yang dikumpulkan dari pelanggan.
- *Data Transaction*. Untuk mengungkap kesalahan dalam pemrosesan transaksi, yang mungkin mencakup item seperti: pemrosesan data dari sistem lain yang berhubungan dengan sistem *collection* ini, integrasi data *collection* dari aplikasi *web* ke aplikasi *mobile* serta sebaliknya. verifikasi data, otorisasi penggua, dan lain sebagainya.

Data Layer Testing

Setelah situs Anda aktif dan berjalan, data yang dikumpulkan menjadi sangat berharga. Kehilangan informasi ini bisa menjadi bencana dan dapat melumpuhkan proses bisnis yang terjadi. Oleh karena itu, harus dikembangkan prosedur untuk melindungi sistem penyimpanan data pada aplikasi. Beberapa aspek yang harus diperhatikan adalah:

- Waktu respon dari database. Pengujian sampai ke tingkat query pada database.
- Integritas data.Verifikasi bahwa data disimpan dengan benar dan akurat.
- Toleransi dan pemulihan kesalahan.

3. METODOLOGI

Studi literature dari:

- Review paper, yang dicari dari IEEE Journal, Google Search.
- Dokumentasi dari pengujian untuk aplikasi yang sejenis.
- Buku-buku metodology pengujian dengan mempergunakan *White Box Testing.*

4. ANALISA DAN PEMBAHASAN

4.1 Tujuan dari Teknik Pengujian Perangkat Lunak

- Kasus uji yang baik adalah kasus yang memiliki probabilitas untuk menemukan kesalahan yang belum ditemukan.
- Tes yang bagus harus detail
- Tes sukses adalah tes yang mengungkap kesalahan yang belum ditemukan.
- Tes yang bagus harus "terbaik untuk berkembang biak".
- Tes yang bagus tidak boleh terlalu sederhana atau terlalu rumit.
- Untuk memeriksa apakah sistem melakukan apa yang diharapkannya dilakukan.
- Untuk memeriksa apakah sistemnya "Fit for purpose".
- Untuk memeriksa apakah sistem memenuhi persyaratan dan berhasil dieksekusi di lingkungan yang Ditujukan.
- Melaksanakan program pengujian dengan maksud untuk menemukan kesalahan yang ada pada aplikasi yang sudah dikembangkan

4.2 Level yang berbeda Pengujian

Pengujian sering dikelompokkan berdasarkan di mana mereka ditambahkan dalam proses pengembangan perangkat lunak, atau berdasarkan tingkat spesifisitas pengujian. Pengujian dilibatkan dalam setiap tahap siklus hidup perangkat lunak, namun pengujian yang dilakukan pada setiap tingkat pengembangan perangkat lunak berbeda sifatnya dan memiliki tujuan yang berbeda.

Unit testing

Hal ini dilakukan pada tingkat terendah. Ini menguji unit perangkat lunak dasar, yang merupakan perangkat tabel terkecil dari perangkat lunak. Hal ini juga disebut pengujian modul atau komponen. Ini mengacu pada tes yang memverifikasi fungsionalitas dari bagian kode tertentu, biasanya pada tingkat fungsi. Dalam lingkungan berorientasi objek, ini biasanya di tingkat kelas, dan unit tes minimal mencakup konstruktor dan destruktor.

Dalam pembahasan disini, *Unit Testing* digunakan untuk menguji aplikasi *Web* dan *Mobile* secara terpisah satu sama lain. Pengujian Unit Testing ini belum meliputi integrase diantara keduanya. Beberapa hal yang diuji pada unit testing pada pembahasan ini adalah:

- *Web Based Application Collection System*
- *Mobile Apps Collection System*
- *Database Collection System*

Integration Testing

Hal ini dilakukan ketika dua atau lebih unit yang diuji digabungkan ke dalam struktur yang lebih besar. Tes ini sering dilakukan pada kedua antarmuka antara komponen dan struktur yang lebih besar yang sedang dibangun, jika sifat kualitasnya tidak dapat dinilai dari komponennya.

Pengujian integrasi adalah semua jenis pengujian perangkat lunak yang berusaha

memverifikasi antarmuka antar komponen terhadap perancangan perangkat lunak. Komponen perangkat lunak dapat diintegrasikan secara berulang-ulang satu per satu atau semua komponen. Pengujian integrasi bekerja untuk mengekspos cacat pada antarmuka dan interaksi antara komponen terintegrasi. Kelompok perangkat lunak yang diuji secara progresif lebih besar yang sesuai dengan desain arsitektur terintegrasi dan diuji sampai perangkat lunak bekerja sebagai sistem. [9] [10].

Untuk Integration test, beberapa hal yang masuk dalam pengujian adalah:

- *Web Base & Mobile Base Integration Test*
- *Rest API Collection System Integration Test*
- *Other System Integration Test,* dimana aplikasi Collection ini diuji secara terintegrasi dengan sistem lain yang memili keterkaitan

System Testing

Ini cenderung untuk menegaskan kualitas end-to-end dari keseluruhan sistem. Uji sistem sering didasarkan pada spesifikasi fungsional atau persyaratan dari sistem. Atribut kualitas non-fungsional, seperti keandalan, keamanan, dan perawatan, juga diperiksa [9].

Acceptance Testing

Penerimaan digunakan untuk melakukan kesiapan operasional suatu produk, layanan atau sistem sebagai bagian dari sistem manajemen mutu. Ini adalah jenis pengujian perangkat lunak non fungsional yang umum digunakan, terutama dalam pengembangan perangkat lunak dan proyek pemeliharaan perangkat lunak. Jenis pengujian ini berfokus pada kesiapan operasional sistem yang akan didukung [9]. Hal itu dilakukan saat sistem selesai diserahkan dari pengembang kepada pelanggan atau pengguna. Tujuan pengujian penerimaan agaknya untuk memberi kepercayaan bahwa sistem tersebut bekerja daripada menemukan kesalahan [10].

Alpha testing

Pengujian alfa adalah simulasi atau pengujian operasional aktual oleh calon pengguna / pelanggan atau tim uji independen di lokasi pengembang. Pengujian alfa sering digunakan untuk perangkat lunak off-the-shelf sebagai bentuk pengujian penerimaan internal, sebelum perangkat lunak masuk ke pengujian beta. [9]

Beta testing

Pengujian beta datang setelah pengujian alpha dan dapat dianggap sebagai bentuk pengujian penerimaan pengguna eksternal. Versi perangkat lunak, yang dikenal sebagai versi beta, dilepaskan ke pemirsa terbatas di luar tim pemrograman. Perangkat lunak ini dilepaskan ke kelompok orang sehingga pengujian lebih lanjut dapat memastikan produk tersebut memiliki sedikit kesalahan atau bug.
Terkadang, versi beta tersedia bagi masyarakat terbuka untuk meningkatkan bidang umpan balik ke jumlah pengguna masa depan yang maksimal. [9]

Regression testing

Ini berfokus pada menemukan cacat setelah perubahan kode utama terjadi. Secara khusus, ia berusaha untuk menemukan regresi perangkat lunak, atau bug lama yang telah kembali. Regresi semacam itu terjadi bilamana fungsi perangkat lunak yang sebelumnya bekerja dengan benar berhenti berfungsi sebagaimana mestinya. Biasanya, regresi terjadi sebagai konsekuensi program yang tidak diinginkan, ketika bagian perangkat lunak yang baru dikembangkan bertabrakan dengan kode yang ada sebelumnya.

Pokok pengujian

Beberapa hal yang harus dimasukkan dalam dokumentasi hasil *Quality Assurance* adalah

berupa *Test Plan* dan *Testing Result* yang terdisi dari:

- *Testing Type* diinput dengan: *Unit Test/Integration Test / System Test/ Acceptance Test*
- *Testing ID*: nomor pengujian yang sifatnya unik
- *Testing Category*: *Web* atau *Mobile* atau *Rest API*, *Integration* dengan Aplikasi Lain
- *Modul Test*: modul dari aplikasi yang akan diuji
- *Test Description*: keterangan dari hal hal yang berhubungan dengan proses testing
- *Data*: data yang dipergunakan dalam proses pengujian
- *Expected Result*: hasil yang diharapkan
- *Actual Result*: hasil yang didapat
- *Remarks*: catatan perbaikan
- *Condition*: pilihan *Completed* atau *To be execution*
- *Status :* pilihan *Passed* atau *Failed*
- *Scenario Date:* tanggal skenario pengujian
- *Test Date:* tanggal pengujian
- *Scenario By:* penyusun skenario pengujian
- *Test By:* yang melakukan pengujian
- *Testing Number*: jumlah pengujian

Hasil dari pengujian atau *Quality Assurance* selanjutnya diserahkan kepada team pengembang untuk dapat segera disesuaikan kembali. Setelah proses perbaikan dan penyesuaian, sistem akan kembali dilakukan tahap pengujian kembali, sampai dengan aplikasi benar-benar telah siap untuk diimplementasikan.

5. KESIMPULAN

1) Pengujian adalah proses untuk mengevaluasi kualitas perangkat lunak. Berbagai jenis pengujian digunakan untuk menguji perangkat lunak. Ada ruang lingkup untuk otomasi dalam kegiatan pengujian tapi pengalaman penguji sangat penting untuk keberhasilan pengujian. Pengujian perangkat lunak adalah komponen pengendalian kualitas perangkat lunak. Berbagai jenis tes digunakan untuk pengujian seperti pengujian unit, pengujian integrasi, pengujian penerimaan, pengujian sistem yang digunakan untuk menguji sistem.
2) Teknik pengujian seperti pengujian statis, pengujian fungsional, pengujian dinamis dan pengujian struktural telah digunakan untuk menguji sistem.
3) Pengujian dapat menunjukkan adanya kesalahan pada suatu sistem; Tidak bisa dibuktikan tidak ada sisa kesalahan.
4) Komponen pengembang bertanggung jawab untuk pengujian komponen; Pengujian sistem adalah tanggung jawab tim yang terpisah.
5) Pengujian integrasi menguji penambahan sistem; Pelepasan pengujian melibatkan pengujian sistem yang akan dilepaskan ke pelanggan.
6) Gunakan pengalaman dan panduan untuk merancang kasus uji dalam pengujian cacat.
7) Antarmuka pengujian dirancang untuk menemukan cacat pada antarmuka komponen komposit.
8) Equivalence partitioning adalah cara untuk menemukan kasus uji - semua kasus dalam partisi harus berperilaku dengan cara yang sama.
9) Analisis struktural bergantung pada analisis sebuah program dan memperoleh tes dari analisis ini.
10) Uji otomasi mengurangi biaya pengujian dengan mendukung proses pengujian dengan berbagai perangkat lunak

DAFTAR PUSTAKA

[1]. Wibisono,W & Baskoro,F (2002, July). Pengujian Perangkat Lunak dengan Menggunakan Model Behaviour UML ,Vol.1 No.1, pp.43.

[2]. Dustin, E.(2003). *Effective Software Testing*, Canada: Pearson Education, Inc.

[3]. Tjandra. S & Pickerling. C.(2015). Aplikasi Metode-metode Software Testing pada *Configuration, Compatibility* dan *Usability* Perangkat Lunak. Makalah di presentasikan pada Seminar Nasional IDeaTech, Surabaya.
[4]. Ghuman S Singh. (2014, October). *Software Testing Techniques,* IJCSMC, Vol. 3, Issue. 10, October 2014, pg.988 – 993.
[5]. Naik, K. & Tripathy, P.(2008). *Software Testing And Quality Assurance Theory And Practice*, Canada:John Wiley & Sons, Inc.
[6]. Myers, Glenford J.(2004). *The art software Testing*, Canada: John Wiley & Sons, Inc.
[7]. M. Ehmer Khan, "*Different Approaches to White Box Testing Technique for Finding Errors," International Journal of Software Engineering and its Application*, vol. 5 No.3, pp. 1-14, 2011.
[8]. M. Ehmer Khan, "*Different Approach to Black box Testing Technique for finding Errors*," *International Journal of Software Engineering & Application*, vol. 2 No.4, pp. 1-10, 2011.
[9]. https://en.wikipedia.org/wiki/Software_testing
[10]. Lu Luo ,“ Software Testing Techniques Technology Maturation and Research Strategy”, Class Report for 17-939A