E-ISSN 2460-9552
P-ISSN 2620-3227

Article history
Received May 4, 2018
Accepted July 2, 2018

EVOLUSI , TANTANGAN , ALAT DAN FRAMEWORK MOBILE APPLICATION : SEBUAH TINJAUAN PUSTAKA

Meilvin Wijaya[1)], Ade Kurniawan[2)]
[1,2)] Program Studi Teknik Informatika, Universitas Universal
Komplek Maha Vihara Duta Maitreya, Sungai Panas, Batam, Kepulauan Riau, Indonesia 29456

Abstract

Mobile apps have come in dramatically in various fields and have changed people's lives. Developing effective mobile applications has become a significant problem for companies today to expand services or generate, and build a direct relationship with customers. Developing a mobile app requires challenging and testing various aspects. To deepen the knowledge of mobile apps requires a literature review. This paper provides basic knowledge about mobile applications and then continued evolution of mobile applications and mobile application challenges continued testing strategies and tools and frameworks for automation of android testing

Keywords: *software development, mobile application, software testing*

Abstrak

Aplikasi seluler telah masuk secara dramatis di berbagai bidang dan telah mengubah kehidupan orang-orang. Mengembangkan aplikasi mobile yang efektif telah menjadi masalah yang signifikan bagi perusahaan saat ini untuk memperluas layanan atau menghasilkan, dan membangun hubungan langsung dengan pelanggan. Untuk mengembangkan aplikasi seluler membutuhkan tantangan dan pengujian berbagai aspek. Untuk memperdalam pengetahuan tentang aplikasi seluler dibutuhkan tinjauan literatur. Makalah ini memberikan pengetahuan dasar tentang aplikasi seluler dan kemudian dilanjutkan evolusi aplikasi mobile dan tantangan aplikasi seluler dilanjutkan strategi pengujian dan Alat dan frameworks untuk otomasi pengujian android

Kata Kunci: pengembangan perangkat lunak, aplikasi mobile, pengujian perangkat lunak.

1. PENDAHULUAN

Aplikasi Mobile dikategorikan sebagai sebuah perangkat lunak yang dirancang untuk melakukan tugas tertentu untuk pengguna ponsel . Aplikasi mobile pertama muncul dan meningkat secara eksponensial dengan pembukaan platform AppStore pada 30 Juli 2008[1]. Dari perkembangan tersebut , jumlah aplikasi dari pihak ketiga yang tersedia di AppStore terus berkembang dari ratusan sampai puluhan juta dan pada tanggal 30 Juli 2011 tercatat dengan jumlah 15 miliar unduhan sejak peluncuran toko[2] .Setelah peluncuran AppStore , beberapa platform distribusi digital mulai muncul dan juga menyediakan mobile software ke perangkat mobile . Platform tersebut seperti Android , Plam webOS, Blackberry OS , Symbias OS , Windows Phone , dll.[3]. Keberhasilan pasar dimana pengguna mobile dapat mencari dan mendownload aplikasi yang tersedia baik gratis maupun dengan biaya , permintaan akan aplikasi mobile terus bertambah , dan menyebabkan semakin banyak developer membuat aplikasi mereka . Terlebih dikarenakan tren penggunaan aplikasi diantara bisnis dan industri , permintaan project yang meningkat . Sarana yang disediakan aplikasi mobile untuk mengakses aplikasi dari lokasi terpencil , untuk berbagi file dengan rekan kerja di seluruh dunia , sebagai alat produktivitas yang lebih baik , dan untuk mendukung pekerja mobile yang semakin banyak.[4]

Ada 2 jenis aplikasi aplikasi mobile : aplikasi yang harus dipasang di perangkat mobile , termasuk yang sudah terpasang dari publisher atau

E-ISSN 2460-9552
P-ISSN 2620-3227

didapatkan dari Store, dan aplikasi mobileWeb . Untuk jenis pertama dapat dikategorikan lebih dalam seperti aplikasi yang dituju untuk jenis handset tertentu seperti iPhone aplikasi yang ditargetkan khusus hanya untuk perangkat iPhone dan aplikasi yang mungkin berjalan di banyak handset , biasanya ditulis dalam bahasa Java. Sedangkan aplikasi mobileWeb berada pada server dan user mengakses aplikasi melalui browser Web yang terhubung dengan jaringan internet . Dalam aspek ini , aplikasi tidak jauh berbeda dengan aplikasi tradisional yang ada pada computer desktop dan biasanya menggunakan teknologi web yang sama seperti HTML , CSS , Java[5]

Dalam beberapa tahun terakhir , perkembangan aplikasi mobile berbasis Android mengalami pertumbuhan yang sangat pesat , untuk itu diperlukan teknik pengujian , strategi dan alat yang efektif dalam pembuatan aplikasi . Tercatat system operasi Android sebagai OS mobile terpopuler kedua , melebihi OS blackberry dan iPhone di tahun 2011[4]. Faktor penentu kesuksesan ini didukung oleh factor kehandalan dalam aplikasi mobile yang strategis dan berkelanjutan .

Aplikasi Android perlu diuji mulai dari beberapa masalah umum dan spesifik . Sebagai contoh , sebagia besar developer tidak terbiasa dengan platform pengembangan Android , sehingga aplikasi rentan terhadap jenis bug baru . Secara umum , aplikasi Android berbeda dengan aplikasi client – server serta GUI desktop . Struktur pusat aplikasi Android bukan sekedar di komponen software tertentu yang ditawarkan oleh framework aplikasi Android yang memerlukan management rules dan lifecycle tertentu[6]

Terlebih dikarenakan platform pengembangan Android masih baru dan belum matang sepenuhnya . Kematangan ini yang memerlukan kegiatan pengujian yang akurat untuk menjamin kualitas aplikasi yang diproduksi. Aplikasi mobile memiliki batasan spesifik dari perangkat genggam yang dapat mempengaruhi strategi dan pengujian untuk aplikasi Android . Contoh , sumber daya perangkat yang terbatas , keamanan yang mencegah serangan yang mempengaruhi rancangan dan pelaksanaan kegiatan pengujian.[5].

2. EVOLUSI APLIKASI MOBILE

Kesuksesan aplikasi mobile dalam beberapa tahun terakhir dengan perbaikan jaringan kemajuan dari sisi hardware di bidang mobile mengalami perkembangan yang pesat. Kita kembali mengingat bahwa pada awalnya telepon seluler hanya digunakan untuk telepon dan mengirim pesan singkat , sementara pada saat ini smartphone yang ada sudah memiliki kemampuan seperti computer pada umumnya . Perusahaan smartphone sekarang berlomba – lomba melengkapi perangkat mereka dengan prosesor yang semakin kuat , memori yang lebih besar dan lebih cepat , layar dengar ratio yang lebih besar , sensitivitas layar sentuh yang canggih , konektivitas nirkabel berkecepatan tinggi . Selain itu perangkat sekarang mengintegrasikan seperangkat sensor perangkat keras juga, seperti penerima GPS, akselerometer, kompas magnetik, dan penerima radio.

Selain kemajuan dari sisi perangkat keras , pengembangan perangkat lunak juga berubah secara signifikan . Pada awalnya , karena keterbatasan perangkat , platform software untuk perangkat mobile tidak menawarkan aplikasi ke mekanisme abstraksi software apapun seperti system operasi untuk menyembunyikan rincian hardware yang mendasarinya. Sebaliknya , smartphone modern sekarang memanfaatkan platform termasuk tidak hanya sistem operasi, tapi juga kerangka kerja aplikasi, perpustakaan, dan beberapa alat pendukung lainnya, seperti utilitas virtualisasi perangkat keras. Dukungan tambahan ini memungkinkan pendekatan pengembangan yang lebih tepat.[7]

Pengembangan aplikasi mobile menjadi semakin mirip dengan pengembangan aplikasi desktop , hal ini dikarenakan untuk keahlian untuk mengembangkan aplikasi mobile lebih sedikit daripada aplikasi desktop . Pengembangan aplikasi mobile mengalami lonjakan yang signifikan dikarenakan ketersediaan platform terbuka seperti Android , ketersediaan alat pengembangan seperti Xamarin dan Android Studio , utilitas gratis , serta ketersediaan pasar aplikasi seperti Google Play Store atau AppStore membuat pengembang pihak ketiga berlomba-lomba berkontribusi pada difusi proyek perangkat lunak kecil atau bahkan sangat kecil yang memperluas kemampuan fungsional perangkat mobile dan pada akhirnya berkontribusi pada kesuksesan komersial mereka.

Menjembatani kesenjangan antara komputer desktop dan perangkat genggam merupakan tantangan utama yang diteliti dalam aplikasi mobile. Menurut guru Android Andy Rubin, "Seharusnya tidak ada hal yang pengguna dapat

E-ISSN 2460-9552
P-ISSN 2620-3227

akses melalui computer tetapi tidak dapat diakses melalui ponsel mereka[4]."

Dengan skenario yang menggembirakan untuk aplikasi mobile ini, proses pengembangan harus mempertimbangkan beberapa implikasi dan tantangan yang relevan.

Pengujian software tetap menjadi salah satu aktivitas yang paling penting dan mahal dalam siklus hidup perangkat lunak. Namun, pengujian aplikasi mobile bisa jadi lebih kompleks karena fitur dan isu spesifik yang menjadi ciri aplikasi sendiri. Kami menganalisis tantangan pengujian aplikasi mobile dan menawarkan perspektif pengujian yang spesifik pada bagian selanjutnya.

3. TANTANGAN PENGUJIAN DAN PERSPEKTIF PENGUJIAN APLIKASI MOBILE

Dua karakteristik utama yang mempengaruhi strategi pengembangan dan pengujian aplikasi mobile adalah:

1. Heterogenitas konfigurasi perangkat keras perangkat mobile
2. Variabilitas kondisi operasinya.

Perangkat mobile komersial dilengkapi dengan berbagai jenis dan jumlah sensor perangkat keras seperti GPS, kompas magnetik, akselerometer, dll., Berbagai jenis tampilan layar memiliki ukuran, karakteristik fisik, dan kemampuan antarmuka pengguna yang berbeda, CPU dengan karakteristik berbeda,dll. Selain itu, perangkat yang mendukung berbagai jaringan seluler dan teknologi komunikasi seperti infrared, Bluetooth, dll. Faktor-faktor ini menyiratkan bahwa aplikasi seluler memerlukan pengembangan dan pengujian cross-platform.

Karena keterbatasan perangkat mobile, tidak praktis menjalankan alat uji secara langsung pada perangkat mobile. Akibatnya, developer sering melakukan pengujian aplikasi dalam dua langkah yang berbeda. Pertama, mereka menguji aplikasi pada desktop atau mesin pengembangan lainnya yang menggunakan emulator, lalu mereka harus memasang dan menguji perangkat seluler target itu sendiri, karena emulator tidak dapat memastikan kompatibilitas penuh dengan perangkat target.[8]

Keterbatasan sumber daya dari platform hardware merupakan tantangan tambahan dalam pengembangan aplikasi mobile modern. Bahkan dalam beberapa tahun terakhir, keterbatasan yang mencakup kemampuan pemrosesan, pembatasan memori, dan power supply, serta fitur antarmuka pengguna . Keterbatasan sumber daya ini memerlukan aktivitas pengujian khusus yang dirancang untuk mengungkapkan kegagalan dalam perilaku aplikasi karena ketersediaan sumber daya. Misalnya, tes harus mengevaluasi aplikasi dalam kondisi penggunaan memori tinggi, tingkat baterai rendah, dan dengan beberapa proses yang berjalan bersaing, untuk mengekspos kemungkinan kegagalan karena penggunaan sumber daya yang ekstrim.

Demikian pula, kemungkinan keterbatasan jaringan nirkabel seperti bandwidth yang berkurang dan ketidakstabilan koneksi juga mempengaruhi pengalaman pengguna dengan aplikasi mobile. Seperti keterbatasan sumber daya perangkat keras yang disebutkan di atas, kegiatan pengujian khusus seperti pengujian mobilitas dan kegunaan harus memenuhi batasan jaringan yang diharapkan .[8]

Pemaparan aplikasi mobile terhadap serangan keamanan juga memerlukan perhatian dari kegiatan pengujian. Karena sifat terbuka platform seluler dan pasar yang memungkinkan pengguna untuk mendownload, menginstal, dan menjalankan aplikasi dengan mudah, data sensitif dan fungsi perangkat yang hebat merupakan sasaran empuk untuk aplikasi yang berjalan di lingkungan yang sama. Kegiatan pengujian keamanan harus mengungkapkan potensi kerentanan aplikasi mobile.

Semua faktor ini menghasilkan tantangan baru untuk proses penjaminan mutu dan pengujian. Meskipun aplikasi mobile memerlukan penekanan pada aktivitas tertentu, pengujian aplikasi mobile didasarkan pada prinsip umum yang mengatur pendekatan pengujian apapun.

Secara khusus, pengujian bergantung pada aspek dasar terkenal yang sama [9], seperti:

- Test Model , mewakili hubungan antara elemen representasi abstrak atau implementasi aktual komponen perangkat lunak;
- Test Levels, menentukan cakupan yang berbeda dari tes yang akan dijalankan, yaitu kumpulan komponen yang akan diuji di setiap tingkat;
- Test Strategies , mendefinisikan heuristik atau algoritma untuk membuat kasus uji dari model representasi perangkat lunak, model implementasi, atau model uji;

E-ISSN 2460-9552
P-ISSN 2620-3227

- Testing Processes, menentukan aliran kegiatan pengujian, dan keputusan lain mengenai kapan pengujian harus dimulai, siapa yang harus melakukan pengujian, berapa banyak usaha yang harus dilakukan, dan masalah serupa.

Selain itu, untuk mengatasi kompleksitas dan tantangan yang melekat dalam pengujian aplikasi mobile, proses pengujian memerlukan pertimbangan berbagai perspektif. Misalnya, persyaratan non fungsional dan fungsional memerlukan pengujian. Perspektif pertama memverifikasi kesesuaian aplikasi dengan persyaratan non-fungsional yang ditentukan seperti kinerja, kegunaan, dan keamanan. Perspektif selanjutnya menguji persyaratan fungsional aplikasi, seperti fungsinya yang mendasar. Yang jelas, menguji aplikasi mobile membutuhkan pertimbangan kedua perspektif tersebut.

4. STRATEGI PENGUJIAN

Strategi pengujian mendefinisikan pendekatan untuk merancang kasus uji. Kami dapat mengkategorikan strategi pengujian secara luas sebagai responsibility-based , implementation-based, atau hybrid. Strategi dapat menargetkan berbagai tingkat pengujian, seperti unit, integrasi, atau pengujian sistem. Baru-baru ini, para peneliti telah menyajikan beberapa strategi untuk menguji aplikasi Android dalam literatur. Beberapa teknik berbasis kejadian menguji GUI aplikasi Android [10]. Pendekatan lainnya, yang disebut teknik berbasis GUI, menggunakan GUI untuk mendefinisikan tes sistem.

4.1 Strategi Event-Based Testing

Aplikasi Android sebenarnya masuk ke kategori aplikasi Event Driven Software (EDS) yang jauh lebih besar — aplikasi yang didorong oleh beberapa jenis acara. Seperti halnya EDS, mereka mengambil peristiwa yang dihasilkan oleh pengguna dan / atau dihasilkan sistem sebagai input, mengubah pola awal, dan (opsional) mengeluarkan urutan peristiwa. Menulis EDS biasanya berpusat pada implementasi kumpulan penangan kejadian yang dirancang untuk menanggapi peristiwa individu. Beberapa faktor membuat pengujian kompleks EDS. Pertanyaan pertama mempertimbangkan definisi oracle uji untuk menentukan apakah suatu kasus uji diterapkan pada aplikasi yang diuji telah berjalan seperti yang diharapkan[11], [12] .Secara tambahan, mendefinisikan representasi dari EDS uji kasus sebagai urutan kejadian dan menemukan teknik yang efektif untuk secara otomatis menghasilkan dan secara otomatis EDS menguji kasus baik tantangan ini.[13] Pendekatan khas yang digunakan untuk pembuatan tes kasus otomatis di EDS melibatkan penciptaan, otomatis atau tidak, dari model abstrak dari aplikasi yang diuji dan menggunakan model untuk menghasilkan kasus uji

Dengan pendekatan ini, peneliti saat ini mempertimbangkan definisi teknik reverse engineering yang sesuai untuk memperoleh model ini sebagai bagian dari proses pengujian EDS . Teknik dalam literatur untuk menguji beberapa jenis sistem EDS tradisional, seperti perangkat lunak berbasis GUI, Rich Internet Applications ,dan perangkat lunak embedded. Kita dapat mengadaptasi teknik ini dari pengujian sistem EDS tradisional tradisional ke aplikasi Android mobile dengan mempertimbangkan konteks aplikasi Android. Ingat bahwa klasifikasi bug menyertakan beberapa jenis cacat yang dapat ditemukan di aplikasi Android, seperti Aktivitas, Event, jenis dinamis, API, I / O, dan kesalahan konkurensi, serta pengecualian yang tidak tertangani.[4] Teknik pengujian berfokus pada aktivitas,

Peristiwa, dan kesalahan tipe dinamis. Penguji menghasilkan uji kasus untuk setiapAktivitas aplikasi yang diuji, mengeksploitasi kelas pengujian Aktivitas dari kerangka pengujian Android yang bekerja bersama dengan JUnit. Teknik ini menggunakan banyak fitur dari kelas Pengujian aktivitas, termasuk: Pengujian kondisi awal (yang menguji kreasi yang tepat dari aktivitas), Pengujian GUI (yang memeriksa apakah aktivitas bekerja dengan benar sesuai dengan spesifikasi GUI), dan Pengujian Manajemen Negara (yang menguji apakah aplikasi dapat masuk dan keluar dengan benar).

4.2 Strategi GUI-Based Testing

Model-based testing (MBT) pendekatan pertama mendefinisikan model formal yang mendeskripsikan aplikasi pada tingkat detail yang diperlukan untuk pembuatan kasus uji otomatis. Algoritma pembuatan kasus uji mengolah model dengan cara sistematis untuk menghasilkan kasus uji.

Teknik MBT menggambarkan GUI dari aplikasi Android oleh state machine, model yang sangat umum untuk mewakili GUI. State Machine

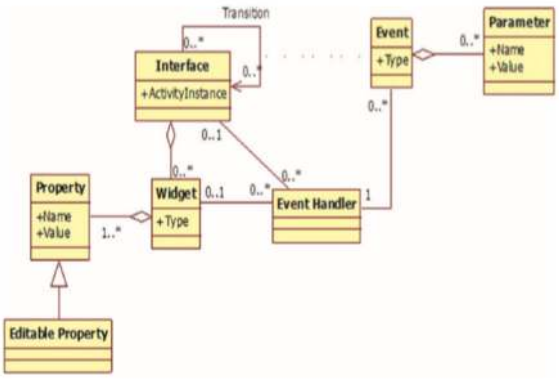
E-ISSN 2460-9552
P-ISSN 2620-3227

terdiri dari states, transisi diantara states, tindakan transisional, dan label yang dilampirkan states. Untuk mengatasi kompleksitas state machine yang mewakili aplikasi ukuran sebenarnya, teknik ini membagi model GUI menjadi komponen model yang lebih kecil. Setiap komponen model sesuai dengan pandangan masing-masing GUI dan dibagi menjadi dua tingkat seperti yang ditentukan oleh dua state machine yang terpisah: mesin aksi dan mesin perbaikan. Mesin aksi menjelaskan fungsionalitas tingkat tinggi menggunakan action words dan state verifications. Sebuah kata aksi, yang dimodelkan sebagai tindakan yang terkait dengan transisi, menjelaskan kasus penggunaan kecil dari aplikasi seperti menyimpan file. State verification yang dimodelkan sebagai label dalam suatu negara, menjelaskan keadaan aplikasi untuk memverifikasi bahwa status aplikasi sebenarnya sesuai dengan keadaan model. Mesin penyempurnaan menggambarkan kata-kata tindakan dan verifikasi negara menggunakan kata kunci. Penguji harus secara manual menghasilkan kedua model.

Pembuatan kasus uji bergantung pada model dan kata kunci. Karena beberapa kata kunci menggambarkan peristiwa pengguna biasa sementara kata kunci lain memverifikasi status aplikasi, kata kunci dapat digunakan untuk mendefinisi dan mengeksekusi kasus uji dengan alat otomatisasi uji.

Alat desain tes termasuk GUI Web untuk merancang tujuan pengujian. Alat pembuat tes menyediakan sejumlah algoritme yang menggunakan model dan tujuan pengujian untuk menghasilkan kasus uji sebenarnya. Alat debugging pengujian di TEMA membantu menginterpretasikan uji coba yang gagal. Akhirnya, satu-satunya alat yang bergantung pada platform Android di toolset mendukung eksekusi otomatis dari kasus uji.

Crawler menghasilkan model GUI yang sebenarnya adalah struktur pohon yang disebut GUITree. Simpul pohon mewakili antarmuka pengguna individual di aplikasi Android, sementara ujung menggambarkan transisi berbasis acara antar antarmuka. Crawler memperoleh pohon dengan analisis dinamis, merekonstruksi model GUI dengan memadukan acara pada antarmuka pengguna aplikasi. Model ini menjelaskan setiap antarmuka pengguna dalam hal widget komponennya, properti widget, dan pengendali acara, sementara menggambarkan setiap transisi antara antarmuka pengguna berturut-turut oleh peristiwa yang menyebabkan transisi.

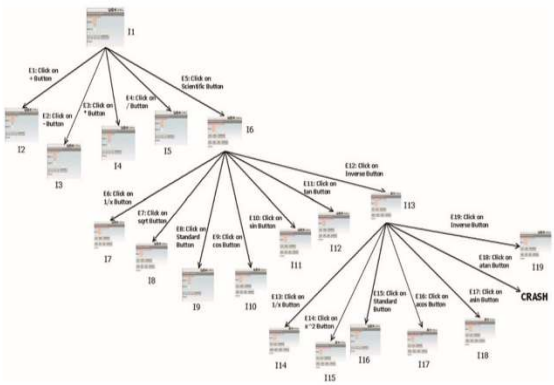


Gambar 1 Conceptual model dari GUITree

Gambar 1 menunjukkan model ini dari Android GUI sebagai diagram kelas.[4]

Kriteria pengakhiran aktivitas eksplorasi GUI merepresentasikan aspek penting dari GUI crawling algorithm apa pun. Kriteria yang sering digunakan mengevaluasi kesetaraan antarmuka pengguna. Eksplorasi GUI berhenti ketika antarmuka saat ini muncul setara dengan antarmuka yang sudah dikunjungi. Algoritma crawling yang diusulkan dalam [14] mengasumsikan dua antarmuka menjadi setara jika mereka memiliki atribut Activity Instance yang sama (lihat model pada Gambar 1) dan set yang sama. Widget dengan Properti dan Penangan Kejadian identik.

Aspek lain yang relevan dari setiap algoritme perayapan GUI terdiri dari menemukan pendekatan untuk mendefinisikan properti widget (seperti nilai bidang masukan) yang harus ditetapkan sebelum memfilter peristiwa pada GUI. Sebagai contoh, perayap disajikan dalam [14] menetapkan nilai acak ke properti. Gambar 2 menunjukkan GUITree dari contoh nyata aplikasi Android.



Gambar 2 GUI Tree diperoleh dengan crawling contoh Aplikasi Android

E-ISSN 2460-9552
P-ISSN 2620-3227

Pohon GUI yang dihasilkan oleh crawler menyediakan titik awal untuk pengujian automatic crash . Menurut Memon dan Xie crash testing bertujuan untuk mengungkap kesalahan aplikasi karena tidak ada pengecualian yang berjalan saat itu [13]. Untuk menjalankan pengujian kerusakan, teknik yang disajikan dalam [14] menggunakan kasus uji yang terdiri dari urutan kejadian yang sesuai dengan jalur pohon GUI (dari simpul akar ke daun pohon). Instrumentasi kode dan pemantauan mendukung deteksi otomatis dari crash selama eksekusi test case.

Kasus uji yang sama yang dihasilkan untuk pengujian kerusakan juga dapat digunakan untuk pengujian regresi. Penguji melakukan pengujian regresi setelah membuat perubahan pada aplikasi yang diuji. Pengujian regresi terdiri dari rerunning tes yang sebelumnya dijalankan dan memeriksa perubahan dalam perilaku program dan munculnya kesalahan baru. Untuk mendeteksi perbedaan, penguji dapat membandingkan urutan antarmuka pengguna yang ditemui dalam kedua uji coba. Perbandingan antarmuka dapat dibuat menggunakan uji oracle memiliki derajat detail atau granularity yang berbeda [12]. Pengujian dapat memeriksa properti Interface seperti komponen Activity, Event Handlers, Widget Properties, danWidget Values. Untuk mendukung pemeriksaan ekstensif ini, kasus uji asli dapat menambahkan pernyataan spesifik sehingga kegagalan tes menunjukkan ketidak konsistenan dalam properti.

4.3. Random Testing Strategies

Urutan GUI atau peristiwa konteks mewakili masukan dari aplikasi seluler. Peristiwa konteks dapat diperoleh dari konteks fisik perangkat (seperti penerima GPS, chip Bluetooth, sensor akselerometer, sensor kelembaban, sensor magnetik, dan jaringan, dll. .), atau dari konteks sosial (seperti teman ngobrol di sekitar, aktivitas pengguna saat ini, dll.). Aplikasi seluler harus menerima dan bereaksi terhadap kedua jenis peristiwa konteks yang terus berubah sebagai masukan untuk menghasilkan output yang benar.

Pembuatan tes kasus acak mengikuti teknik Adaptive Random Testing (ART). Peneliti menyarankan bahwa pendekatan ART yang dapat menawarkan peningkatan keefektifan melalui pengujian acak murni[15]. Pengujian acak, memang, memiliki manfaat biaya rendah sebagai hasil dari generasi acak dari kasus uji dan kemudahan otomatisasi. Sayangnya, meskipun teknik sederhana dan sepenuhnya otomatis, sepenuhnya acak tidak dapat menawarkan jaminan keefektifan dalam mendeteksi kesalahan. ART, sebaliknya, didasarkan pada observasi bahwa wilayah kegagalan input dari suatu aplikasi cenderung mengelompok bersama-sama. Oleh karena itu, ART mencoba untuk menyebarkan kasus-kasus uji yang dihasilkan secara acak serata mungkin dengan menggunakan ukuran jarak test case. Sebuah studi eksperimental awal menunjukkan teknik ART menjadi lebih efektif untuk pembuatan tes kasus daripada pengujian acak dalam hal deteksi kesalahan dalam aplikasi Android.

5. ALAT DAN FRAMEWORKS UNTUK OTOMASI PENGUJIAN ANDROID

Proses pengujian juga dipengaruhi oleh alat dan teknologi untuk otomatisasi pengujian perangkat lunak. Secara umum , alat uji dapat mengotomatisasi beberapa tugas dari proses pengujian , seperti test-case generation , test-case execution , dan evaluasi test-case . Selain itu , alat pengujian dapat mendukung produksi dokumentasi pengujian yang berguna dan memberikan pengelolaan konfigurasi dokumentasi berdampingan dengan source code

Pada aktivitas pengujian terdapat dua kategori utama untuk alat pengujian[4]: alat untuk pengujian kebutuhan fungsional dan alat untuk pengujian kebutuhan non-fungsional , seperti pengujian kinerja, beban, keamanan, atau kegunaan.Dalam beberapa tahun terakhir, para periset telah mengembangkan beberapa alat dan teknologi untuk membantu pengujian aplikasi mobile. Hingga kini, sebagian besar alat dikembangkan untuk platform mobile selain Android. Alat yang ada sering menawarkan fitur berikut:

- Black-Box Testing: lebih dikenal sebagai behavioral testing karena dimana proses jalannya testing tidak diketahui oleh penguji . Black-box testing sendiri digunakan untuk mencari fungsi yang tidak benar atau hilang; Kesalahan antarmuka; Kesalahan dalam struktur data atau akses database eksternal; Perilaku atau kesalahan kinerja.Inisialisasi dan terminasi kesalahan[15]
- White-box Testing: juga dikenal dengan nama lain, seperti pengujian kaca-kotak,pengujian struktural, pengujian kotak jernih, pengujian open-box, pengujian berbasis logika, dan pengujian berorientasi jalur.Pada White-box

E-ISSN 2460-9552
P-ISSN 2620-3227

testing uji kasus dipilih berdasarkan pemeriksaan kode, bukan spesifikasinya. Dengan menggunakan metode pengujian white-box kasus yang dapat diuji oleh tester : Menjamin bahwa semua jalur independen dalam modul telah dilakukan setidaknya sekali. Melaksanakan semua keputusan logis pada sisi true dan false mereka. Menjalankan semua loop di batasan mereka. Latih struktur data internal untuk memastikan validitasnya.[15]

Seperti aplikasi mobile yang dikembangkan untuk platform Android,Platform pengembangan Android menyediakan sumber daya dan teknologi paling besar untuk otomasi . SDK Android dibangun di Java SDK yang berisi semua perpustakaan kode Java yang dibutuhkan untuk membuat aplikasi yang berjalan secara khusus di platform Android. SDK juga menyertakan file bantuan, dokumentasi, Android Emulator, dan alat lainnya untuk pengembangan dan debugging [16], [17]

Open Handset Alliance telah merilis plugin Android untuk Eclipse, ADT[16] (Android Development Tools). Plugin ini mengintegrasikan pembuatan, kompilasi, pengemasan, pengekspor, pengujian, debugging, dan bahkan persaingan proyek khusus Android di lingkungan Eclipse yang sudah dikenal.

Di masa depan, alat tambahan berpotensi membuat otomatisasi pengujian Android lebih layak dan efektif. Sebagai contoh, alat untuk pengujian kebutuhan fungsional aplikasi Android harus menyediakan fitur untuk mengotomatisasi kegiatan pengujian dasar berikut[4]:

- Test Model Generation: untuk menghasilkan sebuah contoh dari model uji yang diinginkan dari aplikasi subjek. Model ini mungkin merupakan model yang telah diproduksi sepanjang proses pengembangan (sehingga alat hanya perlu mengimpornya), atau dapat diproduksi oleh teknik reverse engineering
- Code Instrumentation: untuk memberi kode kode anAndroid secara otomatis dengan memasukkan utilitas untuk mengumpulkan data secara otomatis tentang eksekusi kasus uji.
- Driver and Stub Generation: untuk membantu menghasilkan kode komponen aplikasi Android yang dibutuhkan untuk eksekusi kasus uji, seperti modul driver, stub, dan tiruan.
- Test Case Management: untuk secara otomatis menghasilkan kasus uji dan untuk mendukung desain uji kasus dan dokumentasi pengujian manajemen.
- Test Case Execution: untuk mendukung eksekusi otomatis dan evaluasi kasus uji pada aplikasi Android yang diuji.
- Test Result Analysis: untuk menganalisa dan mengevaluasi secara otomatis hasil uji.
- Report Generation: menghasilkan laporan yang memadai yang menganalisis hasil uji, seperti laporan cakupan tentang komponen yang digunakan selama pengujian.

6. KESIMPULAN

Berkembangnya permintaan untuk aplikasi mobile yang berkualitas terus mendorong minat penelitian di aplikasi mobile. Peran dominan yang diasumsikan oleh aplikasi yang dikembangkan untuk platform mobile terbuka membuat aplikasi Android menjadi target yang masuk akal untuk penelitian ini. Beberapa prakarsa saat ini menangani masalah dalam menemukan prinsip pengujian, pedoman, model, teknik yang memungkinkan proses pengujian yang lebih efektif dan efisien untuk aplikasi ini.

Aspek pengujian Android yang perlu diteliti lebih lanjut dan penekanan di masa depan antara lain:

- Mencari model yang sesuai pengujian Android baik dari sisi system dan komponen yang digunakan
- Mencari teknik yang efektif secara otomatis menemukan test-case dari penelitan.
- Mengembangkan alat khusus untuk membantu langkah-langkah dalam pengujian Android.

7. REFERENSI

D. Amalfitano, A. R. Fasolino, and P. Tramontana, “Techniques and tools for Rich Internet Applications testing,” *2010 12th IEEE Int. Symp. Web Syst. Evol.*, pp. 63–72, 2010.

D. Amalfitano, a. R. Fasolino, and P. Tramontana, “Rich Internet Application Testing Using Execution Trace Data,” *Softw. Testing, Verif. Valid. Work. (ICSTW), 2010 Third Int. Conf.*, pp. 274–283, 2010.

W. F. Ableson, R. Sen, and C. King, *Android in Action*, vol. 36. 2003.

E-ISSN 2460-9552
P-ISSN 2620-3227

A. Memon, *Advances in Computers*, vol. 89. Chichester, UK: John Wiley & Sons, Ltd, 2013.

Information Resources Management Association, *Application development and design : concepts, methodologies, tools, and applications*. Hershey, PA : Engineering Science Reference, 2018.

W. Hu, D. Han, A. Hindle, and K. Wong, "The build dependency perspective of Android's concrete architecture," *IEEE Int. Work. Conf. Min. Softw. Repos.*, pp. 128–131, 2012.

C. Tao, J. Gao, and T. Wang, "An Approach to Mobile Application Testing Based on Natural Language Scripting," no. May, pp. 260–265, 2017.

I. Satoh, "Software testing for wireless mobile computing," *IEEE Wirel. Commun.*, vol. 11, no. 5, pp. 58–64, 2004.

J. D. McGregor and D. A. Sykes, "A practical guide to testing object-oriented software," *Object Technol. Ser.*, p. 393, 2001.

C. Hu and I. Neamtiu, "Automating gui testing for android applications," *Proceeding 6th Int. Work. Autom. Softw. test - AST '11*, no. Section 4, p. 77, 2011.

A. Memon and Q. Xie, "Using transient/persistent errors to develop automated test oracles for event-driven software," *Proc. - 19th Int. Conf. Autom. Softw. Eng. ASE 2004*, pp. 186–195, 2004.

Q. Xie and A. M. Memon, "Designing and comparing automated test oracles for GUI-based software applications," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 1, p. 4–es, 2007.

A. M. Memon and Q. Xie, "Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 884–896, 2005.

D. Amalfitano, A. R. Fasolino, and P. Tramontana, "A GUI Crawling-Based Technique for Android Mobile Application Testing," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011, pp. 252–261.

B. B. Agarwal, S. P. Tayal, and M. (Mahesh) Gupta, *Software engineering and testing: An introduction*. 2010.

R. Sen, "Android in Action." 2012.

T. G. Project, "The Developer ' s Guide," 2011.