URGENSI PENGUJIAN PADA KEMAJEMUKAN PERANGKAT LUNAK DALAM MULTI PERSPEKTIF

Hernawan Sulistyanto[1], Azhari SN[2]

[1]Program Studi Teknik Infomatika, Fak. Komunikasi dan Informatika,
Universitas Muhammadiyah Surakarta
Jl. A. Yani Tromol Pos 1 Pabelan, Surakarta
[2]Program Studi Ilmu Komputer, FMIPA, Universitas Gadjah Mada, Yogyakarta
Sekip Utara, Bulaksumur, Yogyakarta
E-mail: Hernawan.Sulistyanto@ums.ac.id, arisn@ugm.ac.id

ABSTRAK

Pengujian perangkat lunak merupakan suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (*under test*). Pengujian perangkat lunak akan memberikan pandangan mengenai perangkat lunak secara obyektif dan independen yang bermanfaat dalam operasional bisnis untuk memahami tingkat risiko pada implementasinya sebelum disampai kepada pelanggan. Tiga konsep yang perlu diperhatikan dalam pengujian perangkat lunak adalah demonstrasi *validitas* perangkat lunak pada setiap tahapan pembangunan sistem, penentuan *validitas* sistem akhir terhadap pemakai kebutuhan, dan pemeriksaan implementasi sistem dengan menjalankan sistem pada suatu contoh data uji. Implementasi pengujian pada perangkat lunak dapat dilaksanakan sesuai dengan kebutuhan dan model sistem. Keragaman dan kemajemukan perangkat lunak mengisyaratkan pentingnya untuk melaksanakan banyak improvisasi saat pengujian dikerjakan, seperti penentuan kasus uji yang tepat, pemilihan model dan metode pengujian yang sesuai, penentuan lingkungan uji yang cocok, serta mempertimbangkan beberapa aspek lain yang bertujuan mengoptimalkan hasil uji yang diperoleh dalam kerangka menjamin kualitas sebuah produk perangkat lunak.

Kata Kunci : pengujian, perangkat lunak, kualitas produk

A. PENDAHULUAN

Meningkatnya *visibilitas* perangkat lunak sebagai suatu elemen sistem dan "biaya" yang muncul akibat kegagalan perangkat lunak telah memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti. Hal ini menjadikan pengujian perangkat lunak sebagai suatu tahapan penting dalam pembangunan perangkat lunak.

Beberapa alasan perlunya melakukan pengujian yaitu karena kita bukan seorang programmer yang cukup baik, kita mungkin tidak dapat cukup berkonsentrasi untuk menghindari kesalahan, kita kadang-kadang lupa menggunakan pemrograman terstruktur secara penuh, kita kadang buruk dalam mengerjakan sesuatu, serta kita seharusnya dapat membedakan apa yang dikatakan programmer lain atau pelanggan dan apa yang sebenarnya mereka pikirkan.

Pengujian dapat dilakukan dengan cara mengevaluasi kofigurasi perangkat lunak yang terdiri dari spesifikasi kebutuhan, deskripsi perancangan dan program yang dihasilkan. Hasil evaluasi kemudian dibandingkn dengan hasil uji yang diharapkan. Jika ditemukan

kesalahan maka perbaikan perangkat lunak harus dilakukan untuk kemudian diuji kembali. Sehingga pada dasarnya aktivitas pengujian dapat dianggap sebagai hal yang merusak daripada membangun.

Bagaimanapun, pentingnya pengujian perangkat lunak dan implikasinya yang mengacu pada kualitas perangkat lunak tidaklah dapat terlalu ditekan karena melibatkan sederetan aktivitas produksi di mana peluang terjadinya kesalahan manusia sangat besar. Maka sudah seyogyanya pengembangan perangkat lunak diiringi dengan aktivitas penjaminan kualitas.

B. Konsep Pengujian Perangkat Lunak

1. Pengertian Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah proses menjalankan dan mengevaluasi sebuah perangkat lunak secara manual maupun otomatis untuk menguji apakah perangkat lunak sudah memenuhi persyaratan atau belum (Clune dan Rood, 2011) dan (Nakagawa dan Maldonado, 2011). Singkat kata, pengujian adalah aktivitas untuk menemukan dan menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.

Sebuah pengujian perangkat lunak dapat dilakukan setelah perekayasa membangun implementasi dari suatu konsep abstrak perangkat lunak. Pengujian perangkat lunak pada dasarnya adalah bermaksud "membongkar" perangkat lunak yang telah terbangun.

Sesuai dengan Jin dan Xue (2011) dan Kumamoto dkk. (2010) menyatakan bahwa pengujian bermaksud untuk mencari sebanyak mungkin kesalahan yang ada pada program serta mengevaluasi kualitasnya. Tujuan pengujian perangkat lunak menurut Xie dkk.(2011) adalah menilai apakah perangkat lunak yang dikembangkan telah memenuhi kebutuhan pemakai, menilai apakah tahap pengembangan perangkat lunak telah sesuai dengan metodologi yang digunakan, dan membuat dokumentasi hasil pengujian yang menginformasikan kesesuaian perangkat lunak yang diuji dengan spesifikasi yang telah ditentukan. Data yang dikumpulkan pada saat pengujian dilakukan akan memberikan indikasi yang baik mengenai reliabilitas dan kualitas perangkat lunak secara keseluruhan.

2. Objektivitas Pengujian

Menurut Pressman (2010), pengujian perangkat lunak mempunyai beberapa sasaran penting, yaitu (1) pengujian dilaksanakan dengan maksud menemukan kesalahan; (2) kesuksesan pengujian adalah kemampuan dalam menemukan kesalahan yang belum pernah ditemukan sebelumnya; dan (3) kasusu uji yang baik adalah sebuah kasusu ujie yang mempunyai probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.

Objektivitas dalam pengujian dapat dicapai apabila ada beberapa actor yang terlibat selama pengujian, diantaranya menurut Lamas dkk.(2013), yaitu *customer* (tim yang mengontrak pengembang untuk mengembangkan perangkat lunak), pengguna (kelompok yang akan menggunakan perangkat lunak),

pengembang perangkat lunak (tim yang membangun perangkat lunak), dan tim pengujian perangkat lunak (tim khusus yang bertugas untuk menguji fungsi-fungsi pada perangkat lunak).

Disamping itu, juga harus selalu berprinsip bahwa pengujian dapat dilacak hingga ke persyaratan pelanggan, pengujian harus direncanakan sebelum pelaksanaan pengujian, pengujian harus dimulai dari hasil yang kecil kemudian diteruskan ke hal-hal yang besar, pengujian yang berlebihan tidak akan mungkin dilaksanakan, dan pengujian sebaiknya dilakukan oleh pihak ketiga (Jiang dan Lu, 2012) (Lemos dkk., 2011).

3. Tahapan Pengujian

Pelaksanaan pengujian sebuah perangkat lunak biasanya disesuaikan dengan metodologi pembangunan perangkat lunak yang digunakan. Reza (2010) dan Sommerville (2011) menyampaikan bahwa pada umumnya pengujian dilaksanakan setelah tahap pemograman, namun perencanaan pengujian sudah dilakukan mulai tahap analisis. Secara keseluruhan, tahapan dalam pengujian meliputi penentuan apa yang akan diukur, bagaimana pengujian akan dilaksanakan, membangun suatu kasus uji (*test case*) yaitu sekumpulan data atau situasi yang akan digunakan dalam pengujian, kemudian menetapkan hasil yang akan diharapkn atau hasil yang sebenarnya, menjalankan kasus pengujian dan membandingkan hasil pengujian dengan hasil yang diharapkan

Pada pengujian tahap analisis menekankan pada validasi terhadap kebutuhan untuk menjamin bahwa kebutuhan telah dispesifikasi dengan benar. Tujuan pengujian pada tahap ini adalah untuk mendapatkan kebutuhn yang layak dan untuk memastikan apakah kebutuhan tersebut sudah dirumuskan dengan baik. Faktor-faktor pengujian yang dilakukan pada tahap analisis yaitu kebutuhan yang berkaitan dengan metodologi, pendefenisian spesifikasi fungsional, penentuan spesifikasi kegunaan, penentuan kebutuhan portabilitas, dan pendefenisian antar muka sistem.

Pengujian tahap perancangan bertujuan untuk menguji struktur perangkat lunak yang diturunkan dari kebutuhan. Kebutuhan yang bersifat umum dirinci menjadi bentuk yang lebih spesifik. Faktor-faktor pengujian yang dilakukan pada tahap perancangan yaitu perancangan yang berkaitan dgn kebutuhan, kesesuaian perancangan dengan metodologi dan teori, portabilitas rancangan, perancangan perawatan, kebenaran rancangan berkaitan dengan fungsi dan aliran data, dan kelengkapan perancangan antar muka.

Pengujian pada tahap implementasi merupakan pengujian unit-unit yang dibuat sebelum diintegrasi menjadi aplikasi keseluruhan. Faktor-faktor pengujian yang dilakukan pada tahap ini yaitu kendali integritas data, kebenaran program, kemudahan pemakaian, sifat *coupling*, dan pengembangan prosedur operasi.

Pengujian tahap pengujian bertujuan untuk menilai apakah spesifikasi program telah ditulis menjadi instruksi-instruksi yang dapat dijalankan pada mesin dan untuk menilai apakah instruksi yang

ditulis tersebut telah sesuai dengan spesifikasi program. Faktor-faktor pengujian yang dilakukan pada tahap ini meliputi pengujian fungsional, dukungan manual, dan kemudahan operasi.

4. Teknik Pengujian

Pada tahapan pengujian diperlukan suatu kasusu uji. Kasus uji didesain dengan sasaran utama untuk mendapatkan serangkaian pengujian yang memiliki kemungkinan tertinggi di dalam mengungkap kesalahan pada perangkat lunak sebagaimana dinyatakan oleh Pressman (2010) dan Sommerville (2011). Pengujian dengan kasusu uji meliputi pengujian unit (berupa prosedur atau fungsi) dan pengujian sistem. Dalam pengujian unit, unit-unit yang diuji meliputi unit-unit yang ada dalam sistem,sedangkan pengujian sistem dilakukan terhadap sistem secara keseluruhan. Setiap pengujian dilakukan dengan menggunakan berbagai data masukan yang valid maupun tidak.

Mengacu pada Wen-hong dan Xin (2010), produk hasil rekayasa dapat diuji dengan cara: (1) mengetahui fungsi yang ditentukan dimana produk dirancang untuk melakukannya. Pengujian dilakukan untuk memastikan bahwa masing-masing fungsi beroperasi dengan sepenuhnya dan mencari kesalahan pada tiap fungsi; (2) mengetahui kerja internal untuk memastikan bahwa komponen internal bekerja sesuai dengan spesifikasi. Sehingga dalam hal ini terdapat dua jenis kasus uji, yaitu pertaman pengetahuan fungsi yang spesifik dari produk yang telah dirancang untuk diperlihatkan, test dapat dilakukan untuk menilai masing-masing fungsi apakah telah berjalan sebagaimana yang diharapkan. Kedua, pengetahuan tentang cara kerja dari produk, tes dapat dilakukan untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya. Ada dua macam pendekatan kasus uji yaitu *white-box* dan *black-box*. Pendekatan *white-box* adalah pengujian untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya (Jiang, 2012) (Pressman, 2010). Jalur logika perangkat lunak akan dites dengan menyediakan kasus uji yang akan mengerjakan kumpulan kondisi dan pengulangan secara spesifik. Sehingga melalui penggunaan metode ini akan dapat memperoleh kasus uji yang menjamin bahwa semua jalur independen pada suatu model telah diigunakan minimal satu kali, penggunaan keputusan logis pada sisi benar dan salah, pengeksekusian semua loop dalam batasan dan batas operasional perekayasa, serta penggunaan struktur data internal guna menjamin validitasnya. Secara sekilas dapat diambil kesimpulan pendekatan pengujian *white-box* mengarah untuk mendapatkan program yang benar secara 100%.

Pendekatan *black-box* merupakan pendekatan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan (Jiang, 2012) (Pressman, 2010). Kasus uji ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya. Teknik pengujian ini berfokus pada domain informasi dari perangkat lunak, yaitu

melakukan kasus uji dengan mempartisi domain *input* dan *output* program. Metode *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian ini berusaha menemukan kesalahan dalam kategori fungsi-fungsi yang tidak benar atau hilang, kesalahan *interface*, kesalahan dalam struktur data atau akses basis data eksternal, kesalahan kinerja, dan inisialisasi dan kesalahan terminasi.

5. Strategi Pengujian

Strategi pengujian perangkat lunak memudahkan para perancang untuk menentukan keberhasilan sistem rancangan. Beberapa hal yang perlu mendapat perhatian adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan memperhitungkan berapa lama waktu, upaya dan sumber daya yang diperlukan. Karakteristik strategi pengujian meliputi pengujian diawali pada tingkat modul yang paling bawah, dilanjutkan dengan modul di atasnya kemudian hasilnya dipadukan, teknik pengujian yang berbeda mungkin menghasilkan sedikit perbedaan (dalam hal waktu). Pengujian dan *debugging* merupakan aktivitas yang berbeda, tetapi *debugging* termasuk dalam strategi pengujian. *Debugging* prinsipnya memperbaiki *error* yang ditemukan pada saat pengujian (yang sukses).

Pengujian perangkat lunak adalah satu elemen dari topic yang lebih luas yang sering disebut sebagai verifikasi dan validasi. Verifikasi merupakan kumpulan aktivitas yang menjamin penerapan perangkat lunak benar-benar sesuai dengan fungsinya. Sementara validasi merupakan kumpulan aktivitas yang berbeda yang memastikan bahwa perangkat lunak yang dibangun dapat memenuhi keperluan pelanggan. Atau dengan kata lain verifikasi adalah *"Apakah kita membuat produk dengan benar?"* dan validasi adalah *" Apakah kita membuat benar-benar suatu produk?"*.

C. IMPLEMENTASI PENGUJIAN

1. Pengujian Validasi Perangkat Lunak

Pengujian validasi dilaksanakan setelah semua kesalahan diperbaiki. Indikator keberhasilan pengujian validasi adalah jika fungsi yang ada pada perangkat lunak sesuai dengan yang diharapkan pemakai. Apabila perangkat lunak dibuat untuk pelanggan maka dapat dilakukan semacam *aceeptance test* sehingga memungkinkan pelanggan untuk memvalidasi seluruh keperluan. Uji ini dilakukan agar memungkinkan pelanggan menemukan kesalahan yang lebih rinci dan membiasakan pelanggan memahami perangkat lunak yang telah dibuat. Bentuk pengujian yang bisa dilaksanakan yaitu pengujian *alpha* dan *beta*. Pengujian *alpha* dilakukan pada sisi pengembang oleh seorang pelanggan (Pressman, 2010). Perangkat lunak digunakan pada setting yang natural dengan pengembang "yang memandang" melalui bahu pemakai dan merekam semua kesalahan dan masalah pemakaian. Sedangkan pengujan *beta* dilakukan pada satu atau lebih pelanggan oleh pemakai akhir perangkat lunak dalam lingkungan yang sebenarnya.

Pengembang biasanya tidak ada pada pengujian ini. Pelanggan merekan semua masalah (*real* atau *imajiner*) yang ditemui selama pengujian dan melaporkan pada pengembang pada interval waktu tertentu (Pressman, 2010).

2. Pengujian Sistem

Pada akhirnya produk perangkat lunak digabungkan dengan elemen system lainnya dan kemudian rentetan tes validasi dilakukan. Jika uji coba gagal atau di luar skope dari daur siklus pengembangan system, langkah yg diambil selama perancangan dan pengujian dapat diperbaiki. Pengujian sistem merupakan rentetan pengujian yang berbeda-beda dengan tujuan utama mengerjakan keseluruhan elemen system yang dikembangkan. Beberapa jenis pengujian system sesuai Pressman (2010) diantaranya *recovery testing, security testing,* dan *stress testing*.

3. Pengujian untuk Aplikasi dan Lingkungan Khusus

Perangkat lunak selalu diimplementasikan dalam suatu aplikasi dan lingkungan yang berbeda. Karenanya terdapat pengujian tersendiri bagi masing-masing system tersebut (Wu, 2010). Pengujian yang dapat dikerjakan dapat diklasifikasikan kedalam bentuk pengujian GUI, pengujian system *client-server*, dan pengujian system waktu nyata. Pengujian GUI sesuai Pressman (2010) untuk windows terdiri atas beberapa langkah standard yaitu menguji apakah window akan membuka secara tepat berdasarkan tipe yang sesuai atau perintah berbasis menu, dapatkah window diresize atau digulung, apakah semua isi data yang diisikan pada window dapat dituju dengan tepat dengan sebuah *mouse*, *function keys*, anak panah penunjuk dan *keyboard*, apakah window dengan cepat muncul kembali apabila dia ditindih dan kemudian dipanggil lag, apakah semua fungsi yang berhubungan dengan window dapat diperoleh bila diperlukan, apakah semua fungsi yang berhubungan window operasional, apakah semua menu *pull-down, scroll bar, tool bar* kotak dialog, tombol, ikon, dapat diperoleh dan dengan tepat ditampilkan untuk window tersebut, pada saat window bertingkat ditampilkan apakah nama window tersebut direpresentasikan secara tepat.

Pengujian perangkat lunak yang menggunakan aplikasi *client-server* umumnya dilaksanakan pada tiga tingkat berbeda, menurut Pressman (2010) yaitu: (1) aplikasi *client* secara individual diuji dalam kondisi tak terhubung (*disconnected*), artinya tidak memperhatikan pengoperasian *server* dan jaringan yang membawahinya; (2) perangkat lunak *client* dan aplikasi *server* terkaitnya diuji bersama-sama, tetapi pengoperasian jaringannya tidak dijalankan sepenuhnya; (3) arsitektur *client-server* seutuhnya termasuk operasi jaringan dan penampilannya diuji. Pengujian aplikasi *client-server* yang umum dijumpai yaitu pertama tes fungsi aplikasi. Fungsi aplikasi *client* diuji dalam model standard untuk menemukan kesalahan pengoperasiannya. Kedua, tes server. Pengujian dilakukan pada koordinasi dan fungsi manajemen datadi server termasuk kinerja server (*respon time* dan *throughput* keseluruhan). Ketiga tes basis data. Pengujian dilakukan pada keakuratan

atau ketepatan dan integritas data yang tersimpan dalam server. Transaksi yang dilakukan pada aplikasi client diperiksa guna memastikan bahwa data sudah tersimpan dengan benar. Pemanggilan kembali data dan pengarsipan juga diuji. Keempat, pengujian transaksi. Pengujian ini dilaksanakan dengan membuat serangkaian tes guna memastikan bahwa masing-masing kelas transaksi diproses menurut *requirement*nya. Kelima, pengujian komunikasi jaringan. Pengujian ini untuk mengetes keberlangsungan komunikasi antar-node jaringan berlangsung dengan benar serta mengetahui apakah pengiriman pesan, transaksi berlangsung tanpa kesalahan. Tes keamanan jaringan dapat termasuk dalam pengujian ini.

Strategi pengujian bagi sistem waktu nyata *(real-time)* menurut Pressman (2010) dan Zhang (2011) meliputi yang pertama adalah pengujian tugas. Maksud langkah ini adalah untuk menguji masing-masing tugas secara independen, yaitu pengujian *white-box dan black-box* yang didesain dan dieksekusi bagi masing-masing tugas. Masing-masing tugas dieksekusi secara independen dan berusaha mengungkap kesalahan di dalam logika dan fungsi. Selanjutnya adalah pengujian tingkah laku. Pengujian dimaksudkan untuk men-simulasi tingkah laku sistem real-time dann meguji tingkah lakunya sebagai konsekuensi dari *event* eksternal. Perilaku diuji untuk mendeteksi kesalahan perilaku. Berikutnya adalah pengujian antar tugas. Pengujian dilaksanakan setelah kesalahan pada tugas individual dan pada perilaku sistem diisolasi. Tugas-tugas asinkronous yang saling berkomunikasi diuji dengan tingkat data yang berbeda dan menentukan apakah terjadi kesalahan sinkronisasi antar tugas. Terakhir adalah pengujian sistem. Pengujian dilakukan terhadap keselururhan sistem baik perangkat keras maupun perangkat lunak. Pengujian dimaksudkan untuk menemukan kesalahan pada *interface* perangkat lunak atau perangkat keras.

D. BAGAIMANAKAH MELAKSANAKAN PENGUJIAN YANG BAIK?

Beberapa atribut yang digunakan agar pengujian dikatakan baik menurut Pressman (2010) maupun Sommerville (2011) yaitu pengujian memiliki probabilitas yang tinggi untuk menemukan kesalahan. Agar hal tersebut dapat terlaksana maka penguji harus mempunyai pemahaman bagaimana perangkat lunak dapat gagal. Disamping itu, pengujian tidak *redundant*. Pada setiap pengujian yang dilakukan harus mempunyai tujuan yang berbeda. Selanjutnya pengujian yang dikerjakan adalah jenis pengujian yang terbaik. Pengujian memungkinkan dilakukan dengan banyak cara. Pengujian yang harus digunakan adalah pengujian yang memiliki kemungkinan paling besar untuk mengungkap semua kelas kesalahan yang tinggi (dalam waktu dan usaha yang seminimal mungkin). Kemudian pengujian tidak terlalu sederhana atau komplek.

Ada beberapa aspek lagi dalam perspektif lain yang dapat dijadikan indikator bagi pelaksanaan sebuah pengujian yang baik dan optimal.

Sebagaimana disampaikan di paparan awal bahwa inti adanya pengujian adalah untuk menemukan kecacatan perangkat lunak dan mengevaluasi kualitasnya (Pressman, 2010)

(Sommerville, 2011)(Wu, 2010). Terkait dengan kualitas tentunya tidak mudah untuk memberikan justifikasi mengenai baik tidaknya kualitas sebuah produk perangkat lunak. Tingkat kualitas dari produk perangkat lunak sebenarnya tidak lepas dari bagaimana kualitas pengujiannya dikerjakan. Oleh karena kualitas adalah bukan sebuah konsep spesifik tetapi suatu ukuran yang abstrak maka pemakai hanya dapat mengetahui dan menilai bahwa kualitas hakekatnya berkaitan dengan tingkat layanan atau produk dan levelnya ditentukan dari tingkat kepuasan pelanggannya. Menilik dari hal ini maka perlu kiranya menetapkan standar ukuran kualitas. Beberapa acuan yang kemungkinan dapat digunakan untuk mengukur level kualitas pengujian perangkat lunak yaitu berupa kualitas dari kasus uji pengujiannya itu sendiri dimana pengujian perangkat lunak dapat mempunyai kecacatan juga dan kekurangan itu bisa berpengaruh buruk pada kemampuan pengujian dalam menemukan "bugs". Acuan berikutnya adalah kualitas proses pengujian yang mana stabilitasnya bergantung pada lingkungan pengujian. Selanjutnya adalah kualitas hasil uji yang mana dapat dilihat dari laporan pengujian, serta kualitas dari klien uji yaitu pembaca laporan. Mereka secara langsung dapat merasakan efek dari pengujian sehingga penilaian kualitas dapat segera dipertimbangkan.

Aspek kedua adalah ketepatan dalam memilih metode dan model pengujian. Tidak selamanya sebuah metode atau model yang menghasilkan sebuah pengujian yang baik pada sebuah perangkat lunak akan cocok pula untuk perangkat lunak yang lain. Pemilihan metode pengujian yang tepat tentunya akan turut berperan pada hasil pengujian yang optimal. Pertimbangan yang dapat digunakan dalam pemilihan metode diantaranya segi waktu, tenaga yang tersedia, serta sumber daya dan peralatan yang dimiliki.

Aspek ketiga adalah variatif dalam pelaksanaan pengujian. Pengkolaborasian beberapa teknik pengujian sudah barang tentu akan meningkatkan reliabilitas dari perangkat lunak yang diuji oleh karena telah melewati lebih dari sekali kasus uji. Kehandalan perangkat lunak bisa juga dicapai dengan pengujian perangkat lunak yang mengimplementasikan suatu metode yang telah terbukti baik kinerjanya, seperti metode Bayesian (Cheng dkk., 2010) atau transformasi matrik (Yang dkk., 2011).

Aspek keempat yaitu mendasarkan pengujian pada arsitektur perangkat lunak. Desain arsitektur memberikan gambaran bentuk tubuh dari perangkat lunak yang berisi komponen dan hubungannya [5]. Pemahaman yang baik pada arsitektur sebuah perangkat lunak akan sangat membantu dalam menentukan kasus uji dan tahapan pengujian yang tepat. Pengujian berdasarkan arsitektur juga akan membantu pendeteksian dan pencegahan kecacatan secara lebih mendalam.

Aspek kelima ialah tidak harus setiap pengujian perangkat lunak selalu menciptakan kasus uji baru dan khsusus. Terdapat kemungkinan pelaksanaan pengujian sebuah perangkat lunak cukup digandengkan (diinangkan) dengan perangkat lunak yang lain. Hal ini mungkinsaja tarjadi karena bisa saja perangkat lunak penggandeng sebenarnya telah membangkitkan secara otomatis suatu aksi-aksi tertentu yang sebenarnya hal itu dapat berperilaku sebagai kasus uji bagi perangkat yang sedang diuji. Apabila ini dapat dilaksanakan tentunya akan setidaknya

mereduksi biaya dalam mendesain kasus uji baru.

E. KESIMPULAN

Target utama dari pengujian perangkat lunak adalah menjamin kualitas produk dari perangkat lunak yang dihasilkan. Banyak parameter yang mempengaruhi untuk menghasilkan sebuah produk perangkat lunak yang berkualitas, di antaranya terkait dengan bagaimana lingkungan saat pengujian, pemilihan kasus uji dan metode, serta pendekatan yang digunakan. Aspek lain yang ikut berkontribusi dalam pengujian perangkat lunak sehingga memperoleh hasil uji yang optimal diantaranya justifikasi segi kualitas dari banyak sudut pandang, ketepatan menentukan metode dan model bentuk pengujian, variatif dalam mengkolaborasi teknik pengujian, menghiraukan bentuk arsitektur perangkat lunak, dan kemungkinan penggabungan (penginangan) pengujian pada perangkat lunak lain.

DAFTAR PUSTAKA

Cheng-Gang, B., J. Chang-Hai, and C. Kai-Yuan, 2010, A reliability improvement predictive approach to software testing with Bayesian method, in IEEE Proceeding of the 29th Chinese Control Conference, July 29-31, Beijing, China, pp. 6031-6036.

Cisar, S.M., et.al., 2012, Computer adaptive tests: a comparative study, IEEE Proceeding of 10th Jubilee International Symposium on Intelligent System and Informatics, September 20-22, Subotica, Serbia, pp. 499-504.

Clune, T.L., and R.B. Rood, 2011, Software testing and verification in climate model development, IEEE Journal, Focus: climate change software, September-October, pp. 49-55.

Jiang, F. and Y. Lu, 2012, Software testing model selection research based on yin-yang testing theory, in IEEE Proceeding of International Conference on Computer Science and Information Processing (CISP), pp. 590-594

Jin, J., and F. Xue, 2011, Rethinking software testing based on software architecture, in IEEE Proceeding of 7th International Conference on Semantics, Knowledge and Grids, pp. 148-151. DOI 10.1109/SKG.2011.32

Jin, H. and F. Zeng, 2011, Research on the definition and model of software testing quality, IEEE Journal, pp. 639-644

Kumamoto, H., et.al., 2010, Destructive testing of software systems by model checking, IEEE Journal, pp. 261-266.

Lamas, E., A.V. Dias, and A.M. da Cunha, 2013, Applying testing to enhance software product quality, in IEEE Proceeding of 10th International Conference on Information Technology: New generation, pp. 349-356. DOI 10.1109/ITNG.2013.56

Lemos, O.A.L., et. al., 2011, "Evaluation studies of software testing research in the Brazilian symposium on software engineering", in IEEE Proceeding of 25th Brazilian Symposium on

Software Engineering, pp. 56-65. DOI 10.1109/SBES.2011.30

Nakagawa, E.Y., and J.S. Maldonado, 2011, Contributions and perspectives in architectures of software testing environments, in IEEE Proceeding of 25th Brazilian Symposium on Software Engineering, pp. 66-71. DOI 10.1109/SBES.2011.42

Pressman, R.S., 2010, Software Engineering: a practitioner's approach, 7th Edition, McGraw-Hill, New York.

Reza, H., and S. Lande, 2010, Model based testing using software architecture, in IEEE Proceeding of 7th International Conference on Information Technology, pp. 188-192. DOI 10.1109/ITNG.2010.122

Sommerville, I., 2011, Software engineering, 9th Edition, Pearson Education, USA.

Wen-hong, L. and W. Xin, 2012, The software quality evaluation method based on software testing, in IEEE Proceeding of International Conference on Computer Science and Service System, pp. 1467-1471. DOI 10.1109/CSSS.2012.369

Wu, Y., Y. Zhang, and M. Lu, 2010, Software reliability accelerated testing method based on mixed testing, IEEE Journal.

Wu, X., and J. Sun, 2010, The study on an intelligent general-purpose automated software testing suite, IEEE Proceeding of International Conference on Intelligent Computation Technology and Automation, pp. 993-996. DOI 10.1109/ICICTA.2010.115

Xie, T., et.al., 2011, A study on methods of software testing based on the design models, in Proceeding of 6th International Conference on Computer Science and Education (ICCSE 2011), August 3-5, Singapore, pp. 111-113.

Yang, Y., L. Lun, and X. Chi, 2011, Research on path generation for software architecture testing matrix transform-based, IEEE Journal, pp. 2483-2486.

Zhang, B., and X. Shen, 2011, The effectiveness of real-time embedded software testing, IEEE Journal, pp. 661-664.