BAB 6
PENUTUP

Pada bab ini akan dijabarkan ringkasan penelitian dan kesimpulan-kesimpulan yang diambil berdasarkan eksperimen yang dijelaskan dari bab ketiga, yakni perencanaan pengujian, hingga bab kelima, yaitu hasil eksperimen dan analisisnya. Selain itu, bab ini juga berisi saran-saran untuk pengembanan pengujian selanjutnya.

6.1 Ringkasan Penelitian

Subbab ini menjelaskan secara singkat hasil pengujian terhadap aplikasi E-paspor mulai dari tahap perencanaan, implementasi, hingga analisis hasil implementasi. Dalam proses perencanaan pengujian didapatkan total *use case* pada aplikasi E-Paspor adalah sebanyak 23 *use case*. Dari ke-23 *use case* tersebut, hanya 11 *use case* yang merupakan *use case* utama sehingga pengujian difokuskan pada ke-11 *use case* tersebut. Skenario pengujian yang dihasilkan dengan mengkombinasikan *alternate flow* dan *basic flow* pada kesebelas *use case* berjumlah sebelas skenario karena hanya mengambil skenario *basic flow* pada masing-masing *use case.* Proses perencanaan diakhiri dengan mendapatkan *test case* pengujian. *Test case* yang dihasilkan adalah sebanyak 19 buah.

Pada bab Perencanaan telah dijelaskan bagaimana menghitung jumlah kemungkinan *path* yang dapat terjadi pada sistem E-Paspor sehingga menghasilkan angka 208 sebagai jumlah kemungkinan *path* tersebut. Demikian juga dengan *test case* yang harus dibuat yaitu minimal harus sejumlah 208 *test case* untuk menguji keseluruhan *path* dalam aplikasi E-Paspor. Sedangkan, *test case* yang dibuat untuk pengujian ini, seperti yang telah dijelaskan, hanya sebanyak 19 *test case*. Dengan demikian, prosentase pengujian baru mencapai angka 9.13%.

Pada proses implementasi, hasil yang didapatkan adalah *test script* pengujian. *Test script* ini merupakan realisasi atau implementasi dari tiap-tiap *test case*. *Test*

script yang dihasilkan dalam proses ini adalah sebanyak 43 buah termasuk *test script* yang berfungsi sebagai *test suite*. Dengan melakukan *playback* pada *test script* yang telah tersedia ini, didapatkan log hasil pengujian yang menyatakan berapa lama waktu pengujian, *script* apa saja yang dijalankan, serta status pengujian apakah berhasil (aplikasi berjalan sesuai fungsinya) atau gagal (terdapat kesalahan pada aplikasi). Pengujian yang dilakukan dengan menggunakan data sejumlah 5 *record* dari *datapool* menghasilkan pengujian yang sukses untuk alur proses permohonan paspor baru saja, alur proses permohonan paspor baru dan exit permit, serta alur proses permohonan paspor baru, exit permit, dan surat rekomendasi visa dengan durasi waktu yang berbeda-beda. Untuk alur proses permohonan paspor saja, waktu yang dibutuhkan oleh Rational Robot dalam proses *playback* adalah selama 47 menit dan 48 detik. Sedangkan durasi waktu untuk pengujian pada alur proses permohonan paspor baru dan exit permit selama 1 jam 6 menit 47 detik serta 1 jam 15 menit 3 detik untuk pengujian pada alur proses permohonan paspor baru, exit permit, dan rekomendasi visa. Pengujian dengan data yang lebih besar dapat dilihat hasilnya pada tabel 19.

6.2 Kesimpulan

Berikut ini beberapa kesimpulan yang diambil berdasarkan eksperimen yang telah dilakukan:

- Pengujian pada sistem yang sudah berjalan pengembangannya dapat dilakukan dengan terlebih dahulu mengetahui *test plan* yang kemudian dapat dijadikan *test case* dengan langkah-langkah yang telah dijelaskan. Proses ini memungkinkan terwujudnya pengujian secara menyeluruh dan efektif.
- Pengujian yang dilakukan secara otomatis dengan menggunakan Rational Robot dan Rational TestManager membutuhkan modularitas *script* agar dapat dijalankan secara independen dan memiliki sifat *reusable* atau dapat digunakan kembali.
- Konsep pengujian terotomatisasi dengan menggunakan Rational Robot dan Rational TestManager mendukung *regression test* dengan melihat

bahwa *script* yang dihasilkan pada pengujian aplikasi versi awal dapat digunakan pada pengujian aplikasi versi yang lebih tinggi untuk memastikan bahwa penambahan fitur pada versi yang lebih tinggi tidak mempengaruhi berjalannya fungsi-fungsi yang terdapat pada versi yang terdahulu.

- Pada langkah perencanaan dihasilkan *test plan* dan *test case* yang melandasi palaksanaan pengujian. *Test plan* dibuat sesuai dengan *use case* yang ada pada aplikasi. Sedangkan *test case* dibuat melalui beberapa tahap penyusunan, yaitu pembuatan skenario pengujian, pembuatan *test case* awal, dan finalisasi *test case* dengan menambahkan nilai data ke dalam *test case* pengujian.
- Langkah kedua, yaitu implementasi, dilakukan dengan membuat *test script* dengan menggunakan Rational Robot. *Test script* pengujian dapat diperkaya dengan menambahkan *verification point* dan menggunakan *datapool* sebagai penyuplai data pengujian.
- Terdapat masalah dalam penggunaan *verification point* dan *test suite* untuk pengujian. Masalah ini berkaitan dengan penggunaan *datapool* sebagai penyuplai data pengujian. Solusi untuk mengatasi masalah *verification point* dilakukan dengan membuat kode-kode tambahan dalam script sehingga dapat menghasilkan *verification point* jenis baru, yaitu yang dinamis, karena menggunakan *datapool* sebagai *baseline*. Sedangkan masalah penggunaan *test suite* diatasi dengan menggunakan *test script* tambahan dengan perintah *callscript.*
- Menganalisis hasil pengujian dilakukan melalui *test log* yang dihasilkan oleh Rational TestManager setelah Rational Robot melakukan *playback script*.

6.3 Saran

Saran yang dapat diberikan berdasarkan eksperimen yang telah dilalui adalah sebagai berikut:

- Pengujian dengan menggunakan *record datapool* yang cukup banyak (> 20) lebih baik dilakukan pada komputer dengan spesifikasi yang tinggi

(kecepatan prosesor > 1.83 GHz dan RAM > 1 GB) karena mempengaruhi kecepatan pengujian.

- Pengujian lebih baik dilakukan dari awal dimulainya proyek pengembangan sistem karena aplikasi yang sudah berjalan bahkan sudah dirilis, seperti E-Paspor, sudah memiliki fungsi-fungsi dan proses bisnis yang cukup rumit untuk dilakukan pengujian. Dengan melakukan pengujian sejak dini, *test script* yang sudah dibuat pada awal pengujian akan dapat digunakan kembali untuk pengujian berikutnya sehingga membantu mengurangi waktu yang dibutuhkan untuk pengujian. Selain itu, *defect* atau kekurangan yang terdapat dalam sistem juga dapat segera ditemukan dan ditangani.
- Penggunaan *test suite* dalam pengujian aplikasi akan sangat membantu untuk mengumpulkan *test script* dan menghasilkan *log* yang tunggal. Selain itu, *test suite* juga akan membantu dalam pembagian pengujian pada beberapa komputer sehingga pengujian menjadi lebih cepat. Pada penelitian ini penggunaan *test suite* gagal karena adanya perintah *datapool* dalam *script* sehingga tidak dapat memanfaatkan *test suite* dalam pengujian.
- Dalam menggunakan *datapool* sebagai penyedia data, sebaiknya ditetapkan dalam jumlah *record* tertentu saja. Setiap kali selesai dilakukan kegiatan pengujian, data-data yang telah dimasukkan ke dalam sistem kemudian dibersihkan *database* sistemnya sehingga data-data yang sama dari *datapool* dapat digunakan kembali tanpa menambah beban penyimpanan pada sistem yang diuji. Hal ini juga mengakibatkan tidak diperlukannya perubahan pada *script* untuk menunjuk *record* tertentu dalam *datapool* setiap kali akan dilakukan *playback*.