1. Pendahuluan

Latar Belakang

Pengujian perangkat lunak adalah proses yang penting dalam pengembangan perangkat lunak yang bertujuan untuk memastikan kesesuaian perangkat lunak dengan kebutuhannya yang mengarah pada pengembangannya [1]. Dalam prosesnya, pengujian perangkat lunak membutuhkan lebih dari separuh waktu dan biaya dari pengembangan perangkat lunak [2]. Sebelum melakukan tahap pengujian dilakukan perencanaan pengujian terlebih dahulu, yang diantaranya adalah membuat kasus uji pada pengujian perangkat lunak. Pembuatan kasus uji adalah salah satu proses yang sulit dalam tahap pengujian, maka dari itu pembuatan kasus uji memakan waktu [2]. Kasus uji dapat dihasilkan dengan secara manual yaitu menuliskan kasus uji atau secara otomatis diantaranya adalah dengan menggunakan beberapa pendekatan dalam menentukan urutan pengujian yaitu kasus uji yang dihasilkan berdasarkan kode sumber program dan kasus uji yang dihasilkan dari spesifikasi kebutuhan dan desain.

Pendekatan menggunakan spesifikasi kebutuhan dan desain banyak digunakan oleh para peneliti dalam beberapa tahun terakhir yang memiliki keuntungan yaitu memungkinkan kasus uji tersedia diawal pengembangan perangkat lunak, sehingga membuat perencanaan lebih efisien dan efektif [3]. Pendekatan menggunakan spesifikasi kebutuhan lebih dikenal dengan *blackbox testing. Black box testing* merupakan metode pengujian dimana produk diuji sesuai dengan spesifikasi atau kebutuhan perangkat lunak [4]. Implementasi pada *black box testing* salah satunya adalah *model-based testing*. *Model-based testing* adalah teknik pengujian perangkat lunak dengan System Under Test (SUT) yang diperiksa kesesuaiannya terhadap prediksi yang dibuat oleh model. *Model-based testing* memiliki beberapa kelebihan diantaranya adalah peningkatan kualitas pengujian, pengurangan biaya dan waktu pengujian [5]. Terdapat beberapa model yang digunakan salah satunya adalah Unified Modeling Language (UML) yang mencakup seperangkat teknik notasi grafis untuk membuat model visual yang dapat menggambarkan perilaku sistem yang sangat rumit dan dapat membantu dalam membuat dokumentasi yang tepat dari perangkat lunak dan memelihara konsistensi antara spesifikasi dan desain dokumen.

Unified Modeling Language diantaranya seperti *use case diagram, activity diagram, sequence diagram* dan lain-lain. Pada tugas akhir, model yang akan digunakan adalah model activity diagram karena memiliki sifat dinamis yang dimiliki oleh model. Aspek dinamis dari objek dapat dibangun, divisualisasikan, dispesifikasikan dan didokumentasikan oleh activity diagram. Activity diagram dapat memodelkan dengan memfokuskan urutan perilaku-perilaku dan kondisi-kondisi untuk mengatur sebuah kejadian yang terurut [2]. Terdapat salah satu metode yang dapat menghasilkan rangkaian pengujian yang lebih efisien dalam meningkatkan kompleksitas sesuai dengan prioritas [6] yaitu metode yang diusulkan oleh Tiwari et al yang dapat menghasilkan urutan kasus uji dengan menggunakan activity diagram yang telah dikonversi dari use case diagram berdasarkan *actor*/ pengguna.

Oleh karena itu, untuk meningkatkan kualitas pengujian, mengurangi biaya dan waktu dalam pengujian, pada tugas akhir ini akan dibangun sebuah prototipe yang mampu menghasilkan kasus uji berdasarkan *activity diagram* dengan menggunakan algoritma Depth First Search (DFS) yang akan menjamin dalam mengunjungi semua kemungkinan node dan menghapus yang berlebihan [2] dengan menguji sebuah studi kasus yaitu website digi-OTA. Hasil dari kasus uji yang didapatkan akan diuji ulang untuk memastikan kesesuaian antara kasus uji yang dihasilkan dengan hasil pengujian menggunakan metode equivalence partitioning.

Topik dan Batasannya

Berdasarkan latar belakang, didapatkan rumusan permasalahan yang ada yaitu bagaimana membangun sebuah prototipe pembangkit kasus uji menggunakan algoritma DFS dari model activity diagram dan apakah hasil kasus uji yang dihasilkan dengan algoritma DFS sesuai dengan hasil pengujian. Adapun batasan masalah pada tugas akhir ini kasus uji yang digunakan adalah website digi-OTA, bentuk file yang digunakan dalam mengubah activity diagram adalah bentuk file XMI, *tools* yang digunakan dalam pembuatan activity diagram adalah IBM Rational Software Architect, pencarian test path menggunakan algoritma DFS, hasil kasus uji dalam bentuk file csv dan notasi activity diagram yang digunakan pada program diantaranya *start, activity final, action, send signal, decision, merge dan input pin.*

Tujuan

Penelitian ini memiliki tujuan yaitu memastikan prototipe yang dibangun dengan metode yang digunakan dapat menghasilkan kasus uji menggunakan algoritma DFS dari model *activity diagram* dan menganalisis tingkat kesesuaian kasus uji yang dibangkitkan dengan hasil pengujian aplikasi Digi-OTA menggunakan selenium.

Organisasi Tulisan

Organisasi tulisan yang terdapat pada proposal ini yaitu pendahuluan, studi terkait, sistem yang dibangun, evaluasi, kesimpulan dan daftar pustaka. Pada pendahuluan merupakan penjelasan detail mengenai latar belakang, masalah dan batasannya, tujuan dan metode penelitian. Pada studi terkait merupakan penjelasanr studi literatur yang mendukung topik TA yang dikerjakan. Pada sistem yang dibangun berisi rancangan dan sistem yang dihasilkan pada tugas akhir. Untuk evaluasi merupakan hasil pengujian dan analisis pengujian. Pada kesimpulan berisi hasil akhir yang didapatkan dari hasil pengujian dan menganalisis hasil pengujian. Pada daftar pustaka berisi literatur yang membantu dalam pengerjaan.