# Analisis Perbandingan Efektivitas Sequencer TestOptimal pada Pengujian Berbasis Model Website EduALL menggunakan Extended Finite State Machine

**Rizky Khoiruddin[1], Rosa Reska Riskiana [2], Dana Sulistyo Kusumo [3]**

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung Indonesia
[1]rizkykhoiruddin@student.telkomuniversity.ac.id,
[2]rosareskaa@telkomuniversity.ac.id, [3]danakusumo@telkomuniversity.ac.id

**Abstrak**

**Website EduALL merupakan platform pendidikan berbasis web yang terus berkembang dengan berbagai fitur baru. Untuk memastikan kualitas sistem, pengujian perangkat lunak menjadi langkah penting, terutama dalam mengidentifikasi bug dan memastikan fungsionalitas tetap berjalan optimal. Selama ini, pengujian pada website EduALL masih dilakukan secara manual, yang membutuhkan waktu dan tenaga yang besar. Oleh karena itu, Model-Based Testing (MBT) berbasis Extended Finite State Machine (EFSM) digunakan sebagai metode pengujian otomatis yang lebih efisien. Penelitian ini membandingkan efektivitas dua model sequencer pada alat TestOptimal, yaitu Optimal Sequencer dan Weighted Random Sequencer, dalam menghasilkan pengujian otomatis yang efektif dan efisien. Model EFSM dibuat untuk dua versi website EduALL (versi lama dan terbaru), dan pengujian dilakukan dengan TestOptimal serta diintegrasikan dengan Selenium. Hasil penelitian menunjukkan bahwa Optimal Sequencer lebih efisien dibandingkan Weighted Random Sequencer dalam hal waktu eksekusi dan penggunaan memori, dengan tetap mencapai cakupan pengujian 100% untuk state dan transition coverage. Meskipun Weighted Random Sequencer memiliki fleksibilitas dalam pemilihan jalur uji, Optimal Sequencer lebih unggul dalam konsistensi hasil dan efisiensi pengujian. Penelitian ini menunjukkan bahwa penggunaan Model-Based Testing dengan EFSM dan TestOptimal dapat meningkatkan efektivitas pengujian otomatis pada website EduALL, khususnya dalam menangani fitur CRUD dan perubahan dinamis.**

**Kata kunci:** ***Automation Testing, Model-Based Testing, TestOptimal, Extended Finite State Machine, Model Sequencer***

**Abstract**

**The EduALL website is a web-based education platform that is constantly evolving with new features. To ensure the quality of the system, software testing is an important step, especially in identifying bugs and ensuring functionality remains optimized. So far, testing on the EduALL website is still done manually, which requires a lot of time and effort. Therefore, Model-Based Testing (MBT) based on Extended Finite State Machine (EFSM) is used as a more efficient automated testing method. This research compares the effectiveness of two sequencer models in the TestOptimal tool, namely Optimal Sequencer and Weighted Random Sequencer, in producing effective and efficient automated testing. EFSM models were created for two versions of the EduALL website (old and new), and testing was performed with TestOptimal and integrated with Selenium. The results show that the Optimal Sequencer is more efficient than the Weighted Random Sequencer in terms of execution time and memory usage, while still achieving 100% test coverage for state and transition coverage. Although the Weighted Random Sequencer has flexibility in test path selection, the Optimal Sequencer is superior in result consistency and test efficiency. This research shows that the use of Model-Based Testing with EFSM and TestOptimal can improve the effectiveness of automated testing on the EduALL website, especially in handling CRUD features and dynamic changes.**

**Keywords:** ***Automation Testing, Model-Based Testing, TestOptimal, Extended Finite State Machine, Model Sequencer***

## 1. Pendahuluan

Website EduALL merupakan platform pendidikan berbasis web yang menyediakan berbagai fitur dan program. Website ini dilengkapi Admin Dashboard dengan sistem CRUD untuk manajemen konten terpusat. Seiring dengan terus berkembangnya fitur dalam website ini, pengujian perangkat lunak menjadi langkah penting untuk memastikan fungsionalitas berjalan dengan baik [1]. *Manual Testing* sering digunakan dalam pengujian perangkat lunak, termasuk pada website EduALL, namun memerlukan banyak waktu dan tenaga [2]. Sebagai alternatif, *Automation Testing* lebih efisien untuk pengujian berulang dan regresi karena menggunakan skrip otomatis, sehingga menghemat waktu, terutama saat ada perubahan atau pembaruan [3].

Salah satu metode pengujian otomatis yang dapat diimplementasikan dalam perangkat lunak yang terus mengalami pembaruan adalah *Model-Based Testing* (MBT) [4]. MBT merupakan metode pengujian

perangkat lunak yang menggunakan model sebagai representasi dari perilaku sistem yang diharapkan [5]. MBT membantu penguji merancang rangkaian pengujian yang berkualitas dengan cakupan yang optimal, sekaligus mengurangi biaya, waktu, dan usaha dalam pengujian [6]. Penerapan MBT juga membuat pengujian lebih mudah dipelihara, ditinjau, dan diperbarui dibandingkan dengan pengujian manual [7].

Setelah membandingkan beberapa tools MBT, seperti *MOTES*, *NModel*, *Spec Explorer*, dan *TestOptimal*, dipilih *TestOptimal* karena mampu menangani sistem CRUD serta pembaruan website secara efisien. Alat ini mendukung desain dan otomatisasi pengujian yang terintegrasi dengan pendekatan MBT. *TestOptimal* mempercepat siklus pengembangan, meningkatkan cakupan pengujian, dan menangani perubahan secara efisien [8]. Model dalam *TestOptimal* berbentuk graf *Extended Finite State Machine* (EFSM), di mana simpul mewakili state dan sisi mewakili transisi [9]. EFSM juga memiliki *guard condition* yang menentukan apakah suatu transisi dapat dijalankan. Fitur ini memungkinkan EFSM mengelola skenario pengujian dengan lebih efektif dan efisien [10].

*TestOptimal* menyediakan *model sequencer* yang dapat digunakan secara gratis yaitu *Optimal*, *Random*, dan *Weighted Random Sequencer* yang digunakan untuk menghasilkan test case. Dibandingkan dengan *Random Sequencer* yang membutuhkan lebih banyak waktu, *Optimal* dan *Weighted Random* Sequencer dapat mengeksekusi model lebih cepat [11]. Oleh karena itu, penelitian ini akan membandingkan *Optimal* dan *Weighted Random sequencer* pada *TestOptimal* dengan menerapkan *Model-Based Testing* (MBT) menggunakan *Extended Finite State Machine* (EFSM) untuk menentukan *model sequencer* yang mampu menghasilkan pengujian yang lebih efektif dan efisien.

## 2. Kajian Teori

### 2.1 *Automation Testing*

*Automation Testing* adalah metode pengujian yang menggunakan skrip atau tools khusus untuk menjalankan pengujian secara otomatis dengan minim campur tangan manusia. Metode ini mengurangi pengujian manual, menekan biaya, serta memungkinkan sistem memasukkan data, membandingkan hasil, dan menghasilkan laporan secara detail [12].

### 2.2 *Model-Based Testing* (MBT)

*Model-Based Testing* (MBT) adalah metode pengujian yang memanfaatkan model sistem perangkat lunak untuk menghasilkan kasus uji (*test case*) [5]. Tujuannya adalah mengotomatiskan proses pengujian, meningkatkan cakupan dan kualitas pengujian, serta mendeteksi lebih banyak kesalahan dibandingkan metode manual. Selain itu, MBT membantu menghemat waktu, biaya, dan sumber daya dalam pengujian perangkat lunak [6]. MBT memiliki beberapa tahapan proses yang dilakukan dalam pembuatan kasus pengujian yang didefinisikan sebagai berikut [6].

1. Membuat Model Uji dari Perangkat Lunak yang diuji (SUT)
2. Menentukan Kriteria dan Fokus Pengujian
3. Membuat Kasus Pengujian
4. Menjalankan Kasus Pengujian pada Perangkat Lunak yang diuji (SUT)

### 2.3 Extended Finite State Machine (EFSM)

*Extended Finite State Machine* (EFSM) adalah pengembangan dari *Finite State Machine* (FSM) yang digunakan untuk menggambarkan perilaku sistem perangkat lunak berdasarkan spesifikasinya. FSM terdiri dari sekumpulan *state* dan *transition*, di mana setiap transisi dipicu oleh suatu peristiwa atau input [13]. Perbedaan antara FSM dengan EFSM terletak pada transisinya, dimana EFSM menggunakan syntax [*guard condition*] / *action*, *guard condition* berfungsi sebagai syarat agar transisi dapat terjadi, sedangkan *action* merupakan output yang dapat digunakan oleh model lain. Selain itu, EFSM mampu memodelkan aspek kontrol dan data dengan menambahkan variabel sebagai memori [10].

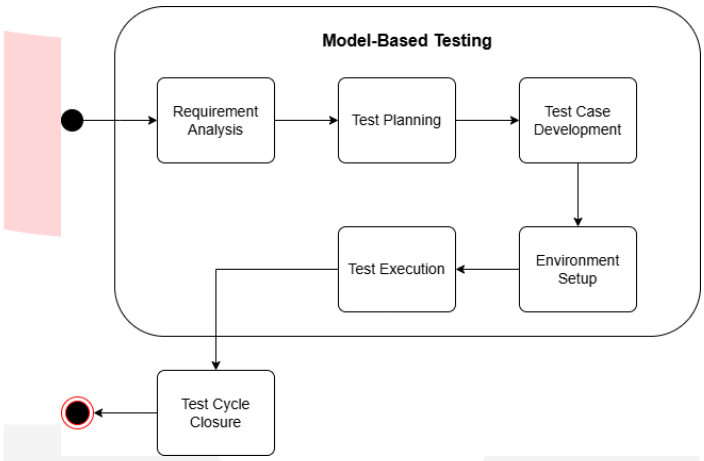
### 2.4 TestOptimal

*TestOptimal* adalah alat otomatisasi pengujian berbasis model dengan antarmuka grafis yang mendukung skrip *Java* atau *XML* [14]. Alat ini memungkinkan pembuatan model berbentuk graf EFSM untuk menguji aplikasi web. Model dapat dibuat melalui klien *HTML* interaktif atau *WebMBT Builder*, serta mendukung pembuatan sub-model yang dapat diintegrasikan. *TestOptimal* secara otomatis menghasilkan jalur terpendek untuk mencakup semua state dan transisi, memeriksa cakupan kebutuhan, memvisualisasikan kasus uji dalam bentuk grafik, dan mengeksekusinya secara otomatis [9]. *TestOptimal* membantu mempercepat siklus

pengembangan dan meningkatkan cakupan pengujian yang sebelumnya belum dilakukan [8]. *TestOptimal* menyediakan model sequencer untuk menghasilkan test case secara gratis, yaitu *Optimal*, *Random* dan *Weighted Random Sequencer*. *Optimal Sequencer* adalah algoritma untuk mencari jalur terpendek atau jalur yang singkat untuk dilintasi. *Random sequencer* menjelajahi model menggunakan variasi dari proses *markov chain* untuk membuat jalur pengujian dari state awal ke state akhir, *Weighted Random Sequencer* adalah algoritma untuk mencari jalur dengan *Weight* (W) atau bobot tertinggi untuk dilintasi terlebih dahulu dan diikuti jalur dengan bobot dibawahnya [11].

## 3. Pemodelan

Terdapat beberapa tahapan dalam penelitian ini yang ditunjukkan pada gambar 1, tahapan ini merupakan tahapan pada *Software Testing Life Cycle* (STLC) [15].




Gambar 1 Tahapan pada STLC [15]

### 3.1 Requirement Analysis

Tahap ini melibatkan identifikasi fitur pada website EduALL yang akan diuji, dengan fokus pada fitur yang mengalami pembaruan atau menyesuaikan kebutuhan terkini. Pemilihan fitur juga mempertimbangkan sistem CRUD serta fitur yang sering mengalami perubahan data. Fitur model versi lama terdapat pada tabel 1, sementara model versi terbaru ada pada tabel 2.

Tabel 1 Fitur-fitur yang dipilih pada Website EduALL versi lama

| No | Fitur | Deskripsi |
|---|---|---|
| 1 | Banner | Membuat, mengedit, merubah status dan menghapus Banner |
| 2 | Blog | Membuat, melihat, mengedit, merubah status, dan menghapus Blog |
| 3 | Mentor | Membuat, melihat, mengedit, merubah status, dan menghapus Mentor |

Tabel 2 Fitur-fitur yang dipilih pada Website EduALL versi terbaru

| No | Fitur | Deskripsi |
|---|---|---|
| 1 | Banner | Membuat, mengedit, merubah status dan menghapus Banner |
| 2 | Blog Category | Membuat, mengedit, dan menghapus Blog Category |
| 3 | Blog | Membuat, melihat, mengedit, merubah status, dan menghapus Blog |
| 4 | Mentor | Membuat, melihat, mengedit, merubah status, dan menghapus Mentor |
| 5 | Mentor Video | Membuat, mengedit, dan menghapus Mentor Video |

### 3.2 Test Planning

Pada tahap ini, pemodelan EFSM untuk website EduALL dimulai dengan membandingkan versi terbaru dan lama dari *Software Under Test* (SUT). Fitur-fitur yang akan diuji dipetakan dengan memberi label pada setiap *state* dan *transition* untuk menjaga kejelasan dan konsistensi. *State* direpresentasikan sebagai *vertex* dengan huruf 'v', sementara *transition* sebagai *edge* dengan huruf 'e'. Pemetaan salah satu fitur dapat dilihat pada Tabel 3.3, dengan detail lengkap tersedia dalam lampiran.

Tabel 3 Contoh Pemetaan pada Fitur Banner pada website EduALL

| Fitur | Label |
|---|---|
| Click Banner | e_ClickBanner |
| Banner Page | v_BannerPage |
| Click Create Banner Page | e_ClickCreateBanner |
| Create Banner Page | v_CreateBannerPage |

3.3 Test Case Development

Setelah pemetaan fitur, model dibangun menggunakan *TestOptimal*, *TestOptimal* berfungsi untuk membangun model, menghasilkan, dan mengeksekusi test case. Peneliti terlebih dahulu memahami fitur pada Admin Dashboard website EduALL yang telah dipilih dalam tahap Requirement Analysis. Model EFSM kemudian dikembangkan dengan *TestOptimal* menggunakan tampilan interaktif *drag and drop* untuk membuat *state* dan *transition*. Pengaturan lebih lanjut dilakukan dengan mengklik dua kali pada *state* atau *transition* untuk menyesuaikan variabel seperti *Data Set*, *Weight*, dan *Guard*. Model EFSM yang dibangun antara lain adalah sebagai berikut.

- Model website EduALL versi terbaru dengan Optimal Sequencer
- Model website EduALL versi lama dengan Optimal Sequencer
- Model website EduALL versi terbaru dengan Weighted Random Sequencer
- Model website EduALL versi lama dengan Weighted Random Sequencer

Perbedaan utama antara model website EduALL versi terbaru dan lama terletak pada penambahan fitur, penyesuaian input field, serta pengurangan beberapa input field sesuai pembaruan terbaru. Sementara itu, perbedaan antara model *Optimal* dan *Weighted Random* terdapat pada pemetaan transition, di mana *Weighted Random* menambahkan penamaan khusus untuk menandai bobot transition, seperti ‘e_ClickBanner_70’.

3.4 Environtment Setup

Setelah model selesai dibuat, dilakukan persiapan pengujian terhadap SUT dengan mengintegrasikan model ke dalam *TestOptimal* menggunakan *Selenium*. *Selenium* digunakan untuk menjalankan browser sesuai dengan skrip pengujian. Setelah integrasi berhasil, skrip dikembangkan untuk mengontrol *state*, *transition*, dan validasi secara akurat, memungkinkan browser menjalankan pengujian otomatis. Identifikasi elemen dilakukan menggunakan *id* dan *XPath* agar browser dapat mengenali elemen dengan tepat.

3.5 Test Execution

Pada tahap ini, dilakukan pengujian otomatis terhadap beberapa model menggunakan dua model sequencer dalam TestOptimal, yaitu Optimal Sequencer dan Weighted Random Sequencer. Optimal Sequencer mencari jalur terpendek dalam model, sementara Weighted Random Sequencer memprioritaskan jalur dengan bobot lebih tinggi. Setelah pengujian selesai, sistem menghasilkan laporan detail untuk dianalisis lebih lanjut.

3.6 Test Cycle Closure

Setelah pengujian selesai dan laporan dibuat, dilakukan analisis terhadap hasilnya. Laporan dari TestOptimal menampilkan metriks seperti cakupan pengujian, durasi, state dan transition, serta test case yang dihasilkan. Analisis ini membantu mengevaluasi efektivitas pengujian dan menentukan model sequencer yang paling efisien untuk siklus pengujian selanjutnya.

## 4. Evaluasi

4.1 Penerapan *Model-Based Testing* menggunakan EFSM

Model yang dibuat terdiri dari empat macam, yaitu: (1) model EFSM New Version dengan *Optimal Sequencer*, (2) model EFSM Old Version dengan *Optimal Sequencer*, (3) model EFSM New Version dengan *Weighted Random Sequencer*, dan (4) model EFSM Old Version dengan *Weighted Random Sequencer*. Berikut ini daftar perbedaan model New Version dan Old Version :

- Penambahan fitur Blog Category
- Penambahan fitur Mentor Video
- Penyesuaian input field Banner : (-) button title, color, dan link (+) region, dan description

- Penyesuaian input field Blog : (+) category, mentor, dan duration read

Penyesuaian pada model terbaru meningkatkan jumlah state dan transition dibanding versi sebelumnya. Pada *Weighted Random*, transition diberi bobot untuk menandakan prioritas, sementara *Optimal Sequencer* tidak menggunakan penanda bobot. Perbedaan model *Optimal* dan *Weighted Random Sequencer* terletak pada konfigurasi bobot, sehingga model EFSM dengan *Optimal Sequencer* dapat digunakan untuk membuat model *Weighted Random Sequencer* tanpa harus membuat ulang model dari awal. Setelah model dan skrip pengujian otomatis selesai dikembangkan, setiap model diuji tiga kali menggunakan *TestOptimal*. Hasil pengujian menghasilkan laporan detail untuk setiap model, yang kemudian dianalisis untuk menentukan model sequencer *TestOptimal* yang paling efektif. Berikut adalah tabel hasil pengujian untuk setiap model yang dijalankan menggunakan *TestOptimal*.

Tabel 4 Hasil Pengujian EFSM New Version dengan *Optimal Sequencer*

| Run | State | Transition | State Coverage | Transition Coverage | Test Cases | Execution Time (Minute : Second) |
|---|---|---|---|---|---|---|
| 1 | 49 | 78 | 100% | 100% | 20 | 03:10 |
| 2 | 49 | 78 | 100% | 100% | 20 | 03:08 |
| 3 | 49 | 78 | 100% | 100% | 20 | 03:12 |

Tabel 5 Hasil Pengujian EFSM Old Version dengan *Optimal Sequencer*

| Run | State | Transition | State Coverage | Transition Coverage | Test Cases | Execution Time (Minute : Second) |
|---|---|---|---|---|---|---|
| 1 | 33 | 52 | 100% | 100% | 14 | 02:03 |
| 2 | 33 | 52 | 100% | 100% | 14 | 02:01 |
| 3 | 33 | 52 | 100% | 100% | 14 | 02:03 |

Tabel 6 Hasil Pengujian EFSM New Version dengan *Weighted Random Sequencer*

| Run | State | Transition | State Coverage | Transition Coverage | Test Cases | Execution Time (Minute : Second) |
|---|---|---|---|---|---|---|
| 1 | 49 | 78 | 100% | 100% | 20 | 03:18 |
| 2 | 49 | 78 | 100% | 100% | 20 | 03:16 |
| 3 | 49 | 78 | 100% | 100% | 20 | 03:16 |

Tabel 7 Hasil Pengujian EFSM Old Version dengan *Weighted Random Sequencer*

| Run | State | Transition | State Coverage | Transition Coverage | Test Cases | Execution Time (Minute : Second) |
|---|---|---|---|---|---|---|
| 1 | 33 | 52 | 100% | 100% | 14 | 02:06 |
| 2 | 33 | 52 | 100% | 100% | 14 | 02:05 |
| 3 | 33 | 52 | 100% | 100% | 14 | 02:07 |

Berdasarkan hasil pengujian, terdapat perbedaan signifikan antara EFSM New Version dan EFSM Old Version, terutama dalam hal jumlah state dan transition yang diuji.

Tabel 8 Perbandingan model EFSM New Version dan EFSM Old Version

| Versi EFSM | State | Transition | State Coverage | Transition Coverage | Test Cases | Execution Time (Minute : Second) |
|---|---|---|---|---|---|---|
| New Version | 49 | 78 | 100% | 100% | 20 | 03:13 |
| Old Version | 33 | 52 | 100% | 100% | 14 | 02:04 |

Dalam hal kompleksitas model, model versi terbaru memiliki kompleksitas lebih tinggi dengan 16 state dan 26 transition lebih banyak dibandingkan model versi lama. Meskipun model versi terbaru lebih kompleks, kedua model berhasil mencapai 100% state dan transition coverage. Namun, Semakin kompleks model EFSM, maka semakin banyak waktu yang dibutuhkan untuk menjalankan pengujian.

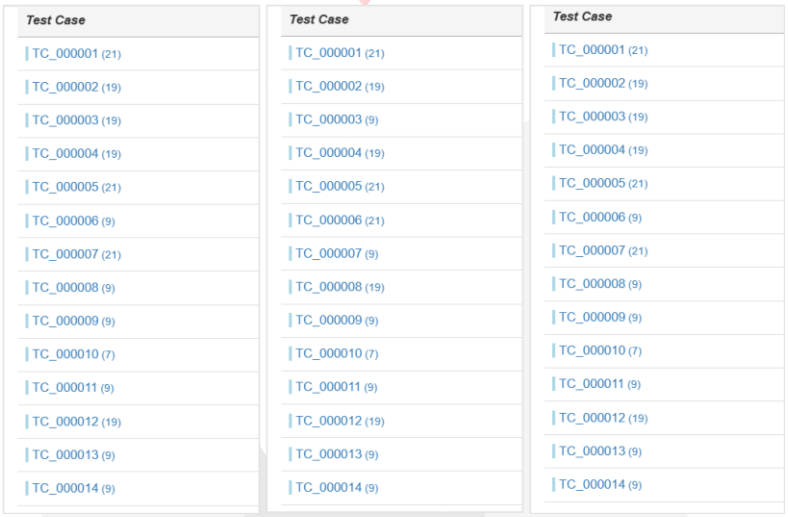
## 4.2 Perbandingan *Optimal Sequencer* dan *Weighted Random Sequencer*

Selanjutnya, analisis dilakukan terhadap hasil pengujian menggunakan dua model sequencer yaitu *Optimal Sequencer* dan *Weighted Random Sequencer*.

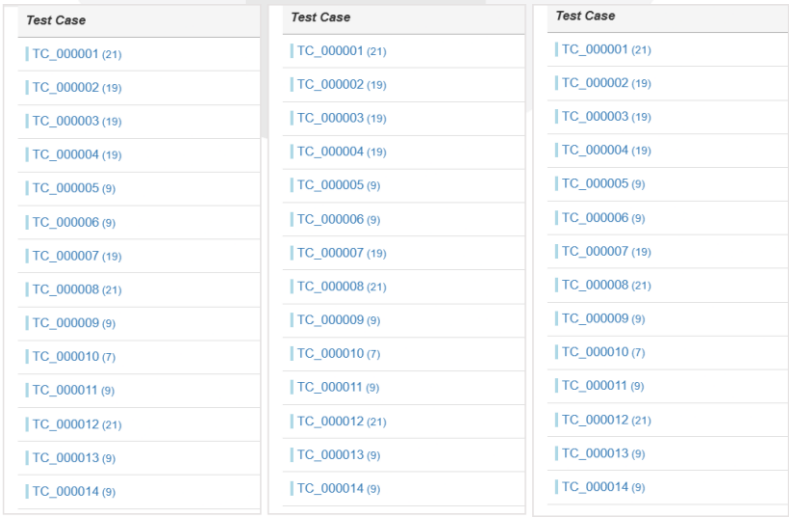
Tabel 9 Perbandingan *Optimal Sequencer* dan *Weighted Random Sequencer*

| Sequencer | Versi EFSM | Execution Time (Rata-rata) | Test Cases |
|---|---|---|---|
| Optimal Sequencer | New Version | 03:10 | 20 |
| Weighted Random Sequencer | New Version | 03:16 | 20 |
| Optimal Sequencer | Old Version | 02:02 | 14 |
| Weighted Random Sequencer | Old Version | 02:06 | 14 |

Dalam perbandingan antara *Optimal Sequencer* dan *Weighted Random Sequencer*, terlihat bahwa dalam hal efisiensi, *Optimal Sequencer* lebih cepat. Pada model versi terbaru, *Optimal Sequencer* mengeksekusi pengujian dalam waktu 03:10 menit, sementara *Weighted Random Sequencer* membutuhkan waktu 03:16 menit. Begitu juga pada Old Version, di mana *Optimal Sequencer* lebih cepat dengan waktu 02:02 menit dibandingkan *Weighted Random Sequencer* yang membutuhkan waktu 02:06 menit. Secara rata-rata, *Optimal Sequencer* lebih efisien dalam mengeksekusi pengujian, dengan perbedaan waktu sekitar 4-7 detik lebih cepat dibandingkan *Weighted Random Sequencer*. Meskipun terdapat perbedaan dalam hal kecepatan eksekusi, kedua model sequencer menghasilkan jumlah test case yang konsisten dalam setiap pengujian. Untuk model versi terbaru, baik *Optimal* dan *Weighted Random Sequencer* ini menghasilkan 20 test case, sedangkan pada model versi lama, baik *Optimal* dan *Weighted Random Sequencer* ini menghasilkan 14 test case.




Gambar 2 Hasil test case yang digenerate dengan *Optimal Sequencer*




Gambar 3 Hasil test case yang digenerate dengan *Weighted Random Sequencer*

Dalam hal konsistensi urutan test case, *Optimal Sequencer* menghasilkan urutan yang berbeda dibandingkan dengan *Weighted Random Sequencer*. Pada *Optimal Sequencer*, urutan test case pada run 1 dan 3 menunjukkan hasil yang sama, sedangkan pada run 2 terdapat perbedaan urutan. Sementara itu, *Weighted Random Sequencer* selalu menghasilkan urutan test case yang sama pada setiap eksekusi. Hal ini disebabkan oleh bobot pada setiap transisi, sehingga *TestOptimal* akan terlebih dahulu melewati transisi dengan bobot yang lebih tinggi.

Tabel 10 Perbandingan Alokasi RAM yang digunakan untuk kedua model sequencer

| Sequencer | Versi EFSM | Alokasi RAM | Penggunaan RAM (Rata-rata) |
|---|---|---|---|
| Optimal Sequencer | New Version | 4 GB | 12% (0,48 GB) |
| Weighted Random Sequencer | New Version | 4 GB | 28% (1,12 GB) |
| Optimal Sequencer | Old Version | 4 GB | 11% (0,44 GB) |
| Weighted Random Sequencer | Old Version | 4 GB | 18% (0,72 GB) |

Dalam hal penggunaan memori atau RAM, *Optimal Sequencer* lebih hemat RAM dibandingkan dengan *Weighted Random Sequencer* baik pada model New Version maupun Old Version. *Weighted Random Sequencer* mengonsumsi lebih banyak RAM karena model sequencer ini memilih urutan pengujian secara acak dengan memilih transisi dengan bobot yang lebih tinggi, sehingga memerlukan lebih banyak pemrosesan. Hal ini menunjukkan bahwa *Optimal Sequencer* lebih efisien dalam mengelola sumber daya memori atau RAM. Selain itu, semakin tinggi kompleksitas model EFSM, semakin besar juga memori yang dibutuhkan untuk menjalankan pengujian.

Berdasarkan perbandingan di atas, *Optimal Sequencer* terbukti lebih unggul dibandingkan *Weighted Random Sequencer*, karena lebih cepat dalam pengembangan dan eksekusi model, menghasilkan jumlah test case secara konsisten, serta berhasil mencapai 100% state coverage dan transition coverage, serta lebih sedikit dalam hal penggunaan memori.

## 5. Kesimpulan dan Saran

Penelitian ini berhasil mengimplementasikan Model-Based Testing (MBT) menggunakan Extended Finite State Machine (EFSM) pada pengujian otomatis website EduALL. Berikut adalah beberapa hal yang dapat disimpulkan, yaitu:

- Kompleksitas Model

  Model EFSM versi terbaru lebih kompleks dengan memiliki 16 state dan 26 transition lebih banyak dibandingkan model versi lama. Namun, semakin kompleks model EFSM, maka semakin lama waktu yang dibutuhkan untuk menjalankan pengujian.
- Perbandingan Sequencer

  Kedua sequencer menghasilkan jumlah test case yang konsisten. *Optimal Sequencer* lebih cepat dan efisien dalam mengeksekusi pengujian, dengan perbedaan waktu sekitar 4-6 detik lebih cepat dibandingkan *Weighted Random Sequencer*. *Optimal Sequencer* lebih efisien dalam mengelola sumber daya memori atau RAM. Selain itu, semakin tinggi kompleksitas model EFSM, semakin besar juga memori yang dibutuhkan untuk menjalankan pengujian.
- Efektivitas Pengujian

  Kedua sequencer mencapai 100% state dan transition coverage, menunjukkan efektivitas dalam menguji fitur website EduALL. *Optimal Sequencer* direkomendasikan untuk pengujian dengan eksekusi yang cepat, efisien dalam sumber daya memori atau RAM, serta minim konfigurasi pada modelnya.

Perusahaan dapat mempertimbangkan penggunaan *Model-Based Testing* (MBT) dengan *Extended Finite State Machine* (EFSM) dan *TestOptimal* untuk meningkatkan efisiensi pengujian otomatis, terutama pada sistem yang kompleks dan sering mengalami pembaruan. Salah satu metode yang dapat diadopsi sebagai pendekatan utama dalam pengujian otomatis adalah *Optimal Sequencer*, karena kemampuannya dalam menghasilkan test case serta mengeksekusinya secara cepat dan efisien. Dengan mengimplementasikan metode ini, perusahaan dapat mengoptimalkan proses pengujian dan mengurangi waktu yang dibutuhkan untuk memastikan kualitas sistem. Di sisi lain, perlu pengujian lebih lanjut tentang penerapan MBT pada sistem yang lebih kompleks atau platform selain web, seperti aplikasi mobile dan desktop untuk memahami sejauh mana metode ini dapat diadaptasi ke berbagai lingkungan.

**Daftar Pustaka**


[1] M. Albarka Umar and C. Zhanfang, “A Study of Automated Software Testing: Automation Tools and Frameworks,” 2019.

[2] M. Kaur and R. Kumari, “Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro,” 2011.

[3] R. Mohamad and N. M. Yassin, “Comparative Evaluation of Automated Unit Testing Tool for PHP,” *International Journal of Software Engineering and Technology*, vol. 03, no. 2, pp. 7–11, 2016.

[4] A. Petrenko, A. Simao, and J. C. Maldonado, “Model-based testing of software and systems: Recent advances and challenges,” *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 4, pp. 383–386, Aug. 2012, doi: 10.1007/s10009-012-0240-3.

[5] L. Mariani, M. Pezzè, and D. Zuddas, “Recent Advances in Automatic Black-Box Testing,” *Advances in Computers*, vol. 99, pp. 157–193, 2015, doi: 10.1016/bs.adcom.2015.04.002.

[6] L. Villalobos-Arias, C. Quesada-López, A. Martínez, and M. Jenkins, “Evaluation of a model-based testing platform for Java applications,” *IET Software*, vol. 14, no. 2, pp. 115–128, Apr. 2020, doi: 10.1049/iet-sen.2019.0036.

[7] S. Dhawan, N. Kumar, and S. Saini, “Model Based Testing Considering Steps, Levels, Tools & Standards of Software Quality,” *Journal of Global Research in Computer Science*, vol. 2, no. 4, 2011, [Online]. Available: www.jgrcs.info

[8] P. Kaur and G. Gupta, “Automated Model-Based Test Path Generation from UML Diagrams via Graph Coverage Techniques,” 2013. [Online]. Available: www.ijcsmc.com

[9] Muhammad. Shafique, *Systematic review of state based model based testing tools*. Library and Archives Canada, Bibliothèque et Archives Canada, 2011.

[10] A. S. Kalaji, R. M. Hierons, and S. Swift, “An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models,” *Inf Softw Technol*, vol. 53, no. 12, pp. 1297–1318, 2011, doi: 10.1016/j.infsof.2011.06.004.

[11] TestOptimal, “TestOptimal Model-Based Testing.” [Online]. Available: https://testoptimal.com/

[12] G. I. Safaat and V. U. Tjhin, “Analysis of Quality Assurance Performance in the Application of Manual Testing and Automation Testing for Software Product Testing,” 2024.

[13] M. M. Mariano, É. F. de Souza, A. T. Endo, and N. L. Vijaykumar, “Comparing Graph-Based Algorithms to Generate Test Cases from Finite State Machines,” *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 35, no. 6, pp. 867–885, Dec. 2019, doi: 10.1007/s10836-019-05844-6.

[14] H. Achkar, “Model Based Testing Of Web Applications What Are Web Applications,” *STANZ*, 2010.

[15] A. Arfan and Hendrik, “Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus: LMS Amikom Center PT.GIT Solution),” 2022.