Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

PENINGKATAN KUALITAS PERANGKAT LUNAK DENGAN REGRESSION TESTING BERBASIS LEAN PADA APLIKASI CLIENT-SERVER

Puguh Jayadi[1], Renny Sari Dewi [2], Sri Anardani [3]
1, 3) Teknik Informatika, Fakultas Teknik, Universitas PGRI Madiun, Indonesia
2) Bisnis Digital, Fakultas Ekonomika dan Bisnis, Universitas Negeri Surabaya, Indonesia

Article Info

Article history:

Received: 29 Januari 2025
Revised: 03 Februari 2025
Accepted: 13 Maret 2025

ABSTRACT

Abstrak

Pengujian perangkat lunak sejak dulu memberikan kontribusi signifikan terhadap kualitas perangkat lunak yang dikembangkan. Terdapat beberapa pendekatan dalam melakukan pengujian perangkat lunak seperti Regression testing. Namun di sisi lain, pengujian ini seringkali memakan waktu dan sumber daya besar, terutama pada aplikasi dengan interaksi kompleks antara client dan server. Untuk mengatasi masalah tersebut, penelitian ini mengusulkan integrasi prinsip Lean, yang berfokus pada penghapusan aktivitas non-nilai tambah (*waste*) dan optimalisasi alur kerja pengujian. Metode ini diimplementasikan secara iteratif, dimulai dari analisis kebutuhan, perancangan test case, pelaksanaan pengujian, identifikasi bug, hingga evaluasi hasil. Eksperimen dilakukan pada aplikasi impor data Excel-MySQL, dengan membandingkan waktu pengujian sebelum dan setelah penerapan Lean. Hasil penelitian menunjukkan pengurangan waktu regression testing hingga 64,28% (dari 14 jam menjadi 9 jam), dengan 94,1% test case (16 dari 17) berhasil sesuai ekspektasi. Temuan ini membuktikan bahwa Lean Development efektif dalam meningkatkan efisiensi tanpa mengorbankan cakupan pengujian. Penelitian ini memberikan kontribusi praktis bagi pengembang aplikasi client-server skala kecil, dengan menawarkan kerangka kerja yang adaptif dan hemat sumber daya.

Kata Kunci: Pengujian Perangkat Lunak, Regression Testing, Lean Development, Aplikasi Client-Server, Kualitas Perangkat Lunak

Abstract

Software testing has long been a significant contributor to the quality of the software developed. There are several approaches to conducting software testing such as regression testing. But on the other hand, this testing is often time-consuming and resource-intensive, especially in applications with complex interactions between clients and servers. To address this problem, this study proposes the integration of Lean principles, which focuses on the elimination of non-value-added activities (waste) and the optimization of testing workflows. This method is implemented iteratively, starting from needs analysis, test case design, test implementation, bug identification, to evaluation of results. The experiment was conducted on an Excel-MySQL data import application, by comparing the testing time before and after the implementation of Lean. The results showed a reduction in regression testing time of up to 64.28% (from 14 hours to 9 hours), with 94.1% of test cases (16 out of 17) succeeding as expected. These findings prove that Lean Development is effective in increasing efficiency without sacrificing test coverage. This research makes a practical contribution to small-scale client-server application developers, by offering an adaptive and resource-efficient framework.

Keywords: Software Testing, Regression Testing, Lean Development, Client-Server Application, Software Quality

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

Corresponding Author:
E-mail : puguh.jayadi@unipma.ac.id

1. PENDAHULUAN

Rekayasa perangkat lunak merupakan bidang yang terus berkembang, dengan tantangan utama dalam memastikan kualitas perangkat lunak seiring kompleksitas sistem yang semakin meningkat [1]. Salah satu aspek kritis dalam pengembangan perangkat lunak adalah regression testing (*regression testing*), yang bertujuan memastikan bahwa modifikasi kode tidak mengganggu fungsionalitas yang sudah ada [2]. Namun, regression testing seringkali memakan waktu dan sumber daya yang besar, terutama pada aplikasi client-server yang melibatkan interaksi kompleks antara komponen client dan server [3], [4]. Hal ini menjadi masalah utama yang perlu diatasi untuk meningkatkan efisiensi dan efektivitas proses pengujian.

Regression testing telah menjadi fokus penelitian dalam beberapa tahun terakhir, dengan berbagai pendekatan yang diusulkan untuk meningkatkan efisiensinya [5], [6]. Beberapa metode yang populer termasuk *test case prioritization*, *test suite reduction*, dan otomatisasi pengujian [7]. Namun, metode-metode ini seringkali tidak memperhatikan aspek pemborosan (*waste*) dalam proses pengujian, seperti pengujian berulang pada fitur yang stabil atau dokumentasi manual yang berlebihan [8]. Di sinilah prinsip Lean Development, yang berfokus pada penghapusan aktivitas non-nilai tambah, dapat memberikan solusi yang efektif [9].

Meskipun Lean Development telah berhasil diterapkan dalam berbagai bidang rekayasa perangkat lunak, penerapannya dalam regression testing masih terbatas [10]. Penelitian sebelumnya menunjukkan bahwa Lean dapat mengurangi waktu pengujian hingga 30% pada lingkungan Agile, namun metode ini belum diadaptasi secara optimal untuk aplikasi client-server skala kecil [11], [12], [13]. Selain itu, integrasi Lean dengan validasi interaksi client-server masih kurang dieksplorasi, yang seringkali menghasilkan

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

false positive atau ketidaksesuaian hasil pengujian [14]. Kesenjangan ini menjadi peluang untuk mengembangkan metode yang lebih adaptif dan efisien [15], [16].

Penelitian ini mengusulkan integrasi prinsip Lean Development ke dalam Regression Testing untuk aplikasi client-server, dengan fokus pada penghapusan aktivitas non-nilai tambah, prioritisasi test case kritis, dan validasi interaksi client-server yang akurat. Metode dirancang secara iteratif dan diimplementasikan pada aplikasi impor data Excel ke MySQL yang melibatkan proses kompleks seperti upload file, validasi data, dan sinkronisasi client-server. Tujuan penerapan Lean Development adalah untuk mengurangi waktu pengujian tanpa mengorbankan cakupan, dengan memfokuskan sumber daya pada area berisiko tinggi seperti validasi format data dan penanganan error. Implementasi konkret Lean Development mencakup tiga langkah utama: (1) identifikasi aktivitas non-nilai tambah (misalnya, pengujian fitur stabil seperti *Landing Page*), (2) prioritisasi test case untuk fitur kritis seperti Import Action dan Add/Import yang memerlukan interaksi intensif client-server, serta (3) validasi skenario *edge case* seperti import file Excel korup atau berukuran besar. Pendekatan ini memanfaatkan *spreadsheet* untuk dokumentasi test case dan hasil pengujian, memastikan transparansi dan kemudahan replikasi. Dengan demikian penelitian ini bertujuan menyeimbangkan efisiensi dan akurasi dalam regression testing untuk aplikasi berbasis data dinamis.

Kebaruan penelitian ini terletak pada integrasi Lean Development dengan pendekatan *client-server-centric* testing dan kerangka iteratif yang adaptif. Tujuan utamanya membuktikan bahwa Lean dapat dioptimalkan untuk aplikasi kecil tanpa mengorbankan akurasi deteksi cacat ataupun *bug*. Dengan demikian, penelitian ini bisa memberikan kontribusi signifikan bagi bidang rekayasa perangkat lunak, khususnya dalam konteks regression testing dan peningkatan kualitas perangkat lunak. Harapan dari penelitian ini adalah memberikan kerangka kerja yang efektif untuk meningkatkan efisiensi regression testing pada aplikasi client-server skala kecil. Nantinya metode ini dapat mengurangi waktu pengujian sekaligus mempertahankan cakupan pengujian yang komprehensif. Selain itu, penelitian ini diharapkan dapat menjadi referensi bagi pengembang dalam menerapkan prinsip Lean untuk meningkatkan kualitas perangkat lunak, terutama pada proyek dengan sumber daya terbatas.

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

2. METODE PENELITIAN

Integrasi prinsip Lean Development ke dalam regression testing untuk meningkatkan efisiensi pengujian pada aplikasi client-server menjadi tujuan utama dalam penelitian yang akan dilakukan. Metode yang digunakan bersifat iteratif dan terstruktur dalam enam tahap utama seperti yang terdapat pada Gambar 1. Tahapan tersebut tediri dari (1) analisis kebutuhan dan perancangan test case, (2) penerapan prinsip Lean, (3) pelaksanaan pengujian, (4) identifikasi cacat dan pemborosan, (5) perbaikan sistem, serta (6) evaluasi hasil. Tahapan ini dirancang untuk memastikan eliminasi aktivitas non-nilai tambah (*waste*) dan optimalisasi alur kerja pengujian [17], [18], [19].

Tahap pertama melibatkan analisis kebutuhan sistem untuk mengidentifikasi fitur-fitur kritis yang memengaruhi stabilitas dan fungsionalitas aplikasi. Berdasarkan kompleksitas dan dampaknya terhadap interaksi client-server, test case dirancang untuk mencakup skenario pengujian unit, integrasi, dan sistem [20]. Test case difokuskan pada validasi input-output, penanganan error, serta kinerja komponen yang rentan terhadap perubahan kode. Prioritas pengujian ditetapkan dengan mempertimbangkan tingkat risiko dan frekuensi modifikasi fitur [21].

Pada tahap kedua, prinsip Lean diimplementasikan melalui optimasi proses pengujian. Aktivitas redundan seperti pengujian berulang pada fitur stabil dan dokumentasi manual diidentifikasi sebagai target pengurangan [22]. Sumber daya dialokasikan secara selektif ke area berisiko tinggi, seperti komponen yang sering mengalami perubahan atau memiliki ketergantungan kompleks antar modul [23], [24]. Mekanisme prioritisasi digunakan untuk memastikan test case kritis dieksekusi terlebih dahulu, sementara test case sekunder dijalankan setelahnya [25], [26].

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

Gambar 1. Metode Penelitian

Tahap ketiga mencakup pelaksanaan regression testing secara bertahap, dimulai dari pengujian unit hingga validasi sistem lengkap. Setiap hasil pengujian didokumentasikan untuk memfasilitasi pelacakan cacat dan analisis pola pemborosan [27]. Proses ini memungkinkan identifikasi ketidaksesuaian fungsional, seperti kesalahan validasi atau respons sistem yang tidak diharapkan, yang kemudian dikategorikan berdasarkan sumber dan tingkat keparahan [28], [29].

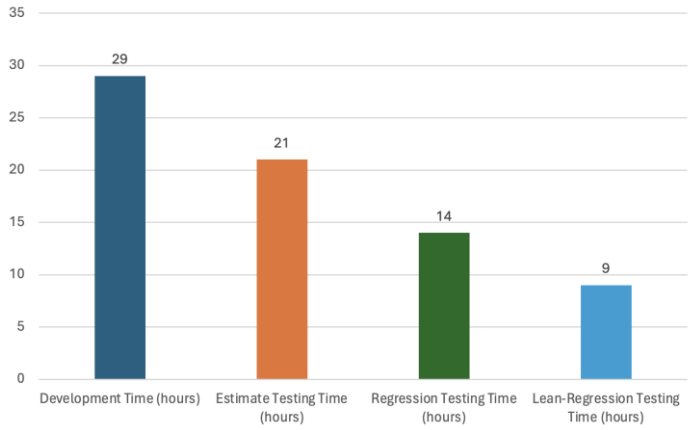
Tahap keempat hingga keenam berfokus pada perbaikan sistem dan evaluasi iteratif. Cacat yang teridentifikasi diperbaiki dengan merevisi logika kode atau meningkatkan mekanisme validasi [30]. Pemborosan seperti prosedur pengujian tidak efisien atau alur pelaporan yang rumit dieliminasi melalui penyederhanaan proses. Siklus ini diulang hingga tidak ada pemborosan signifikan yang terdeteksi, sesuai dengan prinsip *continuous improvement* dalam Lean Development. Metode ini memastikan bahwa setiap iterasi tidak hanya meningkatkan kualitas sistem, tetapi juga memperkuat efisiensi alur kerja pengujian.

3. HASIL DAN PEMBAHASAN

Dari metode penelitian yang diterapkan berbasis prinsip Lean Development, menghasilkan efisiensi waktu pengujian yang signifikan melalui pendekatan iteratif. Siklus Analisis Kebutuhan, Pelaksanaan Pengujian, dan Evaluasi Pemborosan memungkinkan identifikasi aktivitas non-nilai tambah secara sistematis. Data pada Gambar 2, menunjukkan bahwa waktu Lean-Regression Testing mencapai 9 jam, turun 42,85% dari estimasi awal (21 jam) dan 64,28% dari waktu regression testing konvensional (14 jam). Penurunan ini terutama terjadi karena eliminasi pengujian redundan pada fitur stabil, seperti pada fitur Excel Template Preparation, yang waktu pengujiannya berkurang dari 0,5 jam menjadi 0,25 jam seperti yang terdapat pada Tabel

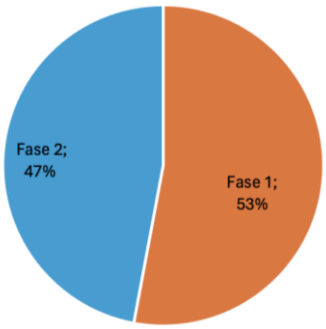
Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

1. Hasil ini membuktikan bahwa pendekatan Lean berhasil memfokuskan sumber daya pada area kritis, seperti validasi interaksi client-server, tanpa mengorbankan cakupan pengujian.



Gambar 2. Total Time

Distribusi test case pada Gambar 3 (*Test Case per Fase*) memperjelas strategi prioritisasi yang selaras dengan prinsip Lean [31], [32]. Sebanyak 53% test case (9 skenario) dialokasikan untuk pengujian sistem, seperti validasi Import action dan Datatable, sementara 47% (8 skenario) ditujukan untuk pengujian integrasi dan unit. Alokasi ini tercermin juga dalam hasil pengujian fungsional pada Tabel 2, di mana 16 dari 17 test case (94,1%) berhasil sesuai ekspektasi. Misalnya, fitur Datatable mampu menampilkan 1.000+ baris data dengan respons cepat, membuktikan efektivitas pengujian integrasi yang terstruktur. Namun, satu ketidaksesuaian ditemukan pada test case "Memastikan notifikasi error muncul setelah import template salah" akibat validasi sisi server yang tidak komprehensif. Temuan ini menegaskan pentingnya iterasi pengujian untuk menyeimbangkan fokus antara client dan server.



Gambar 3. Test Case per Fase

Analisis pada Tabel 1, mengungkap pola penghematan waktu tertinggi pada beberapa fitur. Contohnya, fitur Import action mengalami pengurangan waktu pengujian dari 4 jam (estimasi) menjadi 2 jam setelah optimasi skenario pengujian. Hal ini dicapai dengan menghapus pengujian berulang pada komponen yang stabil dan memfokuskan pada validasi alur impor data yang rentan error. Selain itu, fitur Landing Page tetap diuji

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

dengan waktu yang dipersingkat dari 3 jam menjadi 2 jam, tanpa mengorbankan navigasi fitur yang ada. Data ini menunjukkan fleksibilitas metodologi dalam mengalokasikan sumber daya sesuai tingkat risiko dan kompleksitas fitur.

Tabel 1. Estimasi Pengembangan dan Pengujian

Fitur	Fase	Waktu Pengembangan (jam)	Estimasi Pengujian (jam)	Pengujian Regression (jam)	Pengujian Lean-Regression (jam)
Import template					
Excel Template Preparation	1	1	1	0,5	0,25
Embedding template HTML	2	1	0,5	0,5	0,25
Add/Import					
HTML Build	1	2	0,5	0,5	0
File Upload	2	1	1	0,5	0,5
Import action					
Client Side	1	4	2	1	1
Server Side	2	2	2	1,5	1
Datatable					
Client Side	1	4	2	1,5	1
Server Side	2	2	2	1,5	0,5
Landing Page					
HTML Build	1	3	2	1,5	1,5
JS Build	2	2	1	1	1
Reload					
Client Side	1	1	1	0,5	0,5
Server Side	2	1	1	0,5	0,5
Empty/Reset					
Confirm Message (Client)	1	1	1	0,5	0,5
Remove All Data (server)	2	1	1	0,5	0,5
Import alert					
Embed Sweatalert	1	1	1	0,5	0
Custom Message (Server)	1	1	1	0,5	0
Message Sweatalert (Client)	2	1	1	1	0,5
	Total	**29**	**21**	**14**	**9**

Tabel 2. Pengujian Fungsional berdasarkan Lean Development

Fitur	Test Case	Haparan (Expected)	Kenyataan (Actual)	Hasil (Result)
Landing Page	Memastikan halaman utama dapat diakses dengan benar	Menampilkan Landing Page	Menampilkan Landing Page	Sesuai
	Memastikan navigasi ke fitur lain berfungsi	Navigasi bisa diakses sesuai fungsi	Navigasi bisa diakses sesuai fungsi	Sesuai
Datatable	Memastikan data yang diimpor ditampilkan dengan benar di tabel	Menampilkan data yang sudah diimport dengan jumlah baris dan kolom yang sesuai	Menampilkan data yang sudah diimport dengan jumlah baris dan kolom yang sesuai	Sesuai
	Memastikan fitur pencarian dan pengurutan data berfungsi	Dapat mengurutkan dan mencari data dalam datatable	Dapat mengurutkan dan mencari data dalam datatable	Sesuai
Empty/Reset	Memastikan ada pesan untuk mengkonfirmasi hapus semua data	Memastikan ada pesan untuk mengkonfirmasi hapus semua data	Memastikan ada pesan untuk mengkonfirmasi hapus semua data	Sesuai
	Memastikan semua data dapat dihapus dari database	Menghapus semua data yang sudah diimport	Menghapus semua data yang sudah diimport	Sesuai

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

Reload	Memastikan data terbaru dapat dimuat ulang dari data sisi server	Memastikan data terbaru dapat dimuat ulang dari data sisi server	Menampilkan data yang terbaru dari data sisi server	Sesuai
	Memastikan data terbaru dapat dimuat ulang dari data sisi client	Memastikan data terbaru dapat dimuat ulang dari data sisi client	Menampilkan data yang terbaru dari data sisi client	Sesuai
Add/Import	Memastikan bagian upload bisa ditampilkan dan diakses	Menampilkan halaman yang digunakan untuk import data Excel	Menampilkan halaman yang digunakan untuk import data Excel	Sesuai
	Memastikan bagian upload file bisa digunakan dengan drag n drop maupun select file	Dapat memilih file maupun dengan drag n drop	Dapat memilih file maupun dengan drag n drop	Sesuai
Import template	Memastikan template Excel dapat diunduh dengan format yang benar	Mengunduh template Excel untuk diimport	Mengunduh template Excel untuk diimport	Sesuai
	Memastikan template dapat diisi dan diimpor kembali	Template bisa diisi nilai	Template bisa diisi nilai	Sesuai
Import action	Memastikan file Excel dapat diupload dan diimpor ke database	File excel template bisa diupload	Melakukan impor sesuai dengan baris dan kolom dari Excel	Sesuai
	Memastikan data yang diimpor sesuai dengan file Excel	Melakukan impor sesuai dengan baris dan kolom dari Excel	Melakukan impor sesuai dengan baris dan kolom dari Excel	Sesuai
	Memastikan aplikasi dapat menangani file Excel dengan jumlah baris dan kolom yang besar	Upload file dengan total baris minimal 1000	Upload file dengan total baris minimal 1000	Sesuai
Import alert	Memastikan notifikasi sukses muncul setelah import dengan template yang benar	Manampilkan pesan hasil import sukses	Manampilkan pesan hasil import sukses	Sesuai
	Memastikan notifikasi error muncul setelah import dengan template yang salah	Manampilkan pesan hasil import error	Manampilkan pesan hasil import error	Tidak Sesuai
			Total Fitur Semua	17
			Total Fitur Sesuai	16
			Hasil Pengujian (%)	94,1

Hasil pengujian fungsional pada Tabel 2 juga mengungkap korelasi antara alokasi test case dan keandalan sistem. Fitur Import action, yang mencakup 3 test case, berhasil memvalidasi semua skenario, termasuk impor file besar (1.000+ baris). Sebaliknya, kegagalan pada Import alert (1 test case) mengindikasikan celah validasi di sisi server, yang harus segera diperbaiki. Proses iteratif ini memastikan bahwa setiap cacat teridentifikasi dan diperbaiki sebelum masuk ke fase berikutnya, sesuai prinsip *continuous improvement* dalam Lean Development.

Evaluasi pemborosan (*waste*) pada Gambar 2 dan Tabel 1 menunjukkan bahwa 64,28% aktivitas non-nilai tambah awalnya berasal dari pengujian fitur stabil dan dokumentasi manual. Setelah penerapan Lean, aktivitas tersebut berkurang menjadi 42,85%, memungkinkan realokasi sumber daya ke pengujian fitur kritis atau aktivitas pengembangan lainnya. Contohnya, waktu yang dihemat dari fitur Datatable yang semula

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

4 jam menjadi 1,5 jam bisa digunakan untuk meningkatkan validasi Server-Side pada fitur Import Action. Pendekatan ini tidak hanya mengurangi waktu, tetapi juga meningkatkan ketepatan deteksi cacat, seperti terlihat pada perbaikan logika validasi error di Import alert.

Secara holistik, kombinasi data yang dilakukan selama penelitian membuktikan bahwa metodologi Lean mampu menjawab tantangan efisiensi tanpa mengorbankan kualitas. Penghematan 6 jam pada fase Lean-Regression Testing di Gambar 2 terjadi bersamaan dengan pencapaian 94,1% kepatuhan test case pada Tabel 2, yang menunjukkan keseimbangan antara kecepatan dan akurasi. Hasil ini sangat relevan untuk aplikasi client-server seperti aplikasi impor data Excel-MySQL, di mana perubahan kode yang cepat memerlukan regression testing yang handal dan memadai. Dengan demikian, penelitian ini tidak hanya memberikan bukti empiris efektivitas Lean dalam regression testing, tetapi juga menawarkan kerangka kerja yang dapat diadaptasi untuk proyek serupa.

4. SIMPULAN

Penelitian ini berhasil mengintegrasikan prinsip Lean Development ke dalam regression testing untuk meningkatkan efisiensi dan kualitas pengujian pada aplikasi client-server. Dengan kebaruan penelitian terletak pada integrasi Lean dengan pendekatan client-server-centric testing dan kerangka iteratif yang adaptif untuk proyek kecil. Hasil eksperimen menunjukkan pengurangan waktu pengujian hingga 64,28% (dari 21 jam menjadi 9 jam) tanpa mengorbankan cakupan pengujian. Sebanyak 94,1% test case (16 dari 17) berhasil sesuai ekspektasi, dengan hanya satu ketidaksesuaian pada validasi error di sisi server yang kemudian diperbaiki melalui iterasi pengujian.

Penerapan Lean berhasil menghilangkan aktivitas non-nilai tambah (waste) seperti pengujian redundan dan dokumentasi manual, yang sebelumnya menambah waktu pengujian. Optimasi ini memungkinkan realokasi sumber daya ke area kritis, seperti validasi interaksi client-server dan fitur prioritas tinggi. Distribusi test case yang terstruktur (53% untuk pengujian sistem dan 47% untuk integrasi/unit) memastikan cakupan pengujian yang komprehensif, dengan fitur seperti Datatable dan Import action berfungsi sesuai harapan. Hasil penelitian ini memberikan kontribusi praktis bagi pengembang yang menghadapi keterbatasan sumber daya namun tetap memprioritaskan kualitas perangkat lunak. Penelitian lanjutan dapat mengembangkan

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

otomatisasi selektif dan mengeksplorasi penerapan metode ini pada sistem yang lebih kompleks, seperti aplikasi mikroservis atau IoT, untuk lebih meningkatkan efisiensi dan skalabilitas pengujian.

REFERENCES

[1] M. A. Jamil and M. K. Nour, "Managing Software Testing Technical Debt Using Evolutionary Algorithms," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 735–747, 2022, doi: https://doi.org/10.32604/cmc.2022.028386.

[2] F. Dobslaw, R. Wan, and Y. Hao, "Generic and industrial scale many-criteria regression test selection," *Journal of Systems and Software*, vol. 205, p. 111802, 2023, doi: https://doi.org/10.1016/j.jss.2023.111802.

[3] R. Sulaiman, D. A. Jawawi, and S. Halim, "Cost-effective test case generation with the hyper-heuristic for software product line testing," *Advances in Engineering Software*, vol. 175, p. 103335, 2023, doi: https://doi.org/10.1016/j.advengsoft.2022.103335.

[4] M. Kessel and C. Atkinson, "Promoting open science in test-driven software experiments," *Journal of Systems and Software*, vol. 212, p. 111971, 2024, doi: https://doi.org/10.1016/j.jss.2024.111971.

[5] F. S. Ahmed, A. Majeed, and T. A. Khan, "Value-Based Test Case Prioritization for Regression Testing Using Genetic Algorithms," *Computers, Materials and Continua*, vol. 74, no. 1, pp. 2211–2238, 2022, doi: https://doi.org/10.32604/cmc.2023.032664.

[6] K. Patel, R. M. Hierons, and D. Clark, "An information theoretic notion of software testability," *Inf Softw Technol*, vol. 143, p. 106759, 2022, doi: https://doi.org/10.1016/j.infsof.2021.106759.

[7] L. Leoni, F. De Carlo, M. M. Abaei, and A. BahooToroody, "A hierarchical Bayesian regression framework for enabling online reliability estimation and condition-based maintenance through accelerated testing," *Comput Ind*, vol. 139, p. 103645, 2022, doi: https://doi.org/10.1016/j.compind.2022.103645.

[8] W. Messner, "From black box to clear box: A hypothesis testing framework for scalar regression problems using deep artificial neural networks," *Appl Soft Comput*, vol. 146, p. 110729, 2023, doi: https://doi.org/10.1016/j.asoc.2023.110729.

[9] T. König and M. Kley, "A software framework for virtual testing of a production control system," *Procedia Comput Sci*, vol. 225, pp. 1495–1503, 2023, doi: https://doi.org/10.1016/j.procs.2023.10.138.

[10] R. Sheikh, M. I. Babar, R. Butt, A. Abdelmaboud, and T. A. Elfadil Eisa, "An Optimized Test Case Minimization Technique Using Genetic Algorithm for Regression Testing," *Computers, Materials and Continua*, vol. 74, no. 3, pp. 6789–6806, 2022, doi: https://doi.org/10.32604/cmc.2023.028625.

[11] M. K. Shin, S. Ghosh, and L. R. Vijayasarathy, "An empirical comparison of four Java-based regression test selection techniques," *Journal of Systems and Software*, vol. 186, p. 111174, 2022, doi: https://doi.org/10.1016/j.jss.2021.111174.

[12] Y. Zhang, "Application and Practice of Intelligent Algorithms in Computer Software Design and Testing," *Procedia Comput Sci*, vol. 243, pp. 708–715, 2024, doi: https://doi.org/10.1016/j.procs.2024.09.085.

[13] P. Jayadi, "Lean Development pada Efisiensi Pengembangan Aplikasi Client-Server untuk Import Data yang Dinamis," *Jurnal Manajemen Sistem Informasi (JMASIF)*, vol. 2, no. 2, pp. 47–55, 2023, doi: 10.59431/jmasif.v2i2.395.

[14] I. Ilays *et al.*, "Towards Improving the Quality of Requirement and Testing Process in Agile Software Development: An Empirical Study," *Computers, Materials and Continua*, vol. 80, no. 3, pp. 3761–3784, 2024, doi: https://doi.org/10.32604/cmc.2024.053830.

[15] S. Najihi, S. Elhadi, R. A. Abdelouahid, and A. Marzak, "Software Testing from an Agile and Traditional view," *Procedia Comput Sci*, vol. 203, pp. 775–782, 2022, doi: https://doi.org/10.1016/j.procs.2022.07.116.

[16] N. M. Minhas, J. Börstler, and K. Petersen, "Checklists to support decision-making in regression testing," *Journal of Systems and Software*, vol. 202, p. 111697, 2023, doi: https://doi.org/10.1016/j.jss.2023.111697.

[17] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, p. 111061, 2021, doi: https://doi.org/10.1016/j.jss.2021.111061.

[18] T. Mårtensson, D. Ståhl, A. Martini, and J. Bosch, "Efficient and effective exploratory testing of large-scale software systems," *Journal of Systems and Software*, vol. 174, p. 110890, 2021, doi: https://doi.org/10.1016/j.jss.2020.110890.

Djtechno : Jurnal Teknologi Informasi
Vol. 6, No. 1, April 2025
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

[19] A. M. Kazerouni, J. C. Davis, A. Basak, C. A. Shaffer, F. Servant, and S. H. Edwards, "Fast and accurate incremental feedback for students' software tests using selective mutation analysis," *Journal of Systems and Software*, vol. 175, p. 110905, 2021, doi: https://doi.org/10.1016/j.jss.2021.110905.

[20] M. Hamidi, M. Adelzadeh, and H. Hajiloo, "Improving stability in hybrid fire testing: Advancements in analysis method and software implementation," *Comput Struct*, vol. 302, p. 107473, 2024, doi: https://doi.org/10.1016/j.compstruc.2024.107473.

[21] V. Casola, A. De Benedictis, C. Mazzocca, and V. Orbinato, "Secure software development and testing: A model-based methodology," *Comput Secur*, vol. 137, p. 103639, 2024, doi: https://doi.org/10.1016/j.cose.2023.103639.

[22] M. Zakeri-Nasrabadi, S. Parsa, and S. Jafari, "Measuring and improving software testability at the design level," *Inf Softw Technol*, vol. 174, p. 107511, 2024, doi: https://doi.org/10.1016/j.infsof.2024.107511.

[23] Z. Peng, X. Lin, M. Simon, and N. Niu, "Unit and regression tests of scientific software: A study on SWMM," *J Comput Sci*, vol. 53, p. 101347, 2021, doi: https://doi.org/10.1016/j.jocs.2021.101347.

[24] A. Mustafa *et al.*, "Automated Test Case Generation from Requirements: A Systematic Literature Review," *Computers, Materials and Continua*, vol. 67, no. 2, pp. 1819–1833, 2021, doi: https://doi.org/10.32604/cmc.2021.014391.

[25] Y. Xing, X. Wang, and Q. Shen, "Test case prioritization based on Artificial Fish School Algorithm," *Comput Commun*, vol. 180, pp. 295–302, 2021, doi: https://doi.org/10.1016/j.comcom.2021.09.014.

[26] E. Passini, X. Zhou, C. Trovato, O. J. Britton, A. Bueno-Orovio, and B. Rodriguez, "The virtual assay software for human in silico drug trials to augment drug cardiac testing," *J Comput Sci*, vol. 52, p. 101202, 2021, doi: https://doi.org/10.1016/j.jocs.2020.101202.

[27] I. Ghani, W. M. N. Wan-Kadir, A. F. Arbain, and N. Ibrahim, "Improved Test Case Selection Algorithm to Reduce Time in Regression Testing," *Computers, Materials and Continua*, vol. 72, no. 1, pp. 635–650, 2022, doi: https://doi.org/10.32604/cmc.2022.025027.

[28] V. Garousi, A. B. Keleş, Y. Balaman, Z. Ö. Güler, and A. Arcuri, "Model-based testing in practice: An experience report from the web applications domain," *Journal of Systems and Software*, vol. 180, p. 111032, 2021, doi: https://doi.org/10.1016/j.jss.2021.111032.

[29] W. D. van Driel, J. W. Bikker, and M. Tijink, "Prediction of software reliability," *Microelectronics Reliability*, vol. 119, p. 114074, 2021, doi: https://doi.org/10.1016/j.microrel.2021.114074.

[30] R. Caballero, E. Martin-Martin, A. Riesco, and S. Tamarit, "A unified framework for declarative debugging and testing," *Inf Softw Technol*, vol. 129, p. 106427, 2021, doi: https://doi.org/10.1016/j.infsof.2020.106427.

[31] H. Pei, B. Yin, M. Xie, and K.-Y. Cai, "Dynamic random testing with test case clustering and distance-based parameter adjustment," *Inf Softw Technol*, vol. 131, p. 106470, 2021, doi: https://doi.org/10.1016/j.infsof.2020.106470.

[32] T. M. Kanner, A. M. Kanner, and A. V Epishkina, "Algorithm for Optimal and Complete Testing of Software and Hardware Data Security Tools," *Procedia Comput Sci*, vol. 190, pp. 408–413, 2021, doi: https://doi.org/10.1016/j.procs.2021.06.049.