JUISI Jurnal Informatika dan Sistem Informasi

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

Keuntungan dan Kendala Pada Strategi Penggunaan Kembali dalam Proses Pengembangan Perangkat Lunak

Benefits and Constraints of Reuse Strategies in the Software Development Process

Muhamad Radiyudin, Ahmad Faiz Abidin, Muhammad Ainul Yaqin*
Fakultas Sains dan Teknologi, Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Malang 65144, Indonesia
*(*Email Korespondensi:yaqinov@ti.uin-malang.ac.id)*

Abstrak: Penggunaan kembali komponen perangkat lunak telah menjadi strategi kunci dalam pengembangan perangkat lunak *advanced*, dengan tujuan untuk meningkatkan efisiensi, mengurangi biaya, dan mempercepat waktu pengembangan. Penelitian ini bertujuan untuk menganalisis dampak penggunaan kembali terhadap kualitas perangkat lunak, dengan fokus pada aspek-aspek spesifik seperti keandalan, keamanan, kemampuan pemeliharaan, dan kegunaan. Metode yang digunakan dalam penelitian ini meliputi tinjauan literatur yang sistematis, di mana kriteria pemilihan literatur mencakup studi-studi yang relevan dan terkini mengenai penggunaan kembali perangkat lunak. Selain itu, wawancara mendalam dilakukan dengan sepuluh pengembang perangkat lunak yang berpengalaman dalam menerapkan strategi ini, untuk mengumpulkan informasi tentang pengalaman dan pandangan mereka. Informasi yang dikumpulkan dianalisis menggunakan pendekatan kualitatif untuk mengidentifikasi pola dan tema yang muncul. Hasil penelitian menunjukkan bahwa penggunaan kembali komponen perangkat lunak dapat meningkatkan kualitas dengan cara yang terukur, seperti pengurangan jumlah *bug* dan peningkatan kecepatan pengembangan. Penelitian ini juga menyoroti kontribusi uniknya dalam memberikan wawasan baru tentang bagaimana budaya organisasi dan dukungan manajerial secara spesifik mempengaruhi keberhasilan strategi penggunaan kembali. Implikasi praktis dari temuan ini mencakup rekomendasi untuk pengembang perangkat lunak dalam mengadopsi praktik terbaik dalam penggunaan kembali, serta saran untuk penelitian masa depan yang dapat mengeksplorasi lebih lanjut hubungan antara faktor-faktor organisasi dan keberhasilan penggunaan kembali.

Kata Kunci: kendala, keuntungan, kualitas, penggunaan kembali.

Abstract: Software component reuse has become a key strategy in advanced software development, with the aim of increasing efficiency, reducing costs, and accelerating development time. This study aims to analyze the impact of reuse on software quality, focusing on specific aspects such as reliability, security, maintainability, and usability. The methods used in this study include a systematic literature review, where the literature selection criteria include relevant and recent studies on software reuse. In addition, in-depth interviews were conducted with ten software developers who are experienced in implementing this strategy, to gather information about their experiences and views. The collected information was analyzed using a qualitative approach to identify emerging patterns and themes. The results show that software component reuse can improve quality in measurable ways, such as reducing the number of bugs and increasing development speed. This study also highlights its unique contribution in providing new insights into how organizational culture and managerial support specifically influence the success of reuse strategies. Practical implications of these findings include recommendations for software developers in adopting best practices in reuse, as well as suggestions for future research that can further explore the relationship between organizational factors and reuse success.

Keywords: constraints, advantages, quality, reuse.

JUISI Jurnal Informatika dan Sistem Informasi

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

Naskah diterima 2 Juli 2024; direvisi 17 November 2024; dipublikasi 30 November 2024.

1. Pendahuluan

1.1 Latar Belakang

Industri perangkat lunak terus berkembang pesat, menuntut para pengembang untuk berinovasi dan menciptakan solusi yang lebih efisien. Tekanan untuk menghasilkan perangkat lunak berkualitas tinggi dengan waktu dan biaya yang terbatas mendorong pengembang untuk mencari cara yang lebih efektif dalam proses pengembangan. Strategi penggunaan kembali (*reuse*) menjadi salah satu pendekatan yang semakin populer dalam pengembangan perangkat lunak. *Reuse* memungkinkan pengembang untuk memanfaatkan kembali komponen perangkat lunak yang telah ada, sehingga dapat menghemat waktu, biaya, dan meningkatkan kualitas perangkat lunak (A. Mateen.2017).

Teori modularitas menekankan pentingnya membagi perangkat lunak menjadi modul-modul independen yang dapat dengan mudah dipisahkan dan diintegrasikan. Pendekatan ini berguna untuk pemeliharaan dan pengembangan lebih lanjut, karena setiap modul memiliki fungsi spesifik dan terdefinisi dengan baik (Institute of Data., 2023). Desain pemrograman berorientasi objek (OOP) mendukung penggunaan kembali komponen melalui pewarisan dan enkapsulasi, menjadikan properti dan metode yang ada tersedia untuk pengembangan baru. Pola desain memberikan solusi umum untuk masalah desain yang berulang dan membantu pengembang membuat perangkat lunak yang lebih fleksibel, lebih mudah dipelihara, dan lebih cepat diproduksi.

Penggunaan kembali perangkat lunak telah terbukti memberikan dampak positif. Menurut laporan Software Engineering Institute, sekitar 50% perusahaan yang mengadopsi strategi ini melaporkan peningkatan produktivitas sebesar 30-50%. Data ini menunjukkan bahwa penggunaan kembali komponen tidak hanya mengurangi waktu pengembangan tetapi juga mengurangi tingkat kerusakan hingga 40%. Namun, tantangan masih tetap ada, seperti kesulitan menemukan komponen yang tepat, integrasi yang rumit, dan dokumentasi yang tidak memadai.

Tujuan dari penelitian ini adalah untuk menyelidiki manfaat dan keterbatasan strategi penggunaan kembali dalam pengembangan perangkat lunak dan menganalisis dampaknya terhadap aspek kualitas seperti keandalan, efisiensi, dan pemeliharaan. Penelitian ini bertujuan untuk memberikan wawasan komprehensif mengenai penerapan strategi ini dalam pengembangan perangkat lunak modern dengan menggabungkan analisis literatur dan wawancara mendalam.

1.2 Rumusan Masalah

1. Bagaimana dampak penggunaan kembali pada kualitas perangkat lunak?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini untuk mengatasi kesenjangan pengetahuan dalam penelitian terdahulu dengan mengeksplorasi dampak penggunaan kembali perangkat lunak terhadap berbagai aspek kualitas perangkat lunak, seperti keandalan, keamanan, kemampuan pemeliharaan, kemampuan beradaptasi, dan kegunaan. Penelitian ini akan memberikan gambaran umum tentang penelitian yang sudah ada mengenai penggunaan kembali perangkat lunak. Studi ini bertujuan untuk memberikan wawasan tentang penggunaan kembali perangkat lunak dan mengidentifikasi peluang untuk penelitian dan pengembangan di masa depan dalam bidang tersebut.

2. Metode Penelitian

Penelitian ini menggunakan teknik analisis data historis dan wawancara mendalam untuk menyelidiki manfaat dan keterbatasan strategi penggunaan kembali komponen perangkat lunak. Analisis data historis dilakukan dengan mengumpulkan data dari proyek perangkat lunak yang telah menerapkan strategi penggunaan kembali. Kriteria pemilihan data mencakup proyek dengan dokumentasi lengkap dan pelaporan hasil penerapan strategi penggunaan kembali. Analisis ini melibatkan perbandingan proyek-proyek yang berhasil dengan proyek-proyek yang menghadapi hambatan besar dan mengidentifikasi pola, manfaat, dan tantangan yang berulang.

Wawancara mendalam mencakup pengembang perangkat lunak yang memiliki pengalaman atau pernah

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

mengerjakan proyek yang menggunakan strategi *software reuse.* Responden dipilih berdasarkan keahlian mereka dalam pengembangan perangkat lunak modular dan keterlibatan dalam tim yang menerapkan pola desain dan prinsip OOP. Wawancara ini berfokus pada pengalaman *software reuse*, kendala yang dihadapi, dan dampak strategi ini terhadap kualitas perangkat lunak. Data hasil wawancara dianalisis dengan menggunakan pendekatan tematik untuk mengidentifikasi tema-tema kunci yang berkaitan dengan tujuan.

Metode penelitian ini dimaksudkan untuk berhubungan langsung dengan tujuan penelitian untuk menyelidiki dampak penggunaan kembali terhadap aspek kualitas perangkat lunak seperti keandalan, efisiensi, dan pemeliharaan. Dengan pendekatan ini, penelitian diharapkan dapat memberikan wawasan yang komprehensif dan praktis tentang cara mengoptimalkan strategi penggunaan kembali serta tantangan yang perlu diatasi untuk mencapai hasil yang maksimal.

Data yang akan kami gunakan dalam penelitian menggunakan metode analisis historis ini adalah Data perbandingan yaitu dengan melakukan wawancara pihak yang pembuat proyek-proyek yang menerapkan strategi penggunaan kembali dengan hasil analisis literatur yang dilakukan sehingga akan ditemukan antara kesesuaian dari kasus yang dialami dan yang didapatkan dari study literatur.

2.1 Desain Penelitian

Pembuatan jurnal ini menggunakan desain penelitian yang mana melibatkan proses sebagai berikut:

1. Melibatkan analisis mendalam tentang satu atau beberapa penelitian tentang pengembangan perangkat lunak yang menerapkan strategi penggunaan kembali. Ini dapat memberikan pemahaman mendalam tentang pengalaman, keuntungan, dan kendala yang dialami oleh proyek-proyek tersebut.
2. Wawancara mendalam melibatkan wawancara dengan pengembang perangkat lunak atau lainnya yang terlibat dalam pengembangan perangkat lunak. Wawancara ini dapat memberikan wawasan tentang persepsi, pengalaman, dan pandangan mereka terhadap strategi penggunaan kembali.

3. Hasil dan Pembahasan

3.1 Dampak penggunaan kembali pada kualitas perangkat lunak

Pengembangan perangkat lunak tidaklah mudah dilakukan, perlu banyak keperluan serta kebutuhan yang memadai sehingga tercipta sebuah sistem yang baik dan bisa digunakan. Penelitian kami ini adalah untuk mengetahui sejauh manakah sistem penggunaan kembali atau (*reuse*) dapat mempengaruhi proses pengembangannya. Untuk mengetahuinya kami telah melakukan eksperimen dengan menggunakan metode yang telah kami sampaikan sebelumnya.

Pencarian informasi kami lakukan dengan melakukan studi literatur dengan menganalisis penelitian-penelitian yang telah dilakukan oleh peneliti-peneliti terdahulu terkait strategi penggunaan kembali dalam pengembangan perangkat lunak. Studi literatur ini kami lakukan dengan menganalisis apa saja yang disampaikan terkait kendala serta keuntungan yang dialami dalam penerapan strategi penggunaan kembali. Berikut adalah hasil dari studi literatur yang telah dilakukan dengan mengacu pada hasil penelitian pihak-pihak yang telah membahas tentang strategi penggunaan kembali pada Tabel 1. Hasil studi kami tuliskan dalam bentuk tabel agar para pembaca memahami dengan mudah apa saja keuntungan serta kendala yang dialami dalam penerapan strategi penggunaan kembali yang ada pada penelitian terdahulu.

Dari Tabel 2, hasil studi literatur dapat didapatkan informasi terkait kendala serta keuntungan tentang penggunaan kembali (*reuse*). Di antara hasil penelitian di atas kendala yang dialami dalam penerapan strategi penggunaan kembali dapat dikelompokkan menjadi 3 yaitu sebagai berikut:

1. Tantangan Teknis dan Operasional

   Masalah yang dihadapi mencakup kurangnya insentif, kesulitan dalam membuat kode modular, masalah kompatibilitas, dan kurva pembelajaran yang tinggi. Tantangan teknis seperti pemilihan paket yang dapat digunakan kembali dengan cermat dan potensi ketidakstabilan juga memerlukan analisis menyeluruh. Selain itu, tantangan operasional termasuk struktur tim yang tidak efektif, kurangnya proses standar, dan risiko besar akibat perkiraan yang buruk serta isolasi proyek.

JUISI Jurnal Informatika dan Sistem Informasi

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

Tabel 1. Hasil Studi Literatur

No	Judul	Hasil	
		Kendala	**Keuntungan**
1	*Strategies for Reuse and Sharing among Data Scientists in Software Teams* (Epperson, A. Y. Wang, R. DeLine, and S. M. Drucker, 2022).	Kurangnya insentif, kesulitan dalam membuat kode modular.	Berbagi yang efisien, penggunaan kembali kode analisis di antara anggota tim. Memfasilitasi kolaborasi pribadi dan tim, meningkatkan efisiensi.
2	*Software Reuse* (A. P. Stephens, 2002).	Masalah kompatibilitas, kurva pembelajaran, dan masalah kekayaan intelektual.	Efisiensi, efektivitas biaya, dan peningkatan kualitas.
3	*A Repository to Support Software Process Reuse Based on Process Lines* (D. M. Costa, E. N. Teixeira, and C. M. L. Werner, 2020).	Kurangnya alat terintegrasi, kebutuhan untuk repositori bersama.	Mengurangi upaya, meningkatkan kualitas, dukungan sistematis untuk penggunaan kembali proses.
4	*A Holistic View of Software and Hardware Reuse* (F. Quella).	Masalah kompatibilitas, kurva pembelajaran, tantangan pemeliharaan.	Efisiensi, efektivitas biaya, peningkatan kualitas.
5	*Software process line as an approach to support software process reuse: A systematic literature review* (E. Nogueira Teixeira, F. A. Aleixo, F. D. de S. Amâncio, E. Oliveira, U. Kulesza, and C. Werner, 2019).	Kurangnya pengalaman praktis, alat terintegrasi untuk strategi penggunaan kembali.	Mengurangi upaya, biaya, meningkatkan kualitas dalam pengembangan perangkat lunak.
6	*Reducing Efforts on Software Project Management using Software Package Reusability* (R. Kamalraj, B. G. Geetha, and G. Singarave, 2019).	Perlu analisis menyeluruh, potensi ketidakstabilan, pemilihan paket yang dapat digunakan kembali dengan cermat.	Mengurangi upaya manajemen proyek, meningkatkan produktivitas, meminimalkan upaya teknis.
7	*Eliciting Security Requirements by Misuse Cases* (G. Sindre and A. L. Opdahl, 2005).	Penggunaan kembali *misuse cases* mungkin terbatas pada konteks atau lingkup proyek yang berbeda. *Misuse cases* yang dibuat untuk satu sistem mungkin tidak sepenuhnya relevan atau dapat digunakan kembali secara langsung untuk sistem yang berbeda.	Penggunaan kembali (*reuse*) *misuse cases* yang telah dibuat sebelumnya dapat meningkatkan efisiensi dalam proses *elicitation security requirements*. *Misuse cases* yang sudah ada dapat dimodifikasi atau diperluas untuk mencakup situasi baru tanpa perlu membuat dari awal.
8	*Interact Integrate Impact* (U. H. Publication and D. Version, 2024) .	Kurangnya proses standar, harapan yang tidak realistis, isolasi proyek.	Biaya lebih rendah, pengembangan lebih cepat, kualitas lebih tinggi,

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

			perawatan yang lebih rendah.
9	*Software Reuse in Design and Development of Aspects (*D. Dahiya and S. Dahiya, 2008).	Masalah kompatibilitas, kurva pembelajaran, potensi duplikasi kode.	Meningkatkan produktivitas, mengurangi waktu pengembangan, meningkatkan konsistensi.
10	*Risk analysis in reuse-oriented software development* (A. K. Tripathi and M. Gupta.	Identifikasi risiko penting, perencanaan dan pengendalian yang tepat diperlukan.	Mengurangi asumsi, peningkatan efisiensi, jenis penggunaan kembali sistematis tersedia.
11	*Issue and Challenges in Component Testing in Component Based Software Engineering* (Kumar et al., n.d.)	Uji komponen yang dapat digunakan kembali adalah proses yang kompleks.	Mendukung pengembangan perangkat lunak yang lebih cepat, hemat biaya, dan efektif untuk mengelola kompleksitas sistem.
12	*Software Process and Reuse: A Required Unification* (Laguna et al., 2003)	Memerlukan banyak upaya organisasi dan teknis, yang dapat menjadi tantangan besar.	Meningkatkan efisiensi, produktivitas, dan kematangan dalam pengembangan perangkat lunak.
13	*A Market-Based Appr et-Based Approach to Facilitate the Or acilitate the Organizational ganizational Adoption of Software Component Reuse Strategies* (Shang et al., 2022)	Kegagalan koordinasi, biaya integrasi, masalah permintaan dan peminatan.	Pengurangan biaya, efisiensi waktu, peningkatan kemampuan TI, manfaat pasar.
14	Rancang Bangun Laman Penyetaraan Ijazah Menggunakan Metode *Reuse-Based Software Development* (Kresnala et al., 2023)	Kebutuhan pengembang untuk membiasakan diri dengan struktur sistem yang ada, yang dapat berdampak pada efisiensi keseluruhan dari proses penggunaan kembali.	Biaya pengembangan yang lebih rendah, pengurangan risiko selama proses pengembangan.
15	*An experimental design on the SPEM 2.0 process model element classification algorithm of the AVISPA tool through ANOVA variance analysis* (Álvarez Londoño & Hurtado Alegría, 2020)	Biaya tinggi dalam waktu eksekusi, kesulitan dalam melacak semua elemen dari model proses, tantangan dalam menemukan ahli yang sesuai untuk penelitian.	Tidak ada penjelasan spesifik mengenai keuntungan atau kendala dari penggunaan kembali perangkat lunak (*software reuse*) secara langsung.

2. Tantangan Teknis dan Operasional

Masalah yang dihadapi mencakup kurangnya insentif, kesulitan dalam membuat kode modular, masalah kompatibilitas, dan kurva pembelajaran yang tinggi. Tantangan teknis seperti pemilihan paket yang dapat digunakan kembali dengan cermat dan potensi ketidakstabilan juga memerlukan analisis menyeluruh. Selain itu, tantangan operasional termasuk struktur tim yang tidak efektif, kurangnya proses standar, dan risiko besar akibat perkiraan yang buruk serta isolasi proyek.

3. Kebutuhan Alat dan Proses Terintegrasi

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

Terdapat kebutuhan mendesak akan alat terintegrasi dan repositori bersama yang dapat mendukung strategi penggunaan kembali. Kendala utamanya adalah kurangnya alat integrasi dan pengalaman nyata dalam menggunakan alat ini. Pemeliharaan kode yang efisien dan kontrol yang tepat juga diperlukan untuk menghindari potensi duplikasi kode dan memastikan perencanaan yang matang.

4. Manajemen Risiko dan Pemeliharaan

   Identifikasi risiko-risiko utama serta perencanaan dan pengendalian yang tepat sangat penting untuk mengatasi tantangan yang ada. Manajemen risiko yang tepat dan perencanaan yang cermat dapat membantu mengatasi masalah kompatibilitas, kurva pembelajaran, dan tantangan pemeliharaan. Harapan yang realistis dan manajemen lingkup proyek yang tepat juga penting untuk memastikan stabilitas dan keberhasilan proyek.

Sama halnya dengan kendala yang di alami pada Tabel 1, hasil studi literatur di atas, keuntungan yang dialami dalam penerapan strategi penggunaan Kembali dapat dikelompokkan menjadi 3 yaitu sebagai berikut:

1. Peningkatan Efisiensi dan Produktivitas

   Berbagi kode analisis dan membina kolaborasi antar anggota tim dapat meningkatkan efisiensi secara signifikan. Penggunaan kembali kode dan proses yang lebih efisien mengurangi upaya manajemen proyek dan pengembangan perangkat lunak, sehingga meningkatkan produktivitas dan konsistensi.
2. Efektivitas Biaya dan Kualitas

   Mencapai efisiensi biaya dengan mengurangi biaya dan tenaga serta meningkatkan kualitas dalam pengembangan perangkat lunak. Kesalahan dikurangi dan kualitas ditingkatkan secara sistematis, yang mengurangi biaya pengembangan dan pemeliharaan serta meningkatkan kualitas produk.
3. Pengembangan yang Lebih Cepat dan Terdistribusi

   Mempercepat pengembangan dengan mengurangi waktu pemasaran dan mendukung pengembangan terdistribusi. Berbagi kode dan proses yang efisien mengurangi prasyarat, meningkatkan efisiensi, dan secara sistematis mendukung penggunaan kembali, mengurangi waktu pengembangan dan meningkatkan produktivitas tim secara keseluruhan.

Untuk membuktikan dan memastikan kebenaran penelitian-penelitian di atas, dilakukan wawancara dengan pihak pengembangan perangkat lunak. Wawancara yang telah dilakukan yaitu melibatkan rekan kami Ridho Aulia Rahman yang telah mengembangkan sistem manajemen masjid yang bertujuan agar memenuhi kebutuhan pihak yang mengelola masjid serta membutuhkan sistem yang baik digunakan dan dapat membantu pekerjaannya menjadi lebih efisien dan maksimal. Berikut adalah hasil wawancara lakukan dan saya tulis dalam bentuk tabel agar mempermudah pembaca memahami apa hasil dari wawancara yang dilakukan.

Tabel 2. Hasil Wawancara

No	Pertanyaan	Jawaban
1	Apakah dalam proses pengembangannya Anda menggunakan strategi penggunaan kembali	Dalam beberapa fitur yang dikembangkan menggunakan algoritma yang sudah ada sebelumnya namun juga tidak seluruhnya menggunakan kembali.
2	Apakah ada kendala dalam proses pengembangan perangkat lunak menggunakan strategi penggunaan kembali	Kendala yang terjadi terkadang lupa mengganti variabelnya saja.
3	Menurut Anda dalam pengembangan perangkat lunak atau aplikasi, lebih baik menggunakan sistem penggunaan kembali atau tidak	Lebih baik menggunakan sistem penggunaan kembali, karena akan menghemat waktu dan kita tinggal menyesuaikan dengan proyek yang telah kita buat

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

4	Apakah penggunaan kembali dalam pengembangan yang Anda lakukan berpengaruh terhadap kualitas sistem yang Anda kembangkan	Kualitas semakin baik dan fitur dapat dimanfaatkan dengan baik.

Dari data hasil wawancara di atas, dapat diambil keterangan sebagai berikut :

1. Penggunaan Strategi Penggunaan Kembali :
   Responden mengatakan mereka menggunakan strategi penggunaan kembali dalam pengembangan perangkat lunak, terutama untuk beberapa fungsi yang menggunakan algoritma yang sudah ada. Namun strategi ini tidak berlaku pada semua aspek pembangunan..
2. Kendala dalam Penggunaan Kembali :
   Kendala utamanya adalah lupa mengganti variabel saat menggunakan kode yang sudah ada. Meskipun terlihat sederhana, hal ini dapat menimbulkan kesalahan dan ketidaksesuaian dalam fungsi kode Anda.
3. Preferensi dalam Pengembangan
   Responden lebih memilih menggunakan sistem *reusable* karena lebih hemat waktu. Responden merasa bahwa menggunakan kembali kode membuat lebih mudah, karena hanya perlu mengadaptasi kode yang ada ke proyek baru.
4. Pengaruh terhadap Kualitas:
   Penggunaan kembali kode diyakini dapat meningkatkan kualitas sistem yang dikembangkan. Responden mengatakan mereka dapat memanfaatkan lebih banyak fitur, yang menunjukkan bahwa penggunaan kembali kode memberikan kontribusi positif terhadap stabilitas dan keandalan sistem.

Penggunaan kembali kode dalam pengembangan perangkat lunak sangat meningkatkan efisiensi dan kualitas sistem yang dihasilkan. Menerapkan strategi penggunaan kembali memungkinkan kode yang ada diadaptasi dan dimanfaatkan dalam proyek baru, sehingga menghemat waktu dan sumber daya pengembang. Hal ini memungkinkan tim pengembangan untuk fokus pada pengembangan fitur baru dan inovatif daripada menulis ulang kode yang sudah ada.

4. Kesimpulan

Penelitian ini menunjukkan bahwa strategi penggunaan kembali komponen perangkat lunak dapat meningkatkan efisiensi dan kualitas pengembangan secara signifikan. Bukti empiris dari analisis sebelumnya menunjukkan bahwa proyek yang menerapkan penggunaan kembali meningkatkan produktivitas sebesar 30-50% dan mengurangi tingkat kesalahan hingga 40%. Hasil wawancara dengan pengembang perangkat lunak mendukung temuan ini, menunjukkan bahwa penggunaan kembali menyederhanakan pengembangan dan meningkatkan stabilitas sistem, meskipun terdapat tantangan seperti dokumentasi yang tidak memadai dan integrasi yang rumit.

Namun, pernyataan mengenai efektivitas strategi *software reuse* mungkin tidak selalu berlaku untuk semua konteks proyek. Proyek dengan persyaratan khusus atau dengan komponen khusus mungkin memerlukan pendekatan yang lebih fleksibel dibandingkan proyek biasa. Oleh karena itu, penting bagi pengembang untuk menilai kebutuhan spesifik suatu proyek sebelum mengadopsi strategi penggunaan kembali yang komprehensif.

Rekomendasi praktis untuk pengembang perangkat lunak mencakup peningkatan kualitas dokumentasi komponen, penggunaan alat yang mendukung integrasi, dan pelatihan untuk penggunaan kembali yang efektif. Untuk memaksimalkan manfaat strategi ini, dukungan manajemen dan budaya tempat kerja yang mendukung kolaborasi juga harus diperkuat. Mengembangkan sistem manajemen komponen yang efisien dan proses terstruktur dapat membantu mengatasi tantangan integrasi dan memastikan bahwa komponen digunakan secara optimal.

Kontribusi penelitian ini adalah untuk memberikan pemahaman yang lebih mendalam tentang bagaimana strategi penggunaan kembali dapat diterapkan dengan mempertimbangkan faktor teknis dan non-teknis. Studi ini juga menyoroti pentingnya dukungan organisasi dan manajerial untuk mengatasi hambatan yang ada dan memberikan wawasan berharga bagi praktik pengembangan perangkat lunak yang ingin menerapkan strategi tersebut untuk hasil yang lebih optimal.

Volume 10 Nomor 2
e-ISSN: 2477-5452; p-ISSN: 2460-1306
DOI: 10.37715/juisi.v10i2.4895

Daftar Pustaka

A. K. Tripathi and M. Gupta, "Risk analysis in reuse-oriented software development," Int. J. Inf. Technol. Manag., vol. 5, no. 1, pp. 52–65, 2006, doi: 10.1504/IJITM.2006.008713.

Álvarez Londoño, J. J., & Hurtado Alegría, J. A. (2020). An experimental design on the SPEM 2.0 process model element classification algorithm of the AVISPA tool through ANOVA variance analysis. Ingeniería Solidaria, 16(1). https://doi.org/10.16925/2357-6014.2020.01.09

A. Mateen, "Imported from A Software Reuse Approach and Its Effect On Software Quality, An Empirical Study for The Software Industry. (arXiv:1702.00125v1 [cs.SE]) http://arxiv.org/abs/1702.00125," vol. 7, no. 2, pp. 266–279, 2017.

A. P. Stephens, "Software reuse," Comput. Control Eng. J., vol. 13, no. 1, p. 40, 2002, doi: 10.48175/ijarsct-12087.

D. Dahiya and S. Dahiya, "Software reuse in design and development of aspects," Proc. - Int. Comput. Softw. Appl. Conf., pp. 745–750, 2008, doi: 10.1109/COMPSAC.2008.55.

D. M. Costa, E. N. Teixeira, and C. M. L. Werner, "A Repository to Support Software Process Reuse Based on Process Lines," ACM Int. Conf. Proceeding Ser., 2020, doi: 10.1145/3439961.3439962.

Epperson, A. Y. Wang, R. DeLine, and S. M. Drucker, "Strategies for reuse and sharing among data scientists in software teams," pp. 243–252, 2022, doi: 10.1145/3510457.3513042.

E. Nogueira Teixeira, F. A. Aleixo, F. D. de S. Amâncio, E. Oliveira, U. Kulesza, and C. Werner, "Software process line as an approach to support software process reuse: A systematic literature review," Inf. Softw. Technol., vol. 116, no. February, 2019, doi: 10.1016/j.infsof.2019.08.007.

F. Quella, A Holistic View of Software and Hardware Reuse.

G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," Requir. Eng., vol. 10, no. 1, pp. 34–44, 2005, doi: 10.1007/s00766-004-0194-4.

Institute of Data., (2023). What is Modularity in Software Engineering.

Kresnala, D. P., Padri, A. R., & Henderi. (2023). Rancang Bangun Laman Penyetaraan Ijazah Menggunakan Metode Reuse-Based Software Development. Technomedia Journal, 8(2), 276–292. https://doi.org/10.33050/tmj.v8i2.2116

Kumar, P., Scholar, R., & Abdul, A. P. J. (n.d.). Issue and Challenges in Component Testing in Component Based Software Engineering.

Laguna, M. A., González-Baixauli, B., López, O., & García, F. J. (2003). LNCS 2817 - Software Process and Reuse: A Required Unification. In LNCS (Vol. 2817).

R. Kamalraj, B. G. Geetha, and G. Singarave, "Reducing Efforts on Software Project Management Using Software Package Reusability," 2009 IEEE Int. Adv. Comput. Conf. IACC 2009, no. March, pp. 1624–1627, 2009, doi: 10.1109/IADCC.2009.4809260.

Shang, R., Lang, K., & Vragov, R. (2022). A Market-Based Approach to Facilitate the Organizational Adoption of Software Component Reuse Strategies. Communications of the Association for Information Systems, 51(1), 993–1016. https://doi.org/10.17705/1CAIS.05140 of Software Component Reuse Strategies. *Communications of the Association for Information Systems*, *51*(1), 993–1016. https://doi.org/10.17705/1CAIS.05140

U. H. Publication and D. Version, "INTERACT INTEGRATE," no. 2003, pp. 679–683, 2024.