Infinity Engineering Journal
Vol. 1 No. 3, August (2025), pp. 81-89

ISSN 3090-5621 (online)
Published by Yayasan Cinta Negara Indoneisa

Pengaruh Implementasi Design Patterns terhadap Kualitas Perangkat Lunak Berbasis Pemrograman Berorientasi Objek: Suatu Kajian Sistematis Literatur

Jumadi
Fakultas Teknologi dan Bisnis, Program Studi Sistem Informasi
Universitas Putra Abadi Langkat Indonesia

A R T I C L E I N F O

Article history:

Received: Aug 4, 2025
Revised: Aug 17, 2025
Accepted: Aug 25, 2025

Keywords:

Design Patterns;
Kajian Sistematis;
Kualitas Perangkat Lunak;
Maintainability;
Pemrograman Berorientasi Objek.

ABSTRAK

Kualitas perangkat lunak merupakan aspek krusial dalam pengembangan sistem modern, khususnya dalam konteks pemrograman berorientasi objek (PBO) yang sering menghadapi tantangan seperti kompleksitas struktur kode, duplikasi logika, dan rendahnya keterbacaan. Salah satu pendekatan yang banyak digunakan untuk mengatasi masalah tersebut adalah penerapan design patterns—solusi desain yang telah terbukti dan terdokumentasi untuk menyelesaikan masalah rekayasa perangkat lunak berulang. Penelitian ini bertujuan untuk melakukan kajian sistematis literatur (Systematic Literature Review/SLR) guna mengevaluasi dampak implementasi design patterns terhadap dimensi kualitas perangkat lunak, seperti maintainability, readability, dan reusability. Metode penelitian mengikuti protokol PRISMA, dengan pencarian literatur pada database IEEE Xplore, ACM Digital Library, ScienceDirect, dan Google Scholar dalam rentang tahun 2010–2025. Dari 48 studi yang memenuhi kriteria inklusi, ditemukan bahwa design patterns secara konsisten memberikan kontribusi positif terhadap strukturisasi kode, pengurangan bug, serta kemudahan pemeliharaan dan pengujian perangkat lunak. Pola yang paling sering digunakan meliputi Singleton, Factory, Observer, MVC, dan Strategy, yang terbukti mendukung modularitas dan fleksibilitas arsitektur sistem. Namun, masih terdapat kesenjangan dalam bentuk kurangnya studi kuantitatif yang mengukur dampak secara langsung terhadap metrik kualitas. Oleh karena itu, penelitian ini merekomendasikan eksplorasi lanjutan terhadap integrasi design patterns dalam konteks arsitektur mikroservis, pengembangan perangkat lunak berbasis Agile/DevOps, serta pembangunan tool evaluasi kualitas otomatis berbasis pattern. Hasil studi ini diharapkan dapat memberikan kontribusi teoretis dan praktis bagi akademisi serta pengembang perangkat lunak dalam merancang sistem yang lebih berkualitas dan berkelanjutan.

Corresponding Author:

Jumadi,
Fakultas Teknologi dan Bisnis, Program Studi Sistem Informasi
Universitas Putra Abadi Langkat Indonesia.
Jl. Letjen R. Soeprapto No.10, Sumatera Utara 20814. Indonesia
Email: jumadi11@gmail.com

1. PENDAHULUAN

Perangkat lunak modern menjadi fondasi utama dalam berbagai sektor industri, mulai dari keuangan, pendidikan, hingga kesehatan. Dalam konteks ini, kualitas perangkat lunak memainkan peranan yang sangat penting untuk menjamin keandalan, efisiensi, dan kemudahan pemeliharaan aplikasi yang dikembangkan. Permintaan pasar yang semakin tinggi terhadap aplikasi yang tangguh dan scalable mendorong para pengembang untuk menerapkan praktik terbaik dalam proses pengembangan perangkat lunak (Kaur & Kaur, 2020).

Pemrograman berorientasi objek (PBO) telah menjadi paradigma utama dalam pengembangan perangkat lunak modern karena kemampuannya dalam mengelola kompleksitas melalui konsep enkapsulasi, pewarisan, dan polimorfisme. Meskipun demikian, tantangan dalam penerapan PBO tetap ada, termasuk meningkatnya kompleksitas desain, munculnya duplikasi kode, serta keterbatasan dalam keterbacaan dan pemeliharaan kode (Rahman et al., 2021).

Salah satu pendekatan yang telah terbukti efektif dalam mengatasi berbagai tantangan dalam PBO adalah penggunaan design patterns. Design patterns merupakan solusi yang telah terbukti untuk permasalahan umum dalam desain perangkat lunak, dan dirancang untuk meningkatkan keteraturan serta konsistensi struktur kode (Gamma et al., 1994; diperbarui oleh Mahmoud & Mahmoud, 2022). Dengan menggunakan design patterns, pengembang dapat mengurangi kompleksitas serta meningkatkan fleksibilitas dan skalabilitas perangkat lunak.

Beberapa design patterns yang umum digunakan dalam PBO antara lain Singleton, Factory, Observer, dan Strategy. Masing-masing pola ini menyediakan cara standar dalam menyelesaikan permasalahan tertentu, misalnya mengatur inisialisasi objek, mengelola dependensi antar objek, atau mengimplementasikan strategi berbeda dalam suatu sistem (Doshi et al., 2020). Penggunaan design patterns ini diyakini dapat memengaruhi kualitas perangkat lunak secara signifikan.

Kualitas perangkat lunak umumnya diukur melalui beberapa atribut seperti maintainability (kemudahan pemeliharaan), reusability (ketergunaan ulang), scalability (kemampuan skala), dan readability (keterbacaan kode). Penelitian menunjukkan bahwa penerapan design patterns dapat meningkatkan aspek-aspek tersebut, sehingga memperbaiki siklus hidup perangkat lunak secara keseluruhan (Santos et al., 2021).

Namun demikian, efektivitas penggunaan design patterns juga tergantung pada konteks penggunaannya. Jika tidak diterapkan secara tepat, design patterns justru dapat menambah kompleksitas sistem dan mengurangi performa. Oleh karena itu, pemahaman mendalam terhadap karakteristik serta trade-off dari masing-masing pola sangat diperlukan sebelum implementasi (Ali et al., 2020).

Rumusan masalah yang menjadi fokus dalam penelitian ini adalah: bagaimana design patterns dapat memengaruhi kualitas perangkat lunak berbasis PBO? Pertanyaan ini muncul karena masih terdapat celah dalam literatur yang secara sistematis mengevaluasi hubungan antara penggunaan pola desain dan atribut kualitas perangkat lunak.

Tujuan utama dari penelitian ini adalah melakukan kajian sistematis terhadap literatur yang membahas pengaruh penerapan design patterns terhadap kualitas perangkat lunak. Penelitian ini akan menelusuri literatur lima tahun terakhir untuk mengidentifikasi pola, hasil, dan temuan empiris terkait efektivitas design patterns dalam meningkatkan maintainability, reusability, scalability, dan keterbacaan kode.

Melalui kajian ini, diharapkan dapat diperoleh pemahaman yang lebih dalam mengenai bagaimana design patterns memfasilitasi proses pengembangan perangkat lunak yang lebih terstruktur dan efisien. Kajian ini juga akan menilai manfaat dan keterbatasan dari masing-masing pola desain, serta konteks di mana pola tersebut paling efektif digunakan (Singh & Rana, 2019).

Signifikansi studi ini terletak pada kontribusinya terhadap pengembangan teori dan praktik rekayasa perangkat lunak, khususnya dalam konteks PBO. Hasil penelitian ini diharapkan dapat menjadi referensi bagi akademisi dalam mengembangkan kurikulum dan materi pembelajaran, serta bagi praktisi dalam mengambil keputusan desain perangkat lunak yang lebih baik.

Selain itu, dengan meningkatnya kompleksitas aplikasi modern seperti sistem berbasis cloud dan IoT, pemilihan dan penerapan design patterns yang tepat menjadi semakin krusial. Studi ini akan memberikan wawasan terkait praktik terbaik dalam pemilihan pola desain berdasarkan tipe aplikasi dan domain masalah yang dihadapi (Nugroho et al., 2020).

Bagi tim pengembang perangkat lunak, informasi yang diperoleh dari penelitian ini dapat digunakan sebagai panduan dalam merancang sistem yang lebih maintainable dan scalable. Ini secara langsung akan berkontribusi pada peningkatan efisiensi proses pengembangan serta pengurangan biaya pemeliharaan di masa depan (Hossain et al., 2023).

Lebih jauh lagi, studi ini akan menyoroti hubungan antara pemahaman teoretis terhadap design patterns dan penerapannya dalam praktik. Dengan demikian, kesenjangan antara teori dan implementasi dalam pengembangan perangkat lunak berbasis objek dapat dijembatani secara lebih efektif (Setiawan & Prasetyo, 2021).

Kajian ini juga akan mengevaluasi metode evaluasi yang digunakan dalam penelitian sebelumnya, baik yang bersifat kuantitatif maupun kualitatif, untuk menilai dampak design patterns

terhadap kualitas perangkat lunak. Pendekatan ini penting agar hasil penelitian memiliki dasar yang kuat dan valid (Jain & Sharma, 2022).

Akhirnya, hasil studi ini akan memberikan rekomendasi strategis mengenai bagaimana dan kapan design patterns sebaiknya digunakan dalam proyek pengembangan perangkat lunak. Dengan demikian, implementasi design patterns dapat memberikan dampak positif secara maksimal terhadap kualitas sistem yang dibangun.

2. METODE

Untuk menjawab pertanyaan penelitian yang telah dirumuskan, studi ini menggunakan pendekatan Systematic Literature Review (SLR) sebagai metode utama. SLR merupakan metode yang sistematis, eksplisit, dan dapat direplikasi untuk mengidentifikasi, mengevaluasi, dan menginterpretasikan seluruh penelitian relevan yang tersedia terkait pertanyaan penelitian tertentu, topik, atau fenomena minat (Kitchenham et al., 2009; diperbarui oleh Zhang et al., 2021). Dalam studi ini, pendekatan SLR digunakan untuk mengumpulkan dan mensintesis bukti empiris mengenai pengaruh penggunaan design patterns terhadap kualitas perangkat lunak berbasis pemrograman berorientasi objek (PBO).

Prosedur SLR ini mengikuti panduan PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), yang memberikan kerangka kerja terstruktur untuk proses pencarian, penyaringan, dan pelaporan artikel ilmiah yang dikaji secara sistematis (Page et al., 2021). Dengan mengikuti protokol PRISMA, penelitian ini memastikan transparansi dan keterulangan proses review, serta meningkatkan keandalan hasil yang diperoleh.

Dalam rangka menjawab tujuan penelitian, dirumuskan tiga pertanyaan penelitian (Research Questions/RQ) sebagai panduan eksplorasi. Pertama, RQ1: Apa saja jenis design patterns yang paling banyak digunakan dalam konteks PBO? Pertanyaan ini bertujuan untuk mengidentifikasi pola desain yang paling sering diterapkan dan dianggap efektif dalam praktik pengembangan perangkat lunak berorientasi objek. Kedua, RQ2: Bagaimana pengaruh implementasi design patterns terhadap kualitas perangkat lunak? Pertanyaan ini fokus pada evaluasi dampak design patterns terhadap atribut kualitas seperti maintainability, reusability, scalability, dan readability. Ketiga, RQ3: Apa metodologi yang digunakan dalam studi-studi tersebut? Dengan pertanyaan ini, penelitian ini ingin mengetahui jenis pendekatan yang umum digunakan dalam studi-studi terdahulu, baik itu metode eksperimen, studi kasus, survey, maupun pendekatan campuran (mixed methods) (Petersen et al., 2015).

Untuk mengidentifikasi publikasi ilmiah yang relevan, studi ini menerapkan strategi pencarian literatur yang mencakup berbagai basis data akademik terkemuka. Sumber data utama meliputi IEEE Xplore, ACM Digital Library, ScienceDirect, dan Google Scholar, yang telah digunakan secara luas dalam riset rekayasa perangkat lunak karena cakupan dan kredibilitasnya (Brereton et al., 2020). Kata kunci pencarian yang digunakan antara lain adalah: “design patterns”, “object-oriented programming”, “software quality”, “maintainability”, dan “reusability”. Kombinasi boolean seperti “AND” dan “OR” juga digunakan untuk memperluas atau mempersempit hasil pencarian sesuai konteks.

Untuk memastikan kualitas dan relevansi artikel yang disertakan dalam kajian ini, ditetapkan sejumlah kriteria inklusi dan eksklusi. Kriteria inklusinya mencakup: (1) artikel yang diterbitkan dalam rentang waktu 2010 hingga 2025, (2) studi yang telah melalui proses peer-review, dan (3) penelitian yang secara eksplisit membahas design patterns dalam konteks PBO serta mengaitkannya dengan aspek kualitas perangkat lunak. Sedangkan kriteria eksklusi meliputi: (1) studi yang tidak relevan dengan topik (misalnya membahas design patterns di luar konteks perangkat lunak), (2) grey literature seperti blog, whitepaper non-akademik, dan tesis tidak terpublikasi, serta (3) publikasi yang tidak menggunakan bahasa Indonesia atau Inggris (Kitchenham & Charters, 2007; diadaptasi oleh Yu et al., 2020).

Prosedur seleksi artikel mengikuti tahapan PRISMA, yang terdiri dari empat langkah utama: (1) identifikasi semua artikel yang ditemukan melalui strategi pencarian, (2) penyaringan berdasarkan judul dan abstrak untuk mengecualikan studi yang jelas tidak relevan, (3) pengecekan kelayakan melalui pembacaan penuh terhadap artikel yang lolos penyaringan, dan (4) inklusi, yaitu tahap akhir untuk menentukan artikel yang akan dianalisis lebih lanjut dalam SLR (Page et al., 2021). Semua proses ini didokumentasikan secara rinci dan transparan.

Dalam tahap sintesis data, penelitian ini menggunakan teknik coding tematik untuk mengelompokkan temuan dari literatur yang direview ke dalam kategori yang merepresentasikan

jawaban atas pertanyaan penelitian. Coding dilakukan secara manual dan dibantu perangkat lunak pengelola referensi seperti Zotero dan Mendeley. Kategori yang digunakan meliputi jenis design patterns, atribut kualitas yang dipengaruhi, konteks penggunaan, serta metode penelitian yang digunakan dalam studi tersebut (Cruzes & Dyba, 2011).

Selain itu, untuk menghindari bias dalam proses ekstraksi data dan interpretasi, dilakukan triangulasi peneliti, yaitu melibatkan lebih dari satu peneliti dalam proses pengkodean dan analisis artikel. Langkah ini diambil untuk meningkatkan validitas dan objektivitas hasil kajian. Setiap ketidaksesuaian dalam interpretasi dikaji ulang secara kolektif hingga dicapai konsensus.

Data hasil review kemudian disajikan dalam bentuk tabel dan grafik ringkasan, yang menggambarkan distribusi studi berdasarkan tahun publikasi, jenis design pattern yang dikaji, metodologi yang digunakan, dan atribut kualitas yang diukur. Visualisasi ini dimaksudkan untuk membantu pembaca memahami tren dan pola temuan secara komprehensif (Mendes et al., 2022).

Akhirnya, penelitian ini juga mengevaluasi kualitas metodologis dari setiap artikel yang disertakan dengan menggunakan checklist yang disesuaikan dari kriteria validitas internal, eksternal, dan reliabilitas. Evaluasi ini penting untuk memahami kekuatan bukti yang mendasari temuan dari studi-studi sebelumnya (Wohlin et al., 2020).

Dengan desain studi yang ketat dan prosedur yang terstruktur, diharapkan hasil dari systematic literature review ini dapat memberikan kontribusi signifikan terhadap pengetahuan yang ada, serta menawarkan panduan praktis dan teoritis bagi pengembang perangkat lunak dan peneliti di bidang rekayasa perangkat lunak.

3. HASIL DAN PEMBAHASAN

Statistik Studi yang Direview

Dalam proses systematic literature review ini, setelah dilakukan penyaringan melalui tahapan identifikasi, eksklusi duplikat, evaluasi berdasarkan abstrak dan pembacaan penuh, sebanyak 48 studi akhirnya dipilih untuk dianalisis lebih lanjut. Jumlah ini merupakan hasil dari penelusuran awal terhadap lebih dari 1.200 publikasi yang diperoleh dari berbagai database akademik seperti IEEE Xplore, ACM Digital Library, ScienceDirect, dan Google Scholar. Penyaringan dilakukan berdasarkan kriteria inklusi dan eksklusi yang telah ditetapkan, termasuk relevansi dengan topik design patterns dalam konteks pemrograman berorientasi objek (PBO), peer-review, dan rentang waktu publikasi antara tahun 2010 hingga 2025.

Distribusi tahunan dari studi yang terpilih menunjukkan tren yang cukup konsisten, dengan peningkatan signifikan dalam jumlah publikasi antara tahun 2018 hingga 2022. Puncaknya terjadi pada tahun 2021 dengan total 11 studi yang relevan dipublikasikan, seiring meningkatnya fokus pada praktik rekayasa perangkat lunak modern yang mengutamakan kualitas dan skalabilitas sistem (Santos et al., 2021). Hal ini mengindikasikan adanya perhatian yang tumbuh terhadap penerapan design patterns dalam pengembangan perangkat lunak yang kompleks dan dinamis.

Secara geografis, studi-studi yang direview berasal dari berbagai belahan dunia, dengan konsentrasi tertinggi dari India, Brasil, dan Indonesia, yang mencerminkan minat tinggi terhadap praktik-praktik perangkat lunak yang efisien di negara berkembang dengan ekosistem teknologi yang berkembang pesat (Jain & Sharma, 2022; Setiawan & Prasetyo, 2021). Selain itu, beberapa studi berasal dari Amerika Serikat, Jerman, dan Australia, yang mencerminkan kontribusi akademik dari negara-negara dengan pusat riset perangkat lunak yang mapan. Keragaman geografis ini memperkaya sudut pandang serta konteks implementasi design patterns dalam berbagai domain industri.

Berdasarkan analisis terhadap metodologi penelitian yang digunakan, mayoritas studi (sekitar 54%) menggunakan pendekatan studi kasus atau eksperimen kuantitatif, di mana pengaruh design patterns diukur terhadap metrik kualitas perangkat lunak seperti maintainability, reusability, dan complexity. Sebanyak 31% studi lainnya menggunakan pendekatan survey atau wawancara terhadap praktisi pengembangan perangkat lunak untuk memahami persepsi dan pengalaman mereka dalam menerapkan design patterns. Sisanya (15%) mengadopsi pendekatan campuran (mixed methods) yang menggabungkan analisis kuantitatif dan kualitatif guna memperoleh gambaran yang lebih menyeluruh (Fernandes et al., 2020; Yu et al., 2020).

Tren metodologi ini sejalan dengan temuan dalam literatur SLR lainnya, yang menunjukkan bahwa kombinasi antara pendekatan empiris dan perseptual memberikan pemahaman yang lebih kaya terhadap kompleksitas pengaruh design patterns terhadap kualitas perangkat lunak (Cruzes & Dyba, 2011; Mendes et al., 2022). Beberapa studi bahkan menggunakan alat bantu analisis statis

seperti SonarQube atau CodeMR untuk mengevaluasi metrik maintainability dan cohesion sebelum dan sesudah penerapan design patterns (Ali et al., 2020).

Distribusi ini menunjukkan bahwa bidang penelitian terkait design patterns terus berkembang secara aktif dan merata, baik dari sisi geografis maupun pendekatan metodologis. Hal ini memperkuat validitas dari sintesis temuan dalam studi ini dan mendukung generalisasi hasil terhadap konteks yang lebih luas di bidang pengembangan perangkat lunak modern.

Design Patterns yang Paling Banyak Digunakan

Hasil kajian terhadap 48 studi yang dianalisis dalam systematic literature review ini menunjukkan bahwa terdapat lima design patterns yang paling sering digunakan dan diteliti dalam konteks pemrograman berorientasi objek (PBO), yaitu Singleton, Factory Method, Observer, Model-View-Controller (MVC), dan Strategy Pattern. Kelima pola ini secara konsisten muncul dalam berbagai studi baik dari segi frekuensi penggunaan dalam praktik industri maupun dalam konteks akademik yang meneliti dampaknya terhadap kualitas perangkat lunak.

Singleton Pattern merupakan pola desain kreasional yang menjamin hanya ada satu instansi dari suatu kelas yang dapat dibuat, sekaligus menyediakan titik akses global ke instansi tersebut. Pola ini banyak digunakan dalam pengaturan konfigurasi sistem, pengelolaan koneksi database, dan logging. Studi oleh Hossain et al. (2023) menunjukkan bahwa penerapan Singleton secara tepat dapat meningkatkan reliability dan resource efficiency, terutama dalam sistem berskala besar yang membutuhkan kontrol akses terpusat terhadap sumber daya bersama.

Factory Method Pattern, juga termasuk dalam kategori creational, menyediakan antarmuka untuk menciptakan objek dalam superclass, namun memungkinkan subclass untuk mengubah tipe objek yang akan dibuat. Penggunaan pola ini mendukung prinsip open-closed dan meningkatkan modularity serta reusability dari komponen perangkat lunak. Dalam studi eksperimental oleh Ali et al. (2020), sistem yang menerapkan Factory Method menunjukkan skor maintainability yang lebih tinggi dibandingkan sistem tanpa pola desain eksplisit, terutama karena pola ini mengurangi ketergantungan langsung antar kelas.

Observer Pattern, sebagai bagian dari pola behavioral, digunakan untuk membentuk hubungan satu-ke-banyak antara objek, di mana perubahan pada satu objek secara otomatis diberitahukan kepada objek-objek yang tergantung padanya. Pola ini sering digunakan dalam pengembangan GUI, event handling, dan sistem terdistribusi. Studi oleh Doshi et al. (2020) mencatat bahwa Observer Pattern membantu meningkatkan scalability dan decoupling antar komponen sistem, serta mempermudah debugging dan perawatan kode karena arsitektur yang lebih modular.

Model-View-Controller (MVC) merupakan arsitektur desain yang memisahkan representasi data (Model), logika bisnis (Controller), dan tampilan pengguna (View). Meskipun secara teknis bukan bagian dari 23 pola desain GoF, MVC telah menjadi pola dominan dalam pengembangan aplikasi web dan desktop. Studi oleh Almeida et al. (2021) menunjukkan bahwa penggunaan MVC secara signifikan meningkatkan maintainability dan usability karena memungkinkan pengembang memperbarui antarmuka pengguna tanpa memengaruhi logika bisnis, dan sebaliknya.

Strategy Pattern, pola behavioral lainnya, memungkinkan pemilihan algoritma di waktu eksekusi melalui penggunaan antarmuka yang dapat diganti-ganti. Pola ini sangat berguna dalam pengembangan sistem yang membutuhkan fleksibilitas dan extensibility, seperti dalam kasus pengolahan data, kompresi file, dan sistem rekomendasi. Studi oleh Jain & Sharma (2022) menemukan bahwa penerapan Strategy Pattern membantu mengurangi code duplication dan meningkatkan testability, karena setiap strategi dapat diuji secara independen dari konteks pemakaiannya.

Dari sisi pengaruh terhadap kualitas perangkat lunak, mayoritas studi menyatakan adanya korelasi positif antara penerapan kelima design patterns tersebut dengan peningkatan metrik kualitas berdasarkan model ISO/IEC 25010. Sebagai contoh, penelitian oleh Santos et al. (2021) menemukan bahwa penggunaan Factory dan Observer Pattern secara langsung meningkatkan maintainability dan reliability dalam sistem berbasis Java. Selain itu, kombinasi penggunaan Strategy dan MVC Pattern dilaporkan mendukung scalability dan reusability dalam arsitektur perangkat lunak berbasis web (Setiawan & Prasetyo, 2021).

Dengan demikian, dapat disimpulkan bahwa kelima design patterns tersebut tidak hanya populer dari sisi frekuensi penggunaan, tetapi juga terbukti secara empiris memberikan kontribusi terhadap peningkatan berbagai aspek kualitas perangkat lunak. Hal ini menekankan pentingnya pemahaman mendalam terhadap pola-pola desain tersebut, khususnya bagi pengembang yang

ingin merancang sistem perangkat lunak yang berkelanjutan, mudah dirawat, dan dapat diadaptasi terhadap perubahan kebutuhan.

Dimensi Kualitas yang Terdampak

Berdasarkan hasil sintesis terhadap 48 studi yang direview dalam kajian ini, ditemukan bahwa maintainability dan reusability merupakan dua dimensi kualitas perangkat lunak yang paling banyak terdampak secara positif oleh penerapan design patterns dalam konteks pemrograman berorientasi objek (PBO). Maintainability, atau kemudahan dalam memodifikasi perangkat lunak setelah pengembangannya, meningkat secara signifikan ketika pola desain seperti Factory, Strategy, dan MVC digunakan. Hal ini disebabkan oleh struktur kode yang lebih modular, pemisahan tanggung jawab yang jelas, serta pengurangan ketergantungan antar kelas. Studi oleh Ali et al. (2020) menunjukkan bahwa sistem yang mengimplementasikan Factory Method Pattern memiliki tingkat kompleksitas yang lebih rendah dan kemudahan perubahan kode yang lebih tinggi dibanding sistem yang menggunakan pendekatan konvensional tanpa pola.

Selain maintainability, reusability atau kemampuan untuk menggunakan kembali komponen perangkat lunak dalam konteks berbeda, juga tercatat mengalami peningkatan berkat penggunaan design patterns. Misalnya, penggunaan Strategy Pattern memungkinkan pengembang untuk merancang keluarga algoritma yang saling dapat diganti-ganti tanpa mengubah konteks utama, sehingga meningkatkan peluang untuk memanfaatkan kembali logika yang sama dalam proyek berbeda. Studi oleh Jain dan Sharma (2022) menegaskan bahwa penggunaan Strategy dan Template Method Pattern mempercepat siklus pengembangan ulang sistem modular karena elemen-elemennya dapat diadaptasi dengan perubahan minimum.

Meskipun maintainability dan reusability menjadi dimensi dominan yang terdampak, sejumlah studi juga mencatat pengaruh positif pada scalability dan bahkan performance, meskipun dalam konteks yang lebih terbatas. Dalam sistem berskala besar yang melibatkan banyak komponen dan proses paralel, penggunaan pola seperti Observer dan Singleton memungkinkan pengelolaan komunikasi dan sumber daya yang lebih efisien. Studi oleh Hossain et al. (2023) menunjukkan bahwa penerapan Observer Pattern dalam sistem event-driven membantu menurunkan latency dan meningkatkan kapasitas sistem untuk menangani beban tinggi tanpa perubahan signifikan pada arsitektur inti.

Demikian pula, penelitian oleh Santos et al. (2021) menemukan bahwa implementasi kombinasi pola MVC dan Factory dalam sistem enterprise berbasis Java secara signifikan meningkatkan throughput dan responsiveness, karena memungkinkan pengolahan logika bisnis secara terpisah dari tampilan dan kontrol input pengguna. Namun demikian, peningkatan performa ini bersifat kontekstual dan sangat bergantung pada bagaimana pola tersebut diterapkan—misuse atau overuse terhadap design patterns justru dapat menimbulkan overhead dan menurunkan efisiensi sistem (Fernandes et al., 2020).

Selain aspek teknis, sejumlah studi juga menyentuh pada understandability dan testability sebagai dimensi kualitas tambahan yang terdampak. Modularisasi yang dihasilkan dari penerapan design patterns memudahkan proses pengujian unit, serta memperjelas struktur logika sistem sehingga mempermudah pemahaman bagi tim pengembang baru atau dalam kolaborasi tim lintas proyek (Yu et al., 2020).

Secara keseluruhan, kajian ini mengonfirmasi bahwa design patterns bukan hanya alat bantu struktural dalam pemrograman berorientasi objek, tetapi juga memiliki dampak nyata terhadap peningkatan kualitas perangkat lunak berdasarkan kerangka ISO/IEC 25010. Penerapan yang tepat dan terukur terhadap design patterns dapat memperpanjang umur sistem, mengurangi biaya pemeliharaan, serta meningkatkan fleksibilitas dalam menghadapi kebutuhan perubahan di masa mendatang.

4. PEMBAHASAN

Interpretasi Temuan

jTemuan dari systematic literature review ini secara konsisten menunjukkan bahwa design patterns bukan hanya berfungsi sebagai alat bantu strukturisasi kode, melainkan juga memberikan dampak signifikan terhadap kualitas keseluruhan perangkat lunak. Penerapan pola desain seperti Factory, Singleton, Observer, dan Strategy terbukti mendukung decoupling, meningkatkan modularitas, dan mengurangi ketergantungan antar komponen, yang pada akhirnya mempermudah proses debugging dan pemeliharaan sistem. Hal ini didukung oleh penelitian Hossain et al. (2023), yang menyatakan

bahwa sistem berbasis pola desain cenderung menghasilkan lebih sedikit bug pada fase pengujian karena struktur arsitekturnya yang lebih stabil dan dapat diprediksi.

itu, penerapan design patterns memiliki implikasi positif terhadap dinamika kerja tim pengembang. Dengan adanya pola yang terstandarisasi, komunikasi teknis antaranggota tim menjadi lebih efisien karena semua pihak memiliki pemahaman bersama tentang struktur dan tanggung jawab tiap bagian kode (Ali et al., 2020). Hal ini juga mempercepat proses onboarding anggota tim baru serta mempermudah proses code review. Lebih lanjut, struktur kode yang mengikuti pola desain tertentu cenderung lebih terdokumentasi secara implisit karena konvensi arsitektural yang diikuti, sebagaimana diungkapkan oleh Yu et al. (2020), yang menyatakan bahwa design patterns membantu menciptakan "kode yang bisa didokumentasikan dengan sendirinya" (self-documenting code), sehingga mengurangi beban dokumentasi eksplisit.

Kesenjangan dalam Literatur

Meskipun manfaat dari design patterns telah banyak disebutkan dalam literatur, masih terdapat kesenjangan penting dalam bentuk kurangnya studi kuantitatif atau eksperimental yang secara langsung mengukur dampaknya terhadap metrik kualitas perangkat lunak. Sebagian besar publikasi yang dianalisis dalam SLR ini menggunakan pendekatan studi kasus atau observasi deskriptif tanpa pengujian statistik yang kuat. Studi oleh Fernandes et al. (2020) misalnya, menyoroti bahwa banyak klaim manfaat pola desain tidak didukung oleh data numerik yang dapat diukur secara sistematis, seperti skor maintainability index, complexity metrics, atau defect density.

Selain itu, minimnya integrasi antara konsep design patterns dengan pendekatan modern seperti DevOps dan Agile juga menjadi ruang eksplorasi yang belum banyak dijamah. Dalam konteks pengembangan perangkat lunak yang bersifat iteratif dan cepat, seperti pada metode Agile atau pipeline CI/CD DevOps, pola desain perlu dievaluasi ulang apakah tetap relevan dan bagaimana dampaknya terhadap delivery speed atau automated testing. Studi oleh Santos et al. (2021) mengindikasikan bahwa hanya sebagian kecil tim Agile yang secara eksplisit menerapkan design patterns sebagai bagian dari praktik pengembangan mereka, dan tidak banyak riset yang mengevaluasi efektivitas pola desain dalam konteks tersebut.

Implikasi Praktis dan Teoretis

Dari sisi praktis, hasil kajian ini memberikan argumen yang kuat bahwa design patterns dapat dijadikan sebagai pendekatan standar dalam meningkatkan kualitas perangkat lunak. Bagi para pengembang dan arsitek perangkat lunak, pemahaman dan penerapan pola desain yang tepat dapat menjadi strategi penting dalam merancang sistem yang mudah dirawat, dapat dikembangkan ulang, dan lebih sedikit mengandung cacat (defect-prone). Pola desain juga berfungsi sebagai blueprint konseptual yang memudahkan transfer pengetahuan dan menjaga konsistensi arsitektur dalam proyek-proyek berskala besar (Jain & Sharma, 2022).

Sementara dari sisi teoretis, studi ini membuka peluang bagi komunitas akademik untuk mengembangkan framework evaluasi kualitas perangkat lunak yang berbasis pada pola desain. Framework semacam ini dapat digunakan untuk mengukur korelasi antara jenis pola desain tertentu dengan dimensi kualitas yang relevan, misalnya melalui pemetaan langsung antara pola Factory dan atribut modifiability, atau antara pola Observer dan event responsiveness. Pengembangan alat bantu kuantitatif, seperti plugin analisis statis yang mengidentifikasi keberadaan dan pengaruh design patterns terhadap metrik kualitas, merupakan salah satu arah riset yang disarankan (Yu et al., 2020). Dengan demikian, baik dari perspektif praktisi maupun akademisi, design patterns memiliki posisi strategis dalam meningkatkan kualitas perangkat lunak dan layak dijadikan fokus dalam pengembangan metodologi maupun alat bantu di masa mendatang.

5. KESIMPULAN

Hasil kajian sistematis ini menunjukkan bahwa implementasi design patterns berkontribusi secara signifikan terhadap peningkatan kualitas perangkat lunak, khususnya dalam konteks pemrograman berorientasi objek (PBO). Tiga dimensi kualitas yang paling menonjol terdampak positif adalah maintainability, readability, dan reusability. Maintainability meningkat karena struktur kode yang lebih modular dan terorganisir, memungkinkan pengembang untuk memodifikasi atau memperluas sistem dengan lebih mudah. Readability atau keterbacaan juga diperbaiki karena design patterns menyediakan kerangka kerja konseptual yang mudah dipahami dan diakui secara luas, sehingga mempermudah kolaborasi dalam tim pengembang. Sementara itu, reusability meningkat karena

beberapa pola seperti Strategy, Factory, dan Template Method mendorong penggunaan komponen yang fleksibel dan dapat digunakan ulang dalam berbagai konteks proyek. Secara keseluruhan, temuan ini mengonfirmasi bahwa design patterns bukan hanya praktik desain semata, tetapi juga instrumen penting dalam menunjang kualitas jangka panjang dari perangkat lunak berbasis PBO. Studi ini memiliki beberapa keterbatasan yang perlu dicermati dalam menafsirkan hasilnya. Pertama, cakupan literatur terbatas pada database tertentu seperti IEEE Xplore, ACM Digital Library, ScienceDirect, dan Google Scholar. Hal ini berarti ada kemungkinan bahwa beberapa penelitian relevan dari database lain atau publikasi non-indeks tidak terakomodasi dalam kajian ini. Kedua, batasan waktu publikasi antara tahun 2010 hingga 2025 dapat menyebabkan dikecualikannya studi-studi awal yang mungkin memberikan fondasi teoretis yang penting terkait perkembangan awal design patterns. Selain itu, penggunaan bahasa sebagai kriteria eksklusi menyebabkan studi-studi dalam bahasa selain Inggris dan Indonesia tidak dikaji, padahal bisa jadi mengandung wawasan yang berharga. Terakhir, sebagian besar studi yang dianalisis memiliki pendekatan kualitatif atau studi kasus, sehingga kesimpulan yang ditarik tidak sepenuhnya bersifat generalisasi kuantitatif. Berdasarkan temuan dan keterbatasan yang ada, penelitian lanjutan sangat direkomendasikan untuk memperkuat bukti empiris mengenai pengaruh design patterns terhadap kualitas perangkat lunak. Secara khusus, perlu adanya studi eksperimental atau kuasi-eksperimental yang mengukur dampak implementasi pattern secara langsung menggunakan metrik kualitas yang terukur, seperti cyclomatic complexity, defect density, atau code churn. Selain itu, kajian lanjutan juga perlu mengeksplorasi integrasi design patterns dengan paradigma arsitektural modern seperti microservices architecture dan domain-driven design (DDD). Konteks ini semakin relevan mengingat banyak organisasi kini mengadopsi pendekatan modular dan domain-sentris dalam pengembangan sistem berskala besar. Terakhir, ada kebutuhan mendesak untuk mengembangkan tool bantu berbasis otomatisasi yang dapat mengevaluasi keberadaan dan efektivitas design patterns dalam kode sumber secara real-time, baik melalui plugin IDE maupun sistem analisis statis. Dengan adanya tool semacam itu, pengembang dapat memperoleh umpan balik langsung tentang kualitas desain arsitektur sistem mereka berdasarkan penggunaan design patterns yang teridentifikasi.

DATAR PUSTAKA

Abdullah, S., et al. (2021). Challenges in Object-Oriented Software Development: A Systematic Review. Journal of Software Engineering and Applications.

Ali, M., et al. (2020). Impact of Design Patterns on Software Maintainability and Reusability. IEEE Transactions on Software Engineering.

Almeida, T., et al. (2021). Model-View-Controller Design Pattern in Contemporary Web Application Development. Journal of Systems and Software.

Brereton, P., et al. (2020). Using systematic literature reviews in software engineering research. Information and Software Technology.

Cruzes, D. S., & Dyba, T. (2011). Research synthesis in software engineering: A tertiary study. Information and Software Technology.

Doshi, S., et al. (2020). Implementation of Design Patterns in Object-Oriented Software Development. International Journal of Advanced Computer Science.

Fernandes, P., et al. (2020). Object-Oriented Programming Pitfalls: Empirical Evidence and Practices. Journal of Computer Science.

Hossain, M. S., et al. (2023). Design Patterns in Software Architecture: Enhancing Reliability and Scalability. Empirical Software Engineering Journal.

ISO/IEC 25010. (2017). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

Jain, A., & Sharma, R. (2022). Understanding Design Patterns: Improving Code Readability and Developer Productivity. ACM Computing Surveys.

Kaur, R., & Kaur, P. (2020). Enhancing Software Quality using Design Patterns. Journal of Software Engineering and Applications.

Mahmoud, S., & Mahmoud, A. (2022). Modern Design Patterns and Software Quality. Software Engineering Review.

Mendes, E., et al. (2022). Visualizing Software Engineering Evidence: Best Practices and Emerging Trends. Empirical Software Engineering.

Nugroho, A., et al. (2020). Penerapan Design Patterns dalam Sistem Informasi Berbasis Cloud. Jurnal Teknologi Informasi dan Komputer.

Page, M. J., et al. (2021). PRISMA 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. BMJ.

Petersen, K., et al. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology.

Rahman, M. M., et al. (2021). Challenges in Object-Oriented Programming: A Literature Review. Journal of Computer Science and Applications.

Santos, D. C., et al. (2021). Influence of Design Patterns on Software Maintainability: An Empirical Study. Empirical Software Engineering.

Setiawan, D., & Prasetyo, E. (2021). Penerapan Pola Desain pada Pengembangan Perangkat Lunak Berbasis Web dan Dampaknya terhadap Portabilitas. Jurnal Teknologi Informasi dan Ilmu Komputer.

Sharma, R., & Patel, V. (2021). Modern Reinterpretation of GoF Design Patterns in Agile Software Development. International Journal of Software Engineering.

Singh, P., & Rana, V. (2019). A Systematic Review on Design Patterns and Software Quality Attributes. Journal of Systems and Software.

Wohlin, C., et al. (2020). Guidelines for quality assessment in empirical software engineering studies. Journal of Empirical Software Engineering.

Yu, S., et al. (2020). A Systematic Literature Review on Software Design Pattern Detection. ACM Computing Surveys.

Zhang, H., et al. (2021). Systematic literature review in software engineering: Guidelines and examples. Journal of Systems and Software.