# Pengembangan Aplikasi Terpusat Berbasis Modular Monolith Architecture Studi Kasus: UKM Peduli Kemanusiaan

Muhammad Hafid Afandi[1], I Made Suartana[2]

[1,2] S1 Teknik Informatika, Universitas Negeri Surabaya
[1]muhammadhafid.22099@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— UKM Peduli Kemanusiaan (UKM-PK) UNESA mengalami kendala pengelolaan data anggota, kegiatan, dan donasi secara manual dan terpencar, menyebabkan redundansi, inkonsistensi, dan sulit koordinasi. Dibutuhkan sistem terpusat yang mudah dipelihara dan dikembangkan lintas periode kepengurusan. Penelitian ini menerapkan dan mengevaluasi arsitektur modular monolith sebagai solusi, dipilih karena menyeimbangkan kesederhanaan monolith dan modularitas microservices tanpa kompleksitas distribusi berlebihan untuk skala UKM. Sistem dikembangkan dengan Go dan pendekatan hexagonal architecture pada sembilan modul: Dashboard, Authentication, User Management, Company Profile, Recruitment, Activity Progress, Donation, Inventory Assets, dan Documentation. Evaluasi kuantitatif menggunakan custom AST analyzer untuk metrik coupling, cohesion, dan maintainability index, lalu dibandingkan dengan arsitektur monolith dan microservices pada domain serupa. Hasilnya, seluruh modul mencapai coupling rata-rata 1,111 (rendah) dengan message coupling (tipe terbaik), cohesion tinggi, dan maintainability index sangat mudah dipelihara. Dibandingkan kedua arsitektur lain, modular monolith menunjukkan keseimbangan optimal antara modularitas terukur dan kemudahan pemeliharaan, sehingga terbukti sesuai untuk organisasi skala UKM.**

***Kata Kunci*— modular monolith, hexagonal architecture, coupling, cohesion, maintainability index, metrik kualitas perangkat lunak**

## I. PENDAHULUAN

Unit Kegiatan Mahasiswa (UKM) merupakan organisasi yang mewadahi aktivitas mahasiswa dalam mengembangkan minat, bakat, dan keterampilan di lingkungan kampus [1]. UKM Peduli Kemanusiaan (UKM-PK) Universitas Negeri Surabaya (UNESA) merupakan salah satu UKM yang berfokus pada kegiatan sosial, penggalangan dana, dan program kerelawanan. Saat ini, pengelolaan data anggota, kegiatan, donasi, dan pelaporan masih dilakukan secara manual menggunakan *shared drive* yang terpisah, sehingga rawan redundansi data, inkonsistensi pencatatan, dan kesulitan koordinasi, . Kondisi serupa juga ditemukan pada UKM di Universitas Negeri Manado, yang dalam pengelolaan data, perekrutan pengurus, dan penyebaran informasi masih menggunakan cara konvensional yang dinilai kurang efektif [2].

Sistem terpusat menjadi solusi yang relevan karena mampu mengintegrasikan berbagai sumber data menjadi satu platform digital[3]. Salah satu tantangan utama dalam merancang aplikasi terpusat adalah bagaimana menyusun struktur internal yang tetap modular, sehingga pengelolaan, pengujian, dan pengembangan fitur dapat dilakukan secara terpisah tanpa mengganggu keseluruhan sistem[4].

Kebutuhan akan sistem terpusat yang modular dan maintainable mendorong pemilihan arsitektur yang tepat. Arsitektur *monolith* menawarkan kesederhanaan implementasi, namun mengalami masalah *tight coupling* dan sulit dipelihara ketika sistem berkembang. Arsitektur *microservices* menawarkan modularitas tinggi tetapi memerlukan infrastruktur terdistribusi yang kompleks dan tidak proporsional untuk organisasi berskala UKM [5].

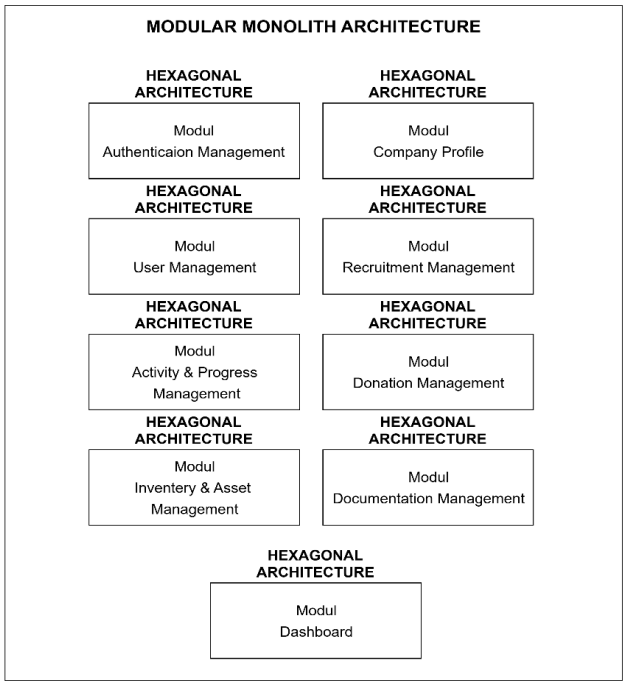
Arsitektur modular *monolith* hadir sebagai pendekatan pragmatis dengan satu deployment unit namun dirancang modular dengan batas domain yang jelas per modul [6]. Pendekatan ini memungkinkan tim kecil mengadopsi prinsip dekomposisi layanan seperti *microservices* tanpa kompleksitas infrastruktur. Meskipun demikian, evaluasi kuantitatif kualitas arsitektur modular *monolith* menggunakan metrik perangkat lunak masih jarang dilakukan, khususnya dalam konteks non-industri seperti UKM [7].

Penelitian ini mengisi kesenjangan tersebut dengan mengimplementasikan dan mengevaluasi arsitektur modular *monolith* pada studi kasus UKM-PK menggunakan metrik yang dapat digunakan untuk mengevaluasi kualitas perangkat lunak yaitu *coupling* (CBM, WCBM, ACBM, Ca, Ce, *Instability*), *cohesion* (LCOM5, NCAM), dan *Maintainability Index* (MI)[8].

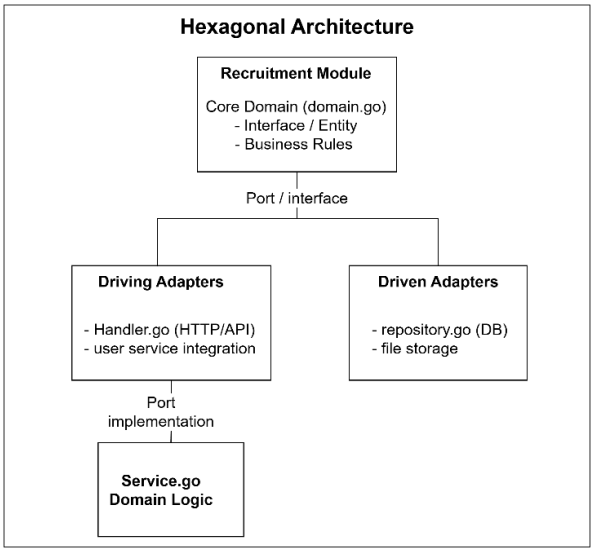
## II. METODE PENELITIHAN

Penelitian ini menggunakan pendekatan *Research and Development* (R&D) dengan studi kasus tunggal pada UKM Peduli Kemanusiaan UNESA, yang alurnya terdiri dari enam tahap berurutan, dimulai dari identifikasi masalah melalui observasi dan wawancara langsung dengan ketua UKM-PK untuk memahami kendala pengelolaan data yang masih manual, kemudian dilanjutkan dengan studi literatur mengenai modular *monolith*, *hexagonal architecture*, dan metrik kualitas perangkat lunak, lalu perencanaan sistem yang mencakup analisis kebutuhan fungsional dan non-fungsional, identifikasi *bounded context*, pembuatan *use case diagram*, *class diagram*, *ERD*, dan *wireframe*, setelah itu implementasi sistem *backend* menggunakan bahasa pemrograman Go dengan *hexagonal architecture* pada 9 modul bisnis, diikuti pengujian sistem melalui *blackbox* testing fungsional dan analisis metrik kualitas kode menggunakan *custom AST analyzer*, dan diakhiri dengan penarikan kesimpulan serta saran.

A. *Desain Modular Monolith dengan Hexagonal Architecture*




Gambar 1 Modular Monolith Architecture




Gambar 2 Hexagonal Architecture

Seperti yang terlihat digambar 1 dan 2, sistem dirancang menggunakan arsitektur modular *monolith* di mana seluruh modul berada dalam satu *deployment* unit namun memiliki batas domain yang eksplisit. Setiap modul mengimplementasikan *hexagonal architecture* (*ports and adapters*) dengan struktur empat lapisan yang konsisten[9]. *Domain layer* berisi *entity* dan *domain error* yang tidak bergantung pada *framework* atau *database*. *Ports layer* mendefinisikan kontrak *interface* sebagai batas isolasi modul dan semua dependensi antar modul hanya boleh melewati *port* yang terdefinisi. *Application layer* mengimplementasikan logika bisnis melalui *service* yang hanya berinteraksi melalui *interface* tersebut. *Infrastructure layer* mengimplementasikan *repository* dengan GORM dan menangani HTTP *request* melalui *handler* menggunakan *framework* Gin.

Sistem mencakup sembilan modul bisnis yaitu, *Dashboard, Authentication, User Management, Company Profile, Recruitment, Activity Progress, Donation, Inventory Assets,* dan *Documentation*. Komunikasi antar modul dilakukan eksklusif melalui *adapter pattern*, modul yang membutuhkan data dari modul lain mendefinisikan *external port*-nya sendiri, kemudian *adapter* mengimplementasikan *port* tersebut dengan membungkus *service* modul asal. Dengan demikian tidak ada *import* langsung antar domain modul, sehingga batas isolasi terjaga secara ketat dan setiap modul dapat dikembangkan dan diuji secara independen.

B. *Pengujian Sistem*

Pengujian sistem dilakukan melalui dua pendekatan utama, yaitu pengujian fungsional dengan metode *blackbox testing* dan pengujian kualitas kode menggunakan *custom* AST *analyzer*. *Blackbox testing* dilakukan untuk memastikan bahwa setiap fitur sistem berfungsi sesuai dengan kebutuhan fungsional yang telah ditetapkan, dengan menguji *input* dan *output* tanpa melihat struktur internal kode. Sementara itu, *custom* AST *analyzer* yang dibuat secara mandiri dengan basis *package* go/ast digunakan untuk menganalisis kualitas kode secara otomatis dengan menghitung metrik *coupling*, *cohesion*, dan *maintainability index*, guna memastikan bahwa kode yang dihasilkan terstruktur dengan baik dan mudah dipelihara.

1) *Coupling*

*Coupling* mengukur tingkat ketergantungan antar modul. Tiga metrik utama yang digunakan adalah CBM, WCBM, dan ACBM. CBM (*Coupling Between Microservices/Modul*) merupakan metrik biner yang bernilai 1 apabila terdapat interaksi antara dua modul dan 0 apabila tidak ada. WCBM (*Weighted* CBM) mengukur total frekuensi interaksi, sedangkan ACBM (*Absolute* CBM) mengukur total interaksi dua arah[10]. Selain itu digunakan metrik *Instability* (*I*) pada (1) yang diturunkan dari *Afferent Coupling* (*Ca*) dan *Efferent Coupling* (*Ce*) dengan rumus [11].

$$I = Ce / (Ca + Ce) \quad (1)$$

Nilai *I* pada (1) berkisar antara 0 hingga 1. Nilai mendekati 0 menunjukkan modul yang stabil (banyak digunakan namun tidak bergantung pada modul lain), sedangkan nilai mendekati 1 menunjukkan modul yang *instable*. Selain nilai numerik, kualitas *coupling* juga dievaluasi berdasarkan tipenya menggunakan taksonomi Page-Jones, *message coupling* (terbaik)*, data coupling, stamp coupling, control coupling, common coupling,* hingga *content coupling* (terburuk)[12].

2) *Cohesion*

*Cohesion* mengukur sejauh mana elemen dalam satu modul bekerja bersama mencapai tujuan tertentu. Penelitian ini menggunakan dua metrik *Cohesion*. Pertama, LCOM5 (*Lack of Cohesion in Methods*) pada tingkat *class*, dihitung dengan rumus [13]:

$$LCOM5 = (a - k \times l) / (l - k \times l) \quad (2)$$

Pada (2) *k* adalah jumlah *method*, *l* adalah jumlah attribute (*field*), dan *a* adalah total jumlah attribute yang diakses oleh seluruh *method*. Nilai mendekati 0 menunjukkan kohesi tinggi, nilai mendekati 1 menunjukkan kohesi rendah.

*Kedua*, *NCAM* (*Normalized Cohesion Among Classes of Microservice*) pada tingkat modul sebagai adaptasi dari metrik CAM [10]. *NCAM* dihitung dengan rumus:

$$NCAM(m) = 1 - Avg(CAM(c)) \quad (3)$$

*Nilai NCAM* pada (3) jika mendekati 0 menunjukkan kohesi modul yang tinggi. Dengan menggunakan *LCOM5* dan *NCAM* secara bersamaan, evaluasi *cohesion* dapat dilakukan pada dua granularitas yaitu tingkat *class* dan tingkat modul.

*3) Maintainability Index*

*Maintainability Index* (*MI*) adalah metrik kuantitatif yang mengukur seberapa mudah sistem perangkat lunak dapat dipelihara. *MI* dihitung menggunakan formula [14]:

$$MI = 171 - 5{,}2 \times \ln(aveV) - 0{,}23 \times aveCC - 16{,}2 \times \ln(aveLOC) \quad (4)$$

Pada (4) *aveV* adalah rata-rata *Halstead Volume* per modul, *aveCC* adalah rata-rata *Cyclomatic Complexity* per modul, dan *aveLOC* adalah rata-rata *Lines of Code* per modul. *MI* dihitung per file kemudian dirata-ratakan per modul. Interpretasi nilai *MI* adalah *MI* lebih dari 84 berarti "sangat mudah dipelihara" *MI* 65–84 berarti "dapat dipelihara" dan *MI* 65 berarti "sulit dipelihara".

## III. Hasil dan Pembahasan

### A. *Implementasi Sistem*

Sistem berhasil diimplementasikan dengan 9 modul bisnis dalam struktur *hexagonal architecture*, *Dashboard*, *Authentication*, *User Management*, *Company Profile*, *Recruitment*, *Activity Progress*, *Donation*, *Inventory Assets*, dan *Documentation*. Implementasi ditunjukkan dari struktur folder pada tabel IV berikut.

Tabel I
Struktur Folder Hasil Implemenasi

```
internal/
├── app/bootstrap/        ← Inisialisasi aplikasi
├── modules/
│   ├── activity_progress/ ← Modul kegiatan & laporan
│   ├── authentication/   ← Modul autentikasi & role
│   ├── company_profile/  ← Modul company profile
│   ├── documentation/    ← Modul dokumentasi
│   ├── donation/         ← Modul donasi
│   ├── inventory_assets/ ← Modul aset & peminjaman
│   ├── recruitment/      ← Modul rekrutmen
│   └── users/            ← Modul manajemen anggota
│       ├── application/
│       │   ├── dtos/     ← Data Transfer Objects
│       │   ├── ports/    ← Interface (services & repositories)
│       │   └── services/ ← Business logic
│       ├── domain/       ← Entities & domain errors
│       ├── infrastructure/
│       │   ├── adapters/ ← Anti-corruption layer
│       │   └── persistence/
│       │       ├── models/      ← GORM models
│       │       └── repositories/ ← Implementasi repository
│       └── interface/
│           ├── module.go ← Dependency injection
│           ├── handlers/ ← HTTP handlers
└──         └── routes/   ← Route registration
```

Setiap modul dalam modular *monolith* bersifat *self-contained*, dependensi *internal* modul diselesaikan di dalam modul melalui mekanisme *ports* dan *adapters*. *Bootstrap* aplikasi cukup memanggil NewModuleX(...) dan RegisterRoutes(...) tanpa mengetahui detail internal modul, sehingga batas isolasi antar domain terjaga dengan ketat.

### B. *Hasil Pengujian Fungsional*

Pengujian *blackbox* testing dilakukan terhadap 82 skenario pada seluruh 9 modul. Seluruh skenario lulus dengan tingkat kelulusan 100%, tidak ditemukan satu pun kegagalan fungsional. Pengujian mencakup skenario *happy path* (*input* valid) dan skenario *error* (*input* tidak valid, akses tidak terotorisasi, data tidak ditemukan) pada seluruh endpoint REST API *backend*.

Tabel II
Ringkasan Hasil Blackbox Testing per Modul

| Modul / Fitur | Skenario | Passed | Failed | Lulus% |
|---|---|---|---|---|
| Authentication | 6 | 6 | 0 | 100% |
| Role Management | 5 | 5 | 0 | 100% |
| Division Management | 5 | 5 | 0 | 100% |
| User Management | 10 | 10 | 0 | 100% |
| Activity Management | 6 | 6 | 0 | 100% |
| Progress Report | 4 | 4 | 0 | 100% |
| LPJ | 4 | 4 | 0 | 100% |
| Documents | 3 | 3 | 0 | 100% |
| Documentation Management | 7 | 7 | 0 | 100% |
| Donation Management | 7 | 7 | 0 | 100% |
| Asset Management | 6 | 6 | 0 | 100% |
| Loan Management | 6 | 6 | 0 | 100% |
| Recruitment Management | 7 | 7 | 0 | 100% |
| Contents (Company Profile) | 8 | 8 | 0 | 100% |
| Dashboard | 1 | 1 | 0 | 100% |
| **Total Keseluruhan** | **82** | **82** | **0** | **100%** |

### C. *Hasil Metrik Coupling*

Tabel III
Hasil Pengukuran Metrik Coupling per Modul

| Modul | CBM | WCBM | ACBM | Ca | Ce | Tipe Coupling |
|---|---|---|---|---|---|---|
| *authentication* | 0 | 0 | 2 | 1 | 0 | Message Coupling |
| *activity_progress* | 0 | 0 | 6 | 2 | 0 | Message Coupling |
| *company_profile* | 0 | 0 | 0 | 0 | 0 | Message Coupling |
| *dashboard* | 5 | 145 | 146 | 1 | 5 | Message Coupling |
| *donation* | 1 | 1 | 38 | 1 | 1 | Message Coupling |
| *inventory_assets* | 1 | 2 | 10 | 1 | 1 | Message Coupling |
| *recruitment* | 1 | 6 | 43 | 1 | 1 | Message Coupling |
| *users* | 1 | 2 | 69 | 3 | 1 | Message Coupling |
| *documentation* | 1 | 2 | 2 | 0 | 1 | Message Coupling |

Dari Tabel VI, terlihat bahwa 7 dari 9 modul memiliki CBM ≤ 1, artinya setiap modul hanya berinteraksi dengan paling banyak satu modul lain melalui *port* antarmuka yang terdefinisi. Modul dashboard dengan CBM = 5 merupakan pengecualian yang dapat diterima secara arsitektural karena perannya sebagai modul *aggregator* yang mengambil data ringkasan dari seluruh modul bisnis.

Seluruh 9 modul menggunakan *Message Coupling* yang merupakan tipe *coupling* terbaik dalam taksonomi Page-Jones, tanpa satu pun instance tipe *coupling* yang lebih buruk (*Stamp, Control, Common,* atau *Content Coupling*). Hal ini mengkonfirmasi bahwa implementasi *hexagonal architecture* berhasil mengendalikan tipe *coupling* di seluruh modul. Nilai *instability* rata-rata sistem adalah 0,398, mencerminkan keseimbangan moderat antara stabilitas dan fleksibilitas yang selaras dengan *Stable Dependencies Principle* (SDP).

D. *Hasil Metrik Cohesion*

TABEL IV
HASIL PENGUKURAN METRIK COHESION PER MODUL

| Modul | NCAM | LCOM5 avg | LCOM5 terburuk | Kategori NCAM |
|---|---|---|---|---|
| *authentication* | 0,333 | 0,800 | 0,800 | Kohesi Tinggi |
| *activity_progress* | 0,050 | 0,642 | 0,844 | Kohesi Tinggi |
| *company_profile* | 0,333 | 0,400 | 0,400 | Kohesi Tinggi |
| *dashboard* | 0,000 | 0,000 | 0,000 | Kohesi Tinggi |
| *donation* | 0,366 | 0,559 | 0,722 | Kohesi Tinggi |
| *inventory_assets* | 0,190 | 0,677 | 0,844 | Kohesi Tinggi |
| *recruitment* | 0,195 | 0,895 | 0,895 | Kohesi Tinggi |
| *users* | 0,050 | 0,458 | 0,569 | Kohesi Tinggi |
| *documentation* | 0,396 | 0,615 | 0,722 | Kohesi Tinggi |

Dari Tabel VII, nilai NCAM seluruh modul berkisar antara 0,000 hingga 0,396, seluruhnya berada pada kategori Kohesi Modul Tinggi (NCAM < 0,5). Nilai NCAM terendah (0,000) dimiliki oleh modul *dashboard* yang hanya berisi DTO agregasi tanpa logika bisnis kompleks.

Nilai LCOM5 rata-rata per modul berkisar 0,000–0,895. Modul recruitment memiliki LCOM5 tertinggi (0,895) pada *service class* utamanya, yang dapat dipahami karena satu *service class* mengelola 20 *use case* rekrutmen dalam satu *bounded context*, karakteristik alami *hexagonal architecture* di mana isolasi *domain* diprioritaskan di atas granularitas *class*. Meskipun LCOM5 class-level lebih tinggi, nilai NCAM modul recruitment (0,195) tetap dalam kategori Kohesi Modul Tinggi, mengkonfirmasi bahwa kohesi fungsional modul secara keseluruhan tetap terjaga.

E. *Hasil Maintainability Index*

TABEL V
HASIL PENGUKURAN METRIK MAINTAINABILITY INDEX PER MODUL

| Modul | MI Rata-rata | Rentang MI | Kategori |
|---|---|---|---|
| *authentication* | 109,81 | 109,81 – 109,81 | Sangat Mudah Dipelihara |
| *activity_progress* | 102,27 | 86,15 – 120,09 | Sangat Mudah Dipelihara |
| *company_profile* | 112,41 | 97,02 – 125,44 | Sangat Mudah Dipelihara |
| *dashboard* | 110,31 | 61,32 – 163,03 | Sangat Mudah Dipelihara* |
| *donation* | 101,25 | 90,18 – 115,74 | Sangat Mudah Dipelihara |
| *inventory_assets* | 102,55 | 88,32 – 125,12 | Sangat Mudah Dipelihara |
| *recruitment* | 105,80 | 91,04 – 123,55 | Sangat Mudah Dipelihara |
| *users* | 98,24 | 80,12 – 118,35 | Sangat Mudah Dipelihara |
| *documentation* | 107,71 | 97,38 – 122,07 | Sangat Mudah Dipelihara |

Dari Tabel VIII, seluruh 9 modul mencapai MI rata-rata dalam rentang 98,24 hingga 112,41, seluruhnya jauh melampaui ambang batas kategori Sangat Mudah Dipelihara (MI ≥ 85). Modul *company_profile* memperoleh MI tertinggi (112,41) karena cakupan *domain* yang terkonsentrasi dengan fungsi-fungsi pendek dan CC rendah. Modul *users* memiliki MI terendah (98,24) namun tetap sangat baik meski mengelola subdomain terbesar dengan 3.945 baris kode.

Satu-satunya file yang berada di bawah ambang batas adalah *dashboard_service.go* dengan MI = 61,32, disebabkan oleh tingginya *Halstead Volume* akibat *agregasi* data dari lima modul dalam satu fungsi *GetDashboard*(). File ini adalah pengecualian tunggal dari 9 modul dan disadari sebagai area perbaikan di masa mendatang.

## IV. KESIMPULAN

Penelitian ini berhasil mengimplementasikan dan mengevaluasi arsitektur modular *monolith* dengan *hexagonal architecture* pada aplikasi terpusat UKM Peduli Kemanusiaan UNESA. Berdasarkan hasil pengujian, dapat ditarik tiga kesimpulan utama.

Pertama, dari sisi coupling, seluruh 9 modul mencapai CBM rata-rata 1,111 dengan seluruh interaksi menggunakan *Message Coupling*, tipe terbaik dalam taksonomi Page-Jones tanpa satu pun *instance* tipe *coupling* yang lebih buruk. Pola *instability* yang teridentifikasi selaras dengan *Stable Dependencies Principle*, modul fondasi bersifat stabil (I = 0,00) sementara modul bisnis *leaf* memiliki *instability* yang proporsional.

Kedua, dari sisi *cohesion*, nilai NCAM seluruh modul berkisar 0,000–0,396, seluruhnya pada kategori Kohesi Modul Tinggi. LCOM5 modul-level tertinggi dimiliki *recruitment* (0,895) yang dapat diinterpretasikan sebagai karakteristik alami *service class* dalam *hexagonal architecture*.

Ketiga, dari sisi *maintainability*, MI rata-rata sistem adalah 105,59 dengan rentang 98,24–112,41 di seluruh modul, seluruhnya melampaui ambang batas sangat mudah dipelihara (MI ≥ 85). Modular monolith menunjukkan keseimbangan optimal antara modularitas terukur, kohesi fungsional tinggi, dan *maintainability* yang baik dengan kompleksitas *deployment* setara *monolith*. Pendekatan ini terbukti tepat untuk sistem terpusat berskala organisasi UKM.

Berdasarkan hasil penelitian dan keterbatasan tersebut, beberapa saran diajukan untuk pengembangan sistem dan penelitian selanjutnya. Penelitian ini berfokus pada evaluasi kualitas kode melalui metrik statis, sehingga penelitian selanjutnya disarankan untuk melengkapi dengan pengujian performa dinamis seperti *load testing* dan *stress testing* menggunakan k6 atau Apache JMeter, serta membandingkan performa arsitektur modular *monolith* dengan *microservices*. Selain itu, beberapa *service class* menunjukkan nilai LCOM5 yang tergolong buruk, yaitu *recruitmentService* (0,895), *activityService* (0,844), dan *authService* (0,800), sehingga pengembangan selanjutnya dapat menerapkan pola CQRS atau decomposisi *service class* menjadi beberapa *class* yang lebih kohesif untuk menurunkan nilai LCOM5 tanpa mengorbankan struktur *bounded context*. Terakhir, file dashboard_service.go merupakan satu-satunya file yang masuk kategori Sulit Dipelihara (MI = 61,32) akibat *halstead volume* yang tinggi dari agregasi lima modul, sehingga disarankan untuk menerapkan *caching* agregasi data atau memecah fungsi GetDashboard() menjadi beberapa sub-fungsi yang lebih kecil guna meningkatkan nilai *maintainability index*.

## REFERENSI

[1] Hi Samiun, S. A., Abdullah, S. D., & Sirajuddin, H. K. (2022). Sistem Informasi Pengelolaan Kegiatan UKM di Lingkungan Universitas Khairun Ternate Berbasis Web.

[2] Mamahani, & Kembuan. (2024). Sistem Informasi Manajemen UKM Universitas Negeri Manado

[3] Prastyo, D., Irawan, D., & Mursyidin, I. H. (2025). Sistem Informasi Terpusat untuk Manajemen Dokumen, Penelitian, dan Pengabdian kepada Masyarakat. *Bit-Tech*, *7*(3), 758–769. https://doi.org/10.32877/bt.v7i3.2182

[4] Ait Said, M., Belouaddane, L., Mihi, S., & Ezzati, A. (2025). Modulith Architecture: Adoption Patterns, Challenges, and Emerging Trends. *International Journal of Computing and Digital Systems*, *17*(1). https://doi.org/10.12785/ijcds/1571016597

[5] Blinowski, G., Ojdowska, A., & Przybyłek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, 10, 20357–20374

[6] Taras, Y., & Segiy, D. (2023). Modular Monolith Architecture: Principles and Practical Implementation. Software Engineering Journal.

[7] Pappula, K. K., & Anasuri, S. (2022). Modular Monoliths in Practice: A Middle Ground for Growing Product Teams. International Journal of Emerging Trends in Computer Science and Information Technology, 3.

[8] Kevin, A., & Kurniawan Christian, H. (2023). Analisis Pengaruh Design Pattern Terhadap Pemeliharaan Perangkat Lunak Learning Management System. *Jurnal Telematika*, *18*(1).

[9] Tsechelidis, M., Nikolaidis, N., Maikantis, T., & Ampatzoglou, A. (2023). Modular Monoliths the way to Standardization. ACM International Conference Proceeding Series, 49–52. https://doi.org/10.1145/3624486.3624506

[10] Hasan, M. H., Hafeez Osman, M., Admodisastro, N. I., & Muhammad, S. (2023). From Monolith to Microservice: Measuring Architecture Maintainability. In IJACSA) International Journal of Advanced Computer Science and Applications (Vol. 14, Number 5). www.ijacsa.thesai.org

[11] Hellhake, D., Bogner, J., Schmid, T., & Wagner, S. (2022). Towards using coupling measures to guide black-box integration testing in component-based systems. Software Testing Verification and Reliability, 32(4). https://doi.org/10.1002/stvr.1811

[12] Thaiya, M. S., Julia, K., & Mbugua, S. (2022). On Software Modular Architecture: Concepts, Metrics and Trends. International Journal of Computer and Organization Trends, 12, 3–10. https://doi.org/10.14445/22492593/IJCTT-V12I1P302

[13] HANER KIRĞIL, E. N., & ERÇELEBİ AYYILDIZ, T. (2021). Analysis of Lack of Cohesion in Methods (LCOM): A Case Study. International Informatics and Software Engineering Conference.

[14] Syed-Mohamad, S. M., Ngah, A., Ali, A.-F. M., & Keikhosrokiani, P. (2025). Measuring Software Maintainability: An Exploration of Metrics and Continuous Practices. Journal of Advanced Research in Applied Sciences and Engineering Technology, 63(2), 181–195. https://doi.org/10.37934/araset.63.2.181195