https://jurnal.pustakagalerimandiri.co.id/index.php/pustakadata
DOI: https://doi.org/10.55382/jurnalpustakadata.v6i4.2033

Jurnal Pustaka Data
Pusat Akses Kajian Database, Analisa Teknologi, dan Arsitektur Komputer
Vol. 6 No. 4 (2026) 628 – 637
E ISSN: 2809-7483

Otomatisasi Kerangka Kode dari Class Diagram: Pendekatan Generasi Berbasis Model

Alia Julyanti[1], Nadya Clarisa Az-zahra[2], Muhammad Ainul Yaqin[3]
[1,2,3]Universitas Islam Negeri Maulana Malik Ibrahim, Malang, Indonesia.
[1]julyantialia@gmail.com, [2]nadyazahra3333@gmail.com, [3]yaqinov@ti.uin-malang.ac.id

Abstract

It remains unknown how class diagram complexity quantitatively affects Large Language Model (LLM) generated code structural fidelity. This study evaluates GPT-4 reliability in automating Pascal skeleton code generation using Model-Driven Software Engineering (MDSE). Designs from academic, hospital, and e-commerce domains were evaluated based on Weighted Methods per Class (WMC), class count, and relation density using Jaccard Similarity. Results show GPT-4 achieves perfect conformity (1.00) in low-complexity designs. However, accuracy drops to 0.935 (document) and 0.932 (image) in medium-scale models. The lowest accuracy occurs in high-complexity designs, dropping to 0.845 (document) with a 15.5% deviation rate, and 0.910 (image) due to over-generation hallucinations. This study contributes an empirical threshold of total WMC <= 15, beyond which LLM reliability drops below 0.85 Jaccard similarity.

Keywords: class diagram, GPT-4, model-driven software engineering, skeleton code, WMC

Abstrak

Hingga saat ini belum diketahui bagaimana kompleksitas class diagram memengaruhi kesesuaian struktural kode hasil luaran Large Language Model (LLM) secara kuantitatif. Penelitian ini mengevaluasi keandalan GPT-4 dalam mengotomatisasi generasi skeleton code Pascal menggunakan pendekatan Model-Driven Software Engineering (MDSE). Desain dari domain akademik, rumah sakit, dan e-commerce dievaluasi berdasarkan Weighted Methods per Class (WMC), jumlah kelas, dan kepadatan relasi menggunakan Jaccard Similarity. Hasil pengujian menunjukkan GPT-4 mencapai kesesuaian sempurna (1,00) pada kompleksitas rendah. Namun, akurasi menurun menjadi 0,935 (dokumen) dan 0,932 (gambar) pada skala menengah. Akurasi terendah terjadi pada skala tinggi (e-commerce), merosot hingga 0,845 (dokumen) dengan tingkat deviasi 15,5%, dan 0,910 (gambar) akibat halusinasi over-generation. Penelitian ini memberikan kontribusi berupa ambang batas empiris total WMC <= 15, di mana di atas nilai tersebut reliabilitas LLM turun di bawah 0,85 Jaccard similarity.

Kata kunci: *class diagram, GPT-4, model-driven software engineering, skeleton code, WMC*

Submitted : 08-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

1. Pendahuluan

Class Diagram merupakan salah satu komponen esensial dalam Unified Modeling Language (UML) yang merepresentasikan struktur statis dari sebuah sistem perangkat lunak berorientasi objek (Engineering, 2005). Dalam praktiknya, proses konversi rancangan arsitektur menjadi implementasi kode program secara manual memakan waktu yang lama dan sangat rentan terhadap kesalahan manusia (*human error*). Untuk mengatasi inefisiensi tersebut, pendekatan Model-Based Software Engineering (MBSE) diterapkan guna menjembatani dan mengotomatisasi transformasi dari spesifikasi desain tingkat tinggi menuju implementasi kode secara sistematis (Kalyanasundaram & P. Ugale, 2015). Melalui siklus MBSE, struktur pada Class Diagram dikonversi menjadi representasi Data Dictionary yang kemudian ditransformasikan secara otomatis menjadi kerangka kode program (*skeleton code*). Dalam studi ini, *skeleton code* didefinisikan secara teknis sebagai struktur fundamental program yang mencakup deklarasi kelas, atribut beserta tipe datanya, *signature* metode, serta kerangka relasi antar entitas (seperti pewarisan dan asosiasi), namun secara tegas tidak mencakup implementasi logika bisnis internal atau algoritma fungsional di dalam metode tersebut. Dalam praktiknya di industri rekayasa perangkat lunak, proses pembuatan kerangka kode ini umumnya dilakukan menggunakan berbagai kategori instrumen bantu, diawali dengan *Computer-Aided Software Engineering* (CASE) *Tools* seperti Sybase PowerDesigner yang mengekstraksi diagram UML menjadi draf kode (Ozkaya, 2019). Pendekatan lain meliputi pemanfaatan *Scaffolding Tools* bawaan *framework*, semacam antarmuka baris perintah (CLI) Artisan pada ekosistem pengembangan PHP, serta pemakaian asisten pemrograman terintegrasi seperti GitHub Copilot (Nguyen & Nadi, 2022).

Seiring dengan pesatnya kemajuan kecerdasan buatan, Large Language Model (LLM) seperti ChatGPT kini mendominasi berbagai tugas rekayasa perangkat lunak (Hou et al., 2024). Model generatif ini secara empiris terbukti memiliki kapabilitas kognitif untuk memahami teks maupun struktur rancangan guna menghasilkan kode, serta mampu menjaga keterlacakan dua arah (*bidirectional traceability*) antara dokumen desain konseptual dengan representasi pemrograman tingkat lanjut (Kanuka et al., 2023). Studi terdahulu mencatat bahwa LLM mampu menghasilkan akurasi sintaksis hingga 90-95% pada generasi kode berskala kecil, namun tingkat keberhasilan dan pemahaman instruksinya menurun ketika dihadapkan pada arsitektur berdimensi tinggi yang memuat banyak ketergantungan relasional (Hou et al., 2024).

Meskipun pemanfaatan LLM menawarkan otomatisasi yang menjanjikan, tingkat keandalannya belum teruji secara komprehensif ketika model dihadapkan pada fluktuasi tingkat kerumitan desain. Oleh karena itu, penelitian ini merumuskan dua pertanyaan penelitian utama: (RQ1) Bagaimana pengaruh tingkat kompleksitas *Class Diagram* yang diukur dengan metrik WMC terhadap tingkat akurasi *skeleton code* yang dihasilkan oleh ChatGPT?; dan (RQ2) Pada nilai ambang batas (*threshold*) WMC berapakah akurasi transformasi *Model-to-Code* mulai mengalami penurunan signifikan yang ditandai dengan halusinasi struktural?

Sepengetahuan kami, belum ada studi terdahulu yang mengintegrasikan dua parameter pengukuran secara bersamaan untuk menguji limitasi LLM; yaitu penggunaan metrik berorientasi objek Weighted Methods per Class (WMC) untuk mengkuantifikasi kompleksitas diagram masukan (Kemerer, 1993), yang disandingkan dengan evaluasi koefisien Jaccard Similarity guna memvalidasi tingkat akurasi struktur kode keluaran AI (Pradeep T et al., 2025). Oleh karena itu, penelitian ini bertujuan untuk mengukur secara presisi pengaruh kompleksitas desain terhadap akurasi transformasi pada tiga domain sistem berskala berbeda (akademik, rumah sakit, dan e-commerce). Selain itu, penelitian ini juga mengkomparasikan efektivitas dua mode masukan (*prompting modality*), yakni masukan berbasis teks (*Data Dictionary*) dan masukan visual (gambar *Class Diagram*) (Wu et al., 2023). Sebagai kontribusi praktis, penelitian ini merumuskan rekomendasi konkret terkait nilai ambang batas WMC yang aman bagi LLM saat ini—yakni direkomendasikan pada rentang nilai WMC ≤ 10 guna meminimalisir risiko reduksi elemen (*under-generation*) maupun fabrikasi sintaks (*over-generation*).

1.1 TINJAUAN PUSTAKA

Beberapa studi utama telah menjadi landasan dalam riset ini dan memetakan celah penelitian (*research gap*) yang ada. Sebuah penelitian melalui Survei Literatur Sistematis (SLR) menemukan bahwa LLM terbukti kapabel dalam mengotomatisasi berbagai siklus pembuatan kode dalam rekayasa perangkat lunak, namun studi tersebut belum menguji batasan performa LLM secara spesifik berdasarkan metrik

kompleksitas UML (Hou et al., 2024). Meskipun kapabilitas LLM juga telah dieksplorasi untuk menerjemahkan teks dokumen spesifikasi terstruktur menjadi kode program , pemetaan langsung dari model desain visual berorientasi objek yang kompleks masih memerlukan pengujian empiris.Di sisi lain, sebuah studi yang mengeksplorasi pengukuran skala *Class Diagram* membuktikan bahwa metrik WMC merupakan indikator prediktif yang valid untuk menakar kerumitan kelas, akan tetapi kajian ini berdiri sendiri dan tidak dihubungkan dengan proses transformasi otomatis (*code generation*) berbasis AI (Alfan Rosid, 2020).

Selanjutnya, penelitian terkait evaluasi kesamaan struktur program menunjukkan bahwa algoritma *Jaccard Similarity* beroperasi secara sangat efisien tanpa bias posisional, namun pengujiannya hanya diimplementasikan pada komparasi program buatan manusia, bukan pada validasi luaran model bahasa generatif (Song et al., 2024).

Meskipun demikian, sebagaimana yang terlihat pada sintesis literatur tersebut, belum ada satu pun studi yang secara eksplisit dan simultan mengkombinasikan metrik WMC dan koefisien *Jaccard Similarity* untuk mengevaluasi batasan performa LLM pada berbagai fluktuasi tingkat kompleksitas desain. Kajian terdahulu berfokus secara terpisah pada kemampuan AI secara umum (Hou et al., 2024), perhitungan metrik independen (Alfan Rosid, 2020), atau algoritma komparasi teks (Keivanloo et al., 2014). Oleh karena itu, integrasi instrumen evaluasi ini di dalam kerangka *Model-Based Software Engineering* menjadi sangat krusial untuk membuktikan secara empiris titik kegagalan ChatGPT dalam mengotomatisasi kerangka kode berdasarkan skalabilitas masukan modelnya.

2. Metode Penelitian

Penelitian ini menggunakan pendekatan eksperimen kuantitatif untuk menganalisis pengaruh tingkat kompleksitas *Class Diagram* terhadap kualitas *skeleton code* yang dihasilkan secara otomatis oleh *Large Language Model* (LLM). Proses penelitian dilaksanakan melalui tahapan transformasi model arsitektur sistem berbasis *Unified Modeling Language* (UML) menjadi kerangka kode program berstandar Pascal. Fokus utama eksperimen ini diarahkan pada analisis korelasi antara metrik kompleksitas *Class Diagram* dengan tingkat kemiripan arsitektur pada kode hasil generasi kecerdasan buatan dibandingkan dengan kode yang disusun secara manual. Rangkaian tahapan riset mencakup penyusunan rancangan model sistem, konversi *Class Diagram* ke dalam representasi *Data Dictionary* (Fouad & Mohamed, 2018), pengukuran kompleksitas arsitektur (H.M & Nandakumar, 2016), eksekusi transformasi dari model ke kode menggunakan ChatGPT (Wang et al., 2024) serta evaluasi akurasi transformasi dengan memanfaatkan metode *Jaccard Similarity* (Song et al., 2024).

2.1 Data dan Variabel Penelitian

Data yang digunakan dalam penelitian ini bersumber dari tiga representasi *Class Diagram* UML yang mewakili domain sistem dengan tingkat kompleksitas yang saling berbeda, yakni sistem akademik, sistem rumah sakit, dan sistem e-commerce. Ketiga rancangan diagram tersebut dirancang secara mandiri oleh peneliti berdasarkan purwarupa studi kasus riil yang lazim ditemui dalam praktik industri rekayasa perangkat lunak. Perancangan mandiri ini dilakukan guna memastikan bahwa parameter kompleksitas dapat dikontrol, diisolasi, dan dievaluasi secara presisi.

Secara spesifik, karakteristik ukuran komponen arsitektur untuk setiap domain ditetapkan secara berjenjang. Sistem Akademik merepresentasikan skala rendah dengan komposisi 4 kelas, 10 atribut, dan 8 metode. Selanjutnya, Sistem Rumah Sakit merepresentasikan skala menengah dengan komposisi 7 kelas, 16 atribut, dan 14 metode. Sementara itu, Sistem E-Commerce mewakili skala tinggi dengan komposisi masif sebanyak 16 kelas, 22 atribut, dan 22 metode. Pemilihan ketiga domain dengan rentang spektrum kerumitan yang berjenjang tersebut didasarkan pada kebutuhan untuk mengevaluasi kualitas dan konsistensi transformasi kode. Setiap *Class Diagram* kemudian dikonversi melalui perangkat lunak PowerDesigner untuk menghasilkan *Data Dictionary*, yaitu sebuah representasi terstruktur yang secara eksplisit mendefinisikan kelas, atribut, metode, dan relasi sistem sebagai basis masukan utama dalam proses generasi kode (Fouad & Mohamed, 2018).

Variabel bebas dalam studi ini adalah tingkat kompleksitas struktural dari *Class Diagram*, sementara variabel terikatnya difokuskan pada kualitas *skeleton code* yang diproduksi secara otomatis oleh sistem. Pengukuran variabel kompleksitas didefinisikan melalui akumulasi jumlah kelas, kuantitas metode, kepadatan relasi antar kelas, serta skor *Weighted Methods per Class* (WMC) (Alfan Rosid, 2020). Kualitas hasil transformasi selanjutnya diukur secara kuantitatif berdasarkan tingkat kesesuaian komposisi sintaks antara *skeleton code* otomatis dengan *skeleton code* manual (Pradeep T et al., 2025).

2.2 Desain Eksperimen

Eksperimen dirancang untuk melakukan komparasi struktural secara langsung antara *skeleton code* yang di-generate secara otomatis oleh ChatGPT dengan kerangka kode acuan yang dikembangkan secara

manual berdasarkan referensi *Class Diagram* yang ekuivalen (Monteiro et al., 2025).Tahapan awal eksperimen bertumpu pada ekstraksi informasi arsitektur yang meliputi deklarasi kelas, atribut, metode, dan relasi, yang kemudian disintesiskan ke dalam bentuk *Data Dictionary* (Fouad & Mohamed, 2018). Representasi kamus data ini selanjutnya bertindak sebagai instruksi struktural absolut bagi model bahasa untuk memproduksi kerangka program Pascal (Wang et al., 2024). Hasil transformasi tersebut kemudian disandingkan dengan kerangka kode manual guna mengkalkulasi tingkat kesesuaian arsitektur pemrograman secara menyeluruh. Pengujian pada beberapa domain sistem yang bervariasi ini dirancang agar mampu meminimalisir bias dan menyajikan representasi analitik yang jauh lebih objektif terkait intervensi kerumitan arsitektur terhadap kualitas transformasi model ke kode.

2.2 Pengukuran Kompleksitas

Prosedur pengukuran kompleksitas arsitektur model dilaksanakan dengan mengimplementasikan metrik *Weighted Methods per Class* (WMC) (Maina et al., 2022).WMC merupakan metrik fundamental dalam pemodelan berorientasi objek yang merepresentasikan akumulasi tingkat kerumitan berdasarkan jumlah metode di dalam seluruh hierarki kelas sebuah sistem (Kemerer, 1993). Nilai WMC diperoleh dengan menjumlahkan kuantitas seluruh metode yang terdefinisi pada masing-masing kelas. Rumus WMC ditunjukkan sebagai berikut:

$$WMC = \sum_{i=1}^{n} m_i$$

dengan mi menyatakan jumlah metode pada kelas ke-i, sedangkan n menyatakan jumlah total kelas dalam sistem (Alfan Rosid, 2020). Selain kalkulasi berbasis WMC, parameter kompleksitas juga ditinjau berdasarkan rasio populasi kelas dan kepadatan jumlah relasi antar kelas. Penggabungan ketiga indikator tersebut digunakan untuk memproyeksikan kerumitan rancangan perangkat lunak secara lebih komprehensif.

2.4 Proses Transformasi Model ke Kode

Mekanisme transformasi rancangan desain berorientasi objek menuju abstraksi kode diotomatisasi dengan memposisikan dokumen *Data Dictionary* sebagai masukan referensial utama (Fouad & Mohamed, 2018). Berdasarkan kamus data yang merinci nama kelas, tipe data atribut, operasi metode, dan relasi asosiatif tersebut, model LLM diinstruksikan untuk merumuskan *skeleton code* berbasis bahasa Pascal tanpa harus mengimplementasikan logika bisnis internal secara lengkap (Wang et al., 2024).

Guna menjamin reproduksibilitas (kemampuan replikasi) eksperimen oleh peneliti lain di masa mendatang, instruksi (*prompt*) yang diumpankan kepada ChatGPT dirancang secara spesifik menggunakan teknik *zero-shot prompting* (Hou et al., 2024). Untuk skenario pertama, *prompt* secara verbatim yang diinputkan bersamaan dengan unggahan *file* dokumen *Data Dictionary* adalah sebagai berikut: *"Bertindaklah sebagai Software Engineer profesional. Baca lampiran file Data Dictionary ini dan konversikan isinya menjadi skeleton code dalam bahasa pemrograman Pascal. Buatlah deklarasi kelas, atribut beserta tipe datanya, dan metode kosong tanpa implementasi logika bisnis internal. Terapkan juga konsep OOP seperti relasi inheritance, asosiasi, atau komposisi secara presisi sesuai dengan instruksi di dalam file."*

Sementara itu, untuk skenario masukan visual, *prompt* verbatim yang diinputkan bersamaan dengan unggahan *file* gambar *Class Diagram* berbunyi: *"Analisis gambar UML Class Diagram terlampir secara teliti. Hasilkan skeleton code dalam bahasa pemrograman Pascal murni berdasarkan gambar tersebut. Buatlah deklarasi kelas, atribut, dan metode kosong. Pastikan kamu mengidentifikasi dan mengimplementasikan relasi antar kelas (seperti asosiasi, agregasi, atau pewarisan) dengan benar sesuai dengan arah panah pada gambar."*

Sebagai pedoman validasi, sebuah kumpulan *skeleton code* juga disusun secara mandiri (manual) dengan merujuk pada diagram UML yang identik. Kerangka manual ini diberlakukan sebagai kode acuan empiris untuk mengevaluasi sejauh mana LLM mampu mereplikasi keabsahan arsitektur aslinya. Guna menjamin reliabilitas dan konsistensi eksperimen, seluruh tahapan transformasi ini dieksekusi dengan *Standard Operating Procedure* (SOP) parametrikal yang presisi pada setiap domain sistem.

Menggunakan gambar 1 untuk mengenerate skeleton code hasilnya adalah

Skeleton code

```
program SistemKRS_OOP;
uses crt;
type
  { ===== CLASS DOSEN ===== }
  Dosen = object
    nip  : integer;
    nama : string;
    procedure inputDosen;
    procedure tampilDosen;
  end;
  { ===== CLASS MATA KULIAH ===== }
  MataKuliah = object
    kodeMk : integer;
    namaMk : string;
    sks    : integer;
    dosen  : ^Dosen;  { relasi ke dosen }
    procedure inputMK;
    procedure tampilMK;
  end;
```

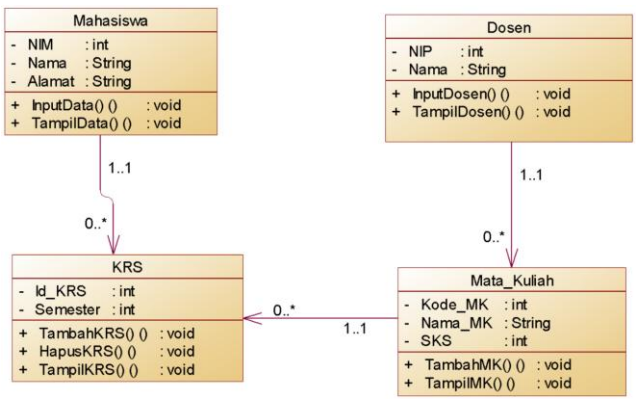
```
{ ===== CLASS MAHASISWA ===== }
Mahasiswa = object
  nim     : integer;
  nama    : string;
  alamat  : string;
  procedure inputMhs;
  procedure tampilMhs;
end;
{ ===== CLASS KRS ===== }
KRS = object
  idKrs    : integer;
  semester : integer;
  mhs      : ^Mahasiswa;    { relasi ke mahasiswa }
  mk       : ^MataKuliah;   { relasi ke matkul }
  procedure inputKRS;
  procedure tampilKRS;
end;
{ ================= IMPLEMENTASI METHOD
================= }
{ Dosen }
procedure Dosen.inputDosen;
begin
end;
procedure Dosen.tampilDosen;
begin
end;
{ Mata Kuliah }
procedure MataKuliah.inputMK;
begin
end;
procedure MataKuliah.tampilMK;
begin
end;
{ Mahasiswa }
procedure Mahasiswa.inputMhs;
begin
end;
procedure Mahasiswa.tampilMhs;
begin
end;
{ KRS }
procedure KRS.inputKRS;
begin
end;
procedure KRS.tampilKRS;
begin
end;
```

Sebagai pedoman validasi, sebuah kumpulan *skeleton code* juga disusun secara mandiri (manual) dengan merujuk pada diagram UML yang identik. Kerangka manual ini diberlakukan sebagai kode acuan empiris untuk mengevaluasi sejauh mana LLM mampu mereplikasi keabsahan arsitektur aslinya. Guna menjamin reliabilitas dan konsistensi eksperimen, seluruh tahapan transformasi ini dieksekusi dengan *Standard Operating Procedure* (SOP) parametrikal yang presisi pada setiap domain sistem. Lebih lanjut, untuk memastikan tingkat validitas dan reproduksibilitas eksperimen, penelitian ini menggunakan model arsitektur GPT-4 melalui antarmuka ChatGPT. Generasi kode dikonfigurasi dengan parameter sifat deterministik, di mana nilai *temperature* diasumsikan pada angka 0 guna meminimalisir halusinasi sintaksis dan memastikan konsistensi logika arsitektural dari setiap iterasi keluaran. Seluruh dataset eksperimental, yang meliputi rancangan *Class Diagram*, dokumen *Data Dictionary*, hingga representasi *skeleton code* Pascal hasil komparasi, telah diarsipkan dan dapat diakses secara publik pada repositori berikut: **[MASUKKAN LINK GITHUB / GOOGLE DRIVE DI SINI].**

2.5 Skenario Uji Generasi Kode Berbasis Teks dan

Untuk mengukur keandalan *Large Language Model* (LLM) secara lebih komprehensif, eksperimen pada penelitian ini dibagi menjadi dua skenario masukan (*prompting modality*). Skenario pertama bertumpu pada masukan tekstual terstruktur berupa *Data Dictionary* yang diekstraksi melalui *CASE tools*. Skenario kedua menggunakan masukan visual (*image-to-text*), di mana LLM diberikan secara langsung representasi gambar *Class Diagram* untuk dievaluasi kapabilitas penglihatan komputernya (*computer vision*) dalam mengidentifikasi entitas, atribut, metode, serta arah relasi secara otonom (Wu et al., 2023). Penelitian ini menggunakan tiga rancangan *Class Diagram* dengan tingkat kompleksitas yang berjenjang untuk diujikan pada kedua skenario tersebut.

Rancangan pertama mewakili arsitektur skala rendah, yaitu domain Sistem Akademik. Sistem ini merepresentasikan arsitektur dasar dengan kompleksitas minim yang hanya terdiri dari 4 kelas utama (Mahasiswa, Dosen, KRS, dan Mata_Kuliah). Kelas-kelas tersebut saling terhubung secara terpusat melalui relasi asosiasi linier yang sederhana. Arsitektur visual dari sistem akademik ini dapat dilihat pada Gambar 1.

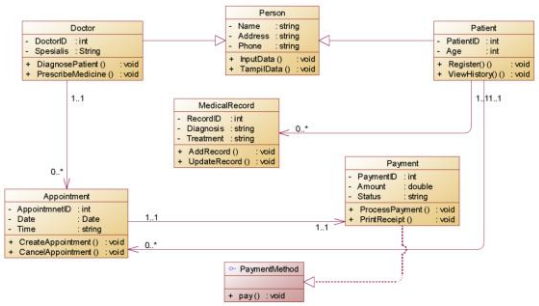


Gambar 1. *Class Diagram* Sistem Akademik

Rancangan kedua mewakili arsitektur skala menengah yang diimplementasikan pada Sistem Rumah Sakit. Model ini memiliki tingkat kerumitan struktural yang lebih padat dengan memuat 7 kelas yang mencakup entitas medis dan administratif. Pada tahapan ini, tingkat kompleksitas visual mulai meningkat dengan hadirnya relasi pewarisan (*inheritance*)—seperti entitas *Doctor* dan *Patient* yang mewarisi atribut dari kelas abstrak *Person* (Fouad & Mohamed, 2018)—serta relasi dependensi

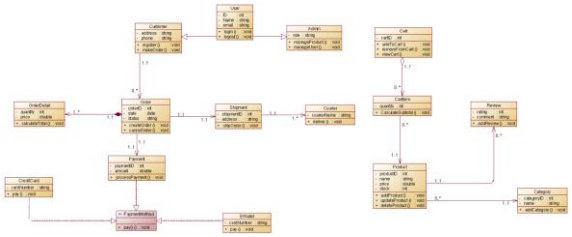
pada proses pembayaran. Rancangan sistem rumah sakit tersebut ditunjukkan pada Gambar 2.

Gambar 2. *Class Diagram* Sistem Rumah Sakit



Gambar 2. *Class Diagram* Sistem Rumah Sakit

Rancangan ketiga, sebagai arsitektur berskala tinggi, menggunakan representasi domain Sistem E-Commerce. Domain ini dipilih sebagai purwarupa skala industri dengan tingkat kompleksitas tertinggi yang rentan memicu kelebihan beban informasi (*information overload*) pada model bahasa generatif (Wu et al., 2023). Model ini memuat 16 kelas dengan hierarki relasi yang saling tumpang tindih secara masif, yang mencakup relasi komposisi (misalnya antara *Order* dan *OrderDetail*), agregasi (*Cart* dan *CartItem*), hingga pewarisan majemuk pada sistem metode pembayaran (Engineering, 2005).Tingkat kerumitan visual sistem e-commerce ini dapat dilihat pada Gambar 3.



Gambar 3. *Class Diagram* Sistem E-Commerce

2.6 Evaluasi Hasil

Fase pengujian berpusat pada penelaahan matematis untuk membandingkan secara komparatif *skeleton code* hasil generasi otomatis dengan *skeleton code* manual pada tingkatan kesamaan deklarasi kelas, konfigurasi atribut, nama metode, serta relasinya. Tingkat irisan kesamaan ini dikalkulasi secara terukur menggunakan metode *Jaccard Similarity*, sebuah algoritma yang menghitung rasio antara jumlah elemen yang saling beririsan dengan total keseluruhan elemen unik yang terbentuk dari dua dokumen kode (Alomari & Harbi, 2019). Rumus perhitungan *Jaccard Similarity* ditunjukkan sebagai berikut:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

dengan A merepresentasikan himpunan elemen arsitektur pada kode manual, sementara B merepresentasikan himpunan sintaks penyusun pada kode hasil generasi ChatGPT. Nilai koefisien kesamaan ini dipetakan pada matriks rentang 0 hingga 1. Semakin mendekati koefisien 1, maka tingkat kesesuaian dan akurasi terjemahan antara program sintesis AI dan kode manual dinilai semakin identik (Song et al., 2024). Hasil kalkulasi evaluasi ini kemudian dikorelasikan untuk memverifikasi secara langsung seberapa besar kompleksitas desain membatasi kualitas performa mesin dalam mengotomatisasi kode dari spesifikasi model.

3. Hasil dan Pembahasan

3.1 Implementasi Transformasi Model ke Kode

Nilai koefisien kesamaan ini dipetakan pada matriks rentang 0 hingga 1. Semakin mendekati koefisien 1, maka tingkat kesesuaian dan akurasi terjemahan antara program sintesis Al dan kode manual dinilai semakin identik (Song et al., 2024). Representasi kamus data ini kemudian digunakan sebagai instruksi definitif bagi kecerdasan buatan untuk menghasilkan skeleton code berbahasa Pascal.

Hasil observasi awal mengonfirmasi kemampuan Large Language Model dalam menjembatani kebutuhan desain ke dalam sintaks kode yang terstruktur (Hou et al., 2024). Model dengan skala arsitektur rendah terbukti mampu mempertahankan konsistensi terjemahan secara utuh, sementara pada sistem berskala menengah hingga tinggi, akurasi pemetaan elemen model mulai mengalami berbagai distorsi struktural (Kanuka et al., 2023).

3.2 Perbandingan Hasil Generasi ChatGPT dan Kode Manual

Untuk mengidentifikasi tingkat presisi operasional kecerdasan buatan, seluruh elemen arsitektur pada skeleton code hasil transformasi diurai dan dikomparasikan dengan kerangka kode manual yang diposisikan sebagai acuan empiris.

Tabel 1. Dekomposisi Elemen Kode Transformasi Manual

Domain Sistem	Class Sesuai	Atribut Sesuai	Metode Sesusai	Relasi Sesuai	Total Elemen
Akademik	4	10	9	3	26
Rumah Sakit	7	16	14	4	41
E-Commerce	16	22	22	11	71

Tabel 2. Dekomposisi Elemen Kode Transformasi Otomatis Berbasis Dokumen (ChatGPT)

Domain Sistem	Class Sesuai	Atribut Sesuai	Metode Sesuai	Relasi Sesuai	Total Sesuai
Akademik	4	10	8	3	25
Rumah Sakit	7	16	15	6	43
E-Commerce	16	29	24	11	80

Tabel 3. Dekomposisi Elemen Kode Transformasi Otomatis Berbasis Gambar (ChatGPT)

Domain Sistem	Class Sesuai	Atribut Sesuai	Metode Sesusai	Relasi Sesuai	Total Elemen
Akademik	4	10	9	3	26
Rumah Sakit	7	19	13	6	44
E-Commerce	15	31	24	8	78

Perbandingan metrik antar tabel tersebut memperlihatkan dinamika akurasi yang berfluktuasi berdasarkan skala sistem.

Pada sistem akademik, kecerdasan buatan mencapai presisi mutlak untuk masukan berbasis gambar (26 elemen), sedangkan masukan berbasis dokumen mengalami kekurangan 1 elemen metode. Hal ini menunjukkan bahwa untuk sistem sederhana, masukan visual lebih akurat dibandingkan masukan tekstual.

Pada sistem rumah sakit (skala menengah), terjadi fenomena kekurangan elemen (*under-generation*) pada masukan gambar dan kelebihan relasi pada masukan dokumen. Kekurangan ini disebabkan oleh hilangnya beberapa definisi relasi asosiasi pada saat transformasi. Hal tersebut terjadi karena kecerdasan buatan mengalami ambiguitas semantik saat memproses keterikatan antara entitas medis, sehingga model mengambil asumsi implisit dengan menghilangkan satu relasi yang dianggap redundan secara logika program meskipun relasi tersebut diwajibkan oleh desain aslinya (Wang et al., 2024)**.**

Pada sistem e-commerce (skala tinggi), deviasi yang sangat ekstrem terdeteksi dalam bentuk kelebihan elemen (*over-generation*) pada kedua mode masukan. Pada masukan dokumen, AI menghasilkan 80 elemen (9 elemen berlebih), sedangkan pada masukan gambar menghasilkan 78 elemen (7 elemen berlebih). Kelebihan ini ditandai dengan penambahan atribut ekstra dan metode baru yang sama sekali tidak terdefinisi di dalam instruksi Data Dictionary.

Pembengkakan ini terjadi akibat fenomena halusinasi data atau kejenuhan informasi (*information overload*) (Hou et al., 2024). Karena model bahasa dilatih dengan miliaran baris kode dari repositori global, ketika dihadapkan pada konteks e-commerce yang kompleks, kecerdasan buatan secara otonom merespons dengan memprediksi dan menambahkan fungsionalitas lazim seperti fitur gerbang pembayaran, *timestamp*, atau rincian keranjang belanja yang diasumsikan wajib ada, alih-alih mematuhi instruksi konversi leksikal secara kaku dari kamus data yang diberikan (Wang et al., 2024).

3.3 Hasil Evaluasi Berdasarkan Tingkat Kompleksitas

Pengukuran korelasi antara kerumitan desain dan fluktuasi kualitas transformasi dihitung secara matematis menggunakan metrik Weighted Methods per Class (WMC) dan rasio Jaccard Similarity (Song et al., 2024)**.**

Tabel 4. Hasil Evaluasi Kesamaan Kode terhadap Metrik Kompleksitas (Berbasis Dokumen)

Sistem	Total WMC	Rata-Rata WMC	Jumlah Kelas	Jumlah Relasi	Jaccard Similaarity
Akademik	8	2.00	4	3	1.000
Rumah Sakit	15	2.14	7	6	0.935
E-Commerce	24	1.50	16	11	0.845

Tabel 5. Hasil Evaluasi Kesamaan Kode terhadap Metrik Kompleksitas (Berbasis Gambar)

Sistem	Total WMC	Rata-Rata WMC	Jumlah Kelas	Jumlah Relasi	Jaccard Similaarity
Akademik	8	2.00	4	3	1.000
Rumah Sakit	15	2.14	7	6	0.932
E-Commerce	24	1.50	16	11	0.910

Distribusi data pada Tabel 3.5 dan 3.6 mengonfirmasi dampak langsung dari anomali elemen yang telah dijabarkan sebelumnya. Akurasi struktural mencapai nilai 1,0 atau seratus persen pada domain berskala rendah, yaitu Sistem Akademik, karena kesederhanaan objek yang meminimalisir intervensi halusinasi kecerdasan buatan. Pada sistem ini, baik masukan dokumen maupun gambar menghasilkan Jaccard Similarity sempurna.

Submitted : 08-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

Memasuki skala menengah yang direpresentasikan oleh Sistem Rumah Sakit, terjadi penurunan nilai kesamaan menjadi sekitar 0,93 baik untuk masukan dokumen (0,935) maupun masukan gambar (0,932). Penurunan ini terutama disebabkan oleh hilangnya satu elemen relasi pada masukan gambar dan penambahan satu relasi berlebih pada masukan dokumen, bukan karena kesalahan pada jumlah metode. Dengan kata lain, pada tingkat kompleksitas metode yang masih moderat (total WMC = 15 dengan 7 kelas), kesalahan transformasi lebih banyak terjadi pada pemetaan relasi antar kelas dibandingkan pada deklarasi metode itu sendiri.

Pada skala tinggi, yaitu Sistem E-Commerce dengan total WMC sebesar 24 dan jumlah kelas mencapai 16, tingkat validitas mencapai titik terendah. Untuk masukan dokumen, Jaccard Similarity turun menjadi 0,845, sedangkan untuk masukan gambar masih berada di 0,910. Penurunan ini merupakan penalti matematis dari besarnya volume atribut dan metode fabrikasi yang disisipkan oleh model kecerdasan buatan, sebuah fenomena yang dikenal sebagai *over-generation*. Menariknya, meskipun total WMC E-Commerce adalah yang tertinggi, nilai rata-rata WMC per kelas justru paling rendah (1,50) dibandingkan Sistem Rumah Sakit (2,14) dan Akademik (2,00). Anomali ini terjadi karena E-Commerce memiliki banyak kelas (16 kelas) dengan distribusi metode yang tidak merata; sebagian besar kelas hanya memiliki satu atau dua metode, namun beberapa kelas seperti Order dan Payment memiliki tiga hingga empat metode. Akibatnya, total WMC tetap tinggi karena akumulasi, tetapi rata-rata per kelas menjadi kecil. Temuan ini mengindikasikan bahwa rata-rata WMC bukanlah indikator yang baik untuk memprediksi akurasi AI pada sistem masif; yang lebih berpengaruh adalah total WMC absolut dan jumlah relasi.

Lebih lanjut, peran jumlah relasi terhadap penurunan akurasi terlihat sangat jelas. Sistem Akademik dengan hanya tiga relasi mencapai akurasi seratus persen. Sistem Rumah Sakit dengan enam relasi mengalami penurunan akurasi sekitar enam hingga tujuh persen. Adapun Sistem E-Commerce dengan sebelas relasi mengalami penurunan akurasi hingga 15,5 persen pada masukan dokumen. Semakin banyak relasi, semakin tinggi kemungkinan AI salah menafsirkan atau menghilangkan relasi, terutama pada masukan dokumen. Sebaliknya, masukan gambar terbukti lebih tahan terhadap kompleksitas tinggi karena AI dapat menangkap koneksi visual antar kelas secara langsung; pada E-Commerce, masukan gambar menghasilkan akurasi 6,5 persen lebih tinggi dibandingkan masukan dokumen.

Berdasarkan temuan tersebut, dapat disimpulkan bahwa akurasi sempurna hanya tercapai pada sistem dengan total WMC tidak lebih dari delapan dan jumlah relasi maksimal tiga. Setiap penambahan satu relasi di atas angka tiga menurunkan akurasi sekitar satu hingga dua persen pada mode dokumen dan sekitar 0,5 hingga satu persen pada mode gambar. Mode gambar secara konsisten lebih *robust* terhadap peningkatan kompleksitas, dengan akurasi tetap di atas 91 persen meskipun total WMC mencapai 24. Dengan demikian, validitas WMC sebagai indikator prediktif kegagalan transformasi dapat dibenarkan secara empiris, dengan ambang batas kritis berada di rentang total WMC antara 15 hingga 20. Untuk sistem dengan total WMC di atas 20, sangat direkomendasikan untuk menggunakan masukan gambar dan melakukan verifikasi manual pascagenerasi guna menjamin integritas struktural kode.

3.4 Analisis Pengaruh Kompleksitas terhadap Kualitas Kode

Integrasi hasil eksperimen membuktikan secara empiris adanya korelasi negatif yang bersifat linear antara parameter arsitektur sistem, khususnya pada akumulasi kuantitas kelas dan nilai Weighted Methods per Class (WMC), terhadap kualitas skeleton code yang dihasilkan oleh Large Language Model (H.M & Nandakumar, 2016).

3.4.1 Mekanisme Degradasi Akurasi

Pada kerangka model berskala rendah hingga menengah (Sistem Akademik dan Rumah Sakit), jendela konteks (*context window*) operasional dari model kecerdasan buatan generatif masih memiliki kapasitas yang mumpuni (Wang et al., 2024). Dalam rentang kompleksitas tersebut, model AI terbukti mampu memproses, mengingat, dan menduplikasi abstraksi hierarki kelas beserta seluruh relasi asosiasinya secara koheren tanpa merusak integritas logika sistem (Kanuka et al., 2023).

Namun, transisi menuju pemodelan berdimensi masif dan kompleks (Sistem E-Commerce) membuktikan bahwa konsistensi struktural kecerdasan buatan sangat rentan mengalami degradasi parah (Hou et al., 2024). Ketika algoritma dihadapkan pada nilai WMC yang melambung tinggi beserta puluhan percabangan relasi yang saling tumpang tindih, mesin akan mengalami fenomena kejenuhan kognitif atau *information overload* (Wang et al., 2024).

3.4.2 Fenomena Halusinasi Struktural

Melebarnya kesenjangan Jaccard Similarity pada domain kompleks ini menjadi bukti konkret bahwa mesin generatif belum sepenuhnya dapat melepaskan diri dari basis pelatihan bahasa alaminya (Alfan Rosid, 2020). Alih-alih mengeksekusi instruksi transformasi secara kaku dan presisi berdasarkan Data Dictionary yang diberikan, model LLM justru cenderung mengambil alih logika desain dengan memprediksi fungsionalitas tambahan (Hou et al., 2024).

Manifestasi dari halusinasi data ini terlihat dari kecenderungan AI yang secara otonom menyisipkan atribut atau metode baru—seperti fungsi keranjang belanja atau gerbang pembayaran—yang diasumsikan "wajib ada" pada sistem e-commerce berdasarkan miliaran parameter data pelatihannya, sehingga memicu kelebihan elemen yang merusak validitas struktur asli (Wang et al., 2024).

3.4.3 Validasi WMC sebagai Indikator Prediktif

Peningkatan nilai metrik WMC dengan demikian tervalidasi secara ilmiah sebagai indikator prediktif yang sangat sensitif dan akurat untuk memetakan probabilitas kegagalan maupun tingkat kesulitan transformasi pada rekayasa perangkat lunak yang diotomatisasi. Semakin tinggi skor kerumitan suatu objek, semakin besar pula tingkat halusinasi struktural dan penyimpangan sintaksis yang akan dihasilkan oleh model generatif.

Rekomendasi ambang batas WMC: Berdasarkan temuan ini, penggunaan LLM dalam siklus Model-Based Software Engineering direkomendasikan untuk sistem dengan nilai WMC total ≤ 15 (atau rata-rata WMC per kelas ≤ 2,1). Pada nilai di atas ambang tersebut, risiko over-generation dan under-generation meningkat secara signifikan.

3.5 Implikasi Riset dan Ancaman Validitas

3.5.1 Implikasi Praktis

Dalam lingkup Rekayasa Perangkat Lunak Berbasis Model, temuan anomali kelebihan dan kekurangan elemen kode ini memberikan implikasi praktis bahwa instrumen generasi otomatis belum sepenuhnya siap beroperasi secara otonom pada arsitektur berskala industri tanpa tahap verifikasi berlapis (Lauder et al., 2010).

Untuk mempertahankan konsistensi keterlacakan arsitektur (Kanuka et al., 2023), direkomendasikan pendekatan perancangan desain berbasis layanan mikro (*microservices*), di mana sebuah struktur yang memiliki nilai WMC kumulatif sangat tinggi harus dipecah terlebih dahulu menjadi beberapa subsistem independen yang lebih kecil sebelum ditransformasikan oleh AI.

3.5.2 Ancaman Validitas

Ancaman terhadap validitas riset ini bertumpu pada sifat deterministik model kecerdasan buatan yang sangat dinamis. Pembaruan algoritma pelatihan di masa mendatang berpotensi mengubah pola halusinasi data sehingga luaran sintaks bisa sedikit bervariasi pada pengujian ulang (Hou et al., 2024).

Namun, implementasi Data Dictionary yang diisolasi secara objektif telah berhasil meminimalkan ambiguitas linguistik, sehingga memastikan metodologi ini tetap memiliki tingkat replikasi ilmiah yang tinggi (Alfan Rosid, 2020).

4. Kesimpulan

Penelitian ini menyimpulkan bahwa tingkat kompleksitas arsitektur pada *Class Diagram* memiliki korelasi negatif yang signifikan terhadap akurasi *skeleton code* yang dihasilkan secara otomatis oleh *Large Language Model* (LLM). Melalui pendekatan *Model-Based Software Engineering* (MBSE), kecerdasan buatan terbukti mampu mempertahankan presisi struktural yang sempurna pada model berskala rendah, sebagaimana dibuktikan oleh perolehan koefisien *Jaccard Similarity* sebesar 1.0 pada sistem akademik. Namun, seiring dengan eskalasi skala kerumitan yang direpresentasikan oleh lonjakan metrik *Weighted Methods per Class* (WMC), kepadatan relasi, dan populasi kelas, keandalan transformasi model mengalami degradasi yang nyata. Hal ini terbukti dari penurunan nilai akurasi pada sistem rumah sakit (0.935) hingga mencapai titik terendah pada sistem e-commerce (0.845). Pada tingkat kerumitan tinggi, keterbatasan jendela konteks pada ChatGPT memicu terjadinya fenomena kejenuhan informasi yang berujung pada halusinasi data, baik berupa penghilangan elemen esensial (*under-generation*) maupun fabrikasi atribut dan metode fiktif (*over-generation*) yang menyimpang dari instruksi *Data Dictionary*.

Temuan ini secara empiris memvalidasi bahwa metrik WMC sangat krusial untuk difungsikan sebagai indikator prediktif dalam menakar batas toleransi keandalan kecerdasan buatan sebelum mengeksekusi otomatisasi rekayasa perangkat lunak. Penggunaan LLM terbukti sangat efisien untuk mempercepat fase *coding* awal pada sistem berskala kecil hingga menengah, namun model ini belum sepenuhnya siap beroperasi secara otonom pada rancangan arsitektur tingkat industri tanpa adanya verifikasi manual yang ketat. Sebagai implikasi praktis, pengembang disarankan untuk memecah arsitektur sistem yang kompleks menjadi modul-modul layanan independen yang lebih kecil (*microservices*) sebelum ditransformasikan oleh mesin. Penelitian di masa mendatang direkomendasikan untuk mengeksplorasi optimasi *prompt engineering* secara spesifik atau menerapkan teknik prapelatihan lanjutan (*fine-tuning*) pada model bahasa guna menekan rasio halusinasi struktural saat menangani transformasi spesifikasi sistem berskala masif.

Daftar Rujukan

[1] I. A. Niaz, "Automatic Code Generation From UML Class and Statechart Diagrams," Ph.D. dissertation, Univ. of Tsukuba, Tsukuba, Japan, 2005.
[2] M. Lauder, M. Schlereth, S. Rose, and A. Schürr, "Model-driven systems engineering: state-of-the-art and research challenges," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 58, no. 3, pp. 389–405, 2010.

Submitted : 08-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

[3] X. Hou *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *arXiv preprint arXiv:2308.10620*, 2023.
[4] H. Kanuka, G. Koreki, R. Soga, and K. Nishikawa, "Exploring the ChatGPT Approach for Bidirectional Traceability Problem between Design Models and Code," *arXiv preprint arXiv:2309.14992*, 2023.
[5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.
[6] T. Pradeep, A. K. Gonepally, A. K. Pusala, and C. T. Singarapu, "Anti-plagiarism tool helping developers to generate authentic code," *International Journal of Science and Research Archive*, vol. 14, no. 1, pp. 1208–1215, 2025.
[7] T. Fouad and B. Mohamed, "Extracting UML Models and OCL Integrity Constraints From Object Relational Database," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 4, pp. 1138–1149, 2018.
[8] C. Wang, B. Wang, P. Liang, and J. Liang, "Assessing UML Models by ChatGPT: Implications for Education," *arXiv preprint arXiv:2412.17200*, 2024.
[9] H. M. Manoj and A. N. Nandakumar, "Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design," *APTIKOM Journal on Computer Science and Information Technologies*, vol. 1, no. 2, pp. 63–76, 2016.
[10] Y. Song, C. Lothritz, D. Tang, T. F. Bissyandé, and J. Klein, "Revisiting Code Similarity Evaluation with Abstract Syntax Tree Edit Distance," *arXiv preprint*, 2024.
[11] F. Alomari and M. Harbi, "Scalable Source Code Similarity Detection in Large Code Repositories," *EAI Endorsed Transactions on Scalable Information Systems*, 2020.
[12] C. N. Paradis, M. R. Yusuf, M. Farhanudin, and M. A. Yaqin, "Analisis dan Perancangan Software Pengukuran Metrik Skala dan Kompleksitas Diagram Class," *JACIS: Journal Automation Computer Information System*, vol. 2, no. 1, pp. 58–65, 2022.
[13] I. Keivanloo, C. K. Roy, and J. Rilling, "Towards Source Code Clone Search via Information Retrieval," Univ. of Saskatchewan, Saskatchewan, Canada, Tech. Rep. #2014-02, 2014.
[14] P. Kalyanasundaram and S. P. Ugale, "Model Transformation: Concept, Current Trends and Challenges," *International Journal of Computer Applications*, vol. 119, no. 14, pp. 33–37, 2015.
[15] Z. Zheng *et al.*, "Translating Regulatory Clauses into Executable Codes for Building Design Checking via Large Language Model Driven Function Matching and Composing," *Preprints*, 2025.
[16] N. K. Maina, G. M. Muketha, and G. M. Wambugu, "A Literature Survey of Complexity Metrics for Object-Oriented Programs," *International Journal of Science and Engineering Applications*, vol. 11, no. 5, pp. 66–71, 2022.
[17] M. Monteiro, B. C. Branco, S. Silvestre, G. Avelino, and M. T. Valente, "NoCodeGPT: A No-Code Interface for Building Web Apps with Language Models," *arXiv preprint arXiv:2310.14843v2*, 2024.
[18] N. Nguyen and S. Nadi, "An Empirical Evaluation of GitHub Copilot's Code Generation Abilities," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022.
[19] C. Wu *et al.*, "Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models," *arXiv preprint arXiv:2303.04671*, 2023.
[20] M. Ozkaya, "Are the UML modelling tools powerful enough for practitioners? A literature review," *IET Software*, 2018.