

TESIS

**PENGEMBANGAN MODEL COMPILER BERBASIS DOMAIN-
SPECIFIC LANGUAGE YANG MENGADOPSI KONSEP XTUML
UNTUK TRANSFORMASI ARTEFAK BACKEND LARAVEL
MENGUNAKAN ANTLR4**



Oleh:

Wahyu Andi Rejeki

24206051007

**PROGRAM STUDI INFORMATIKA
PROGRAM MAGISTER FAKULTAS SAINS DAN TEKNOLOGI
UIN SUNAN KALIJAGA
YOGYAKARTA**

2026

PENGESAHAN TUGAS AKHIR



KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1990/Un.02/DST/PP.00.9/08/2026

Tugas Akhir dengan judul : PENGEMBANGAN MODEL COMPILER BERBASIS DOMAIN SPECIFIC LANGUAGE YANG MENGADOPSI KONSEP XTUML UNTUK TRANSFORMASI ARTEFAK BACKEND LARAVEL MENGGUNAKAN ANTLR4

yang dipersiapkan dan disusun oleh:

Nama : WAHYU ANDI REJEKI, S.Kom
Nomor Induk Mahasiswa : 24206051007
Telah diujikan pada : Kamis, 13 Agustus 2026
Nilai ujian Tugas Akhir : A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR



Valid ID: 6a87f3a3e43d

Ketua Sidang
Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED



Valid ID: 6a87d015768f

Penguji I
Muhammad Mustakim, S.T. M.T.
SIGNED



Valid ID: 6a87e5d599b18

Penguji II
Ir. Muhammad Taufiq Nuruzzaman, S.T.
M.Eng., Ph.D.
SIGNED



Valid ID: 6a87fe485c46

Yogyakarta, 13 Agustus 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi
Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

SURAT PERNYATAAN KEASLIAN

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Wahyu Andi Rejeki
NIM : 24206051007
Jenjang : Magister
Program Studi : Informatika

menyatakan bahwa tesis ini secara keseluruhan adalah hasil penelitian/karya sendiri, kecuali pada bagian-bagian tertentu yang penyusun ambil sebagai acuan.

Apabila terbukti pernyataan ini tidak benar, maka sepenuhnya menjadi tanggung jawab penyusun.

Yogyakarta, 11 Agustus 2026

Saya yang menyatakan,



Wahyu Andi Rejeki
24206051007

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

SURAT PERNYATAAN BEBAS PLAGIASI

SURAT PERNYATAAN BEBAS PLAGIASI

Yang bertanda tangan di bawah ini:

Nama : Wahyu Andi Rejeki
NIM : 24206051007
Jenjang : Magister
Program Studi : Informatika

menyatakan bahwa naskah tesis ini secara keseluruhan benar-benar bebas dari plagiasi. Jika di kemudian hari terbukti melakukan plagiasi, maka saya siap ditindak sesuai ketentuan hukum yang berlaku.

Yogyakarta, 11 Agustus 2026

Saya yang menyatakan,



Wahyu Andi Rejeki
24206051007

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

NOTA DINAS PEMBIMBING

NOTA DINAS PEMBIMBING

Kepada,
Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga
Yogyakarta

Assalamu 'alaikum wr. wb.

Disampaikan dengan hormat, setelah melakukan bimbingan, arahan, dan koreksi terhadap penulisan tesis yang berjudul:

**PENGEMBANGAN MODEL COMPILER BERBASIS DOMAIN SPECIFIC-
LANGUAGE YANG MENGADOPSI KONSEP XTUML UNTUK
TRANSFORMASI ARTEFAK BACKEND LARAVEL MENGGUNAKAN
ANTLR4**

Yang ditulis oleh:

Nama : Wahyu Andi Rejeki
NIM : 24206051007
Fakultas : Sains dan Teknologi
Jenjang : Program Studi Magister
Program Studi : Informatika

Saya berpendapat bahwa tesis tersebut sudah dapat diajukan kepada Program Studi Magister (S2) Informatika UIN Sunan Kalijaga untuk diujikan dalam rangka memperoleh gelar Magister Komputer.

Wassalamu'alaikum wr. wb.

Yogyakarta, 11 Agustus 2026
Pembimbing,

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA


Dr. Agung Fatwanto, S.Si., M.Kom.
19770103 200501 003

ABSTRAK

Fase konstruksi kode dalam *Software Development Life Cycle* (SDLC) merupakan salah satu tahapan yang paling banyak menyerap sumber daya waktu dan biaya. Pendekatan *Model-Driven Engineering* (MDE) menawarkan mekanisme untuk mengotomatisasi transformasi model menjadi artefak implementasi. Salah satu konsep yang menginspirasi penelitian ini adalah *Executable and Translatable UML* (xtUML), khususnya dalam merepresentasikan aspek struktural dan perilaku sistem secara terpisah dari detail implementasi. Namun, penerapan pendekatan *model-driven* pada pengembangan aplikasi web masih menghadapi tantangan dalam mentransformasikan spesifikasi perilaku sistem ke dalam artefak backend secara otomatis dan konsisten, sehingga logika perilaku (*action semantics*) masih sering memerlukan implementasi manual.

Penelitian ini mengembangkan *Model Compiler* berbasis PHP menggunakan ANTLR4 untuk mentransformasikan spesifikasi *Domain-Specific Language* (DSL) yang mengadopsi prinsip *Object Action Language* (OAL) menjadi artefak backend Laravel berbasis arsitektur MVC. Penelitian mencakup operasi CRUD relasional, validasi data, *middleware*, dan *routing*. Artefak *backend* yang dihasilkan diuji menggunakan *black-box testing* melalui eksekusi *endpoint* untuk mengevaluasi kesesuaiannya dengan spesifikasi DSL. Hasil penelitian menunjukkan bahwa *model compiler* mampu menghasilkan artefak *backend* Laravel yang berfungsi sesuai spesifikasi dan memenuhi skenario pengujian CRUD relasional.

Kata Kunci: Compiler, xtUML, Action Semantics, ANTLR4, PHP, Laravel, Visitor Pattern.

ABSTRACT

The code construction phase in the Software Development Life Cycle (SDLC) is one of the stages that requires substantial time and cost. Model-Driven Engineering (MDE) offers an approach for automating the transformation of models into implementation artifacts. This research is inspired by the concept of Executable and Translatable UML (xtUML), particularly its representation of the structural and behavioral aspects of a system separately from implementation details. However, automatically transforming behavioral specifications into backend artifacts remains a challenge in web application development.

This research develops a PHP-based Model Compiler using ANTLR4 to transform a Domain-Specific Language (DSL) specification that adopts principles from Object Action Language (OAL) into MVC-based Laravel backend artifacts. The research covers relational CRUD operations, data validation, middleware, and routing. The generated backend artifacts were tested using black-box testing through endpoint execution to evaluate their conformance to the DSL specifications. The results show that the compiler can generate functional Laravel backend artifacts that conform to the specifications and satisfy the defined relational CRUD test scenarios.

Keywords: Compiler, xtUML, Action Semantics, ANTLR4, PHP, Laravel, Visitor Pattern.

MOTTO

"Penderitaan adalah jalan untuk mencapai kedamaian"

Pain Akatsuki



PERSEMBAHAN

Dengan penuh rasa syukur kepada Allah SWT atas segala rahmat, nikmat, dan kekuatan yang telah diberikan, karya tesis ini saya persembahkan kepada:

1. Kepada istri saya tercinta, Dewi Lestari yang senantiasa memberikan dukungan, pengertian, doa dan semangat di tengah segala proses dan tantangan yang saya hadapi selama menyelesaikan pendidikan ini.
2. Almarhum anak saya Muhammad.
3. Almarhum anak saya Ahmad.
4. Anak saya Elina Ghina Pratista, yang menjadi salah satu alasan terbesar untuk terus berjuang dan menyelesaikan setiap proses dengan sebaik-baiknya. Semoga kelak engkau memahami bahwa setiap langkah dalam perjalanan ini juga merupakan bagian dari usaha untuk memberikan teladan bahwa pendidikan dan perjuangan tidak pernah sia-sia.
5. Kepada kedua orang tua yang senantiasa menjadi sumber doa, kasih sayang dan kekuatan.
6. Kepada dosen pembimbing, dosen penguji, serta seluruh pihak akademik yang telah memberikan ilmu, arahan, dan bimbingan selama proses penyusunan tesis ini.
7. Kepada rekan-rekan dan sahabat yang selalu memberikan bantuan, motivasi, serta kebersamaan dalam menyelesaikan studi ini.
8. Terakhir, karya ini saya persembahkan untuk diri saya sendiri, yang telah berusaha bertahan, belajar, dan menyelesaikan setiap proses dengan penuh kesabaran dan tanggung jawab.

Tesis ini bukan hanya tentang sebuah karya ilmiah, tetapi juga tentang doa, perjuangan, kesabaran dan proses panjang yang mengajarkan arti ketekunan.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Swt. atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tesis yang berjudul “Pengembangan Model Compiler Berbasis Domain-Specific Language Yang Mengadopsi Konsep xtUML Untuk Transformasi Artefak Backend Laravel Menggunakan Antlr4” dengan baik. Tesis ini disusun sebagai salah satu syarat untuk memperoleh gelar Magister pada program Studi Magister Informatika, UIN Sunan Kalijaga Yogyakarta.

Penulis menyadari bahwa penyusunan tesis ini tidak lepas dari bantuan, bimbingan, dukungan, serta doa dari berbagai pihak. Oleh karena itu, dengan segala kerendahan hati, penulis menyampaikan terima kasih kepada:

1. Bapak Dr. Agung Fatwanto, S.Si., M.Kom., selaku dosen pembimbing yang telah memberikan arahan, bimbingan, masukan, dan motivasi selama proses penyusunan tesis ini.
2. Bapak Muhammad Mustakim, S.T., M.T., selaku Penguji 1 yang telah memberikan masukan, kritik, dan saran yang membangun demi penyempurnaan penelitian ini.
3. Bapak Ir. Muhammad Taufiq Nuruzzaman, S.T., M.Eng., Ph.D., selaku Penguji II yang telah memberikan masukan, kritik, dan saran yang membangun demi penyempurnaan penelitian ini.
4. Seluruh dosen Program Studi Magister Informatika UIN Sunan Kalijaga yang telah memberikan ilmu, wawasan, dan pengalaman akademik selama masa perkuliahan.
5. Rekan-rekan mahasiswa magister informatika serta semua pihak yang telah membantu, baik secara langsung maupun tidak langsung, dalam proses penyelesaian tesis ini.

Penulis menyadari bahwa tesis ini masih memiliki keterbatasan dan belum

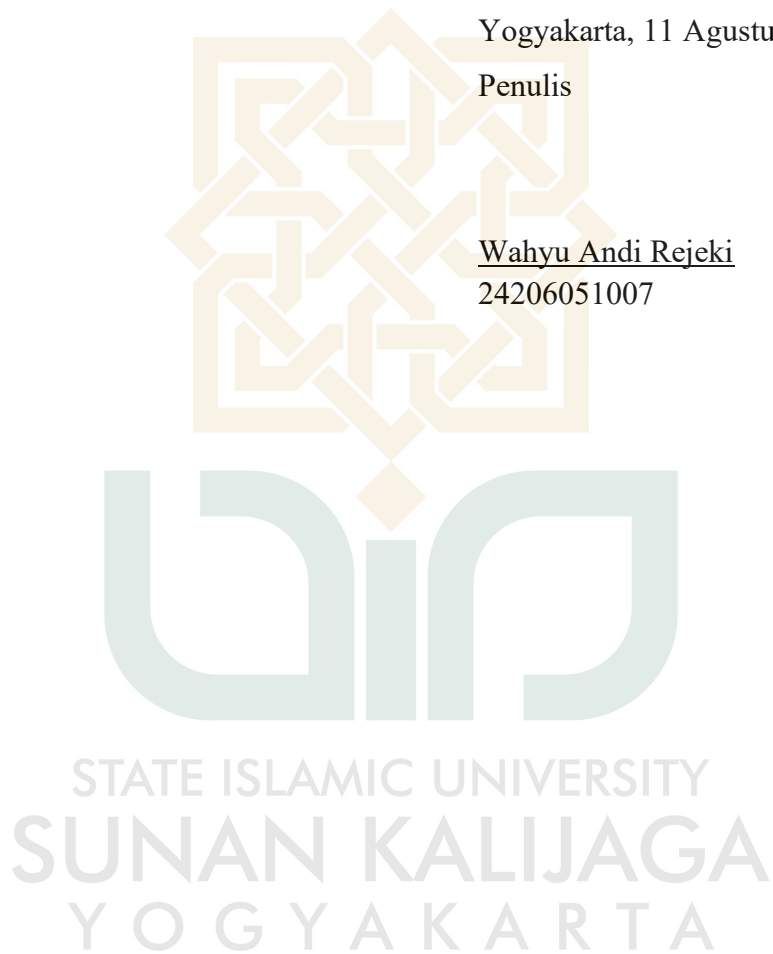
sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun demi perbaikan dan pengembangan penelitian di masa mendatang.

Yogyakarta, 11 Agustus 2026

Penulis

Wahyu Andi Rejeki

24206051007



DAFTAR ISI

TESIS	i
PENGESAHAN TUGAS AKHIR	ii
SURAT PERNYATAAN KEASLIAN	iii
SURAT PERNYATAAN BEBAS PLAGIASI	iv
NOTA DINAS PEMBIMBING	v
ABSTRAK	vi
MOTTO	viii
PERSEMBAHAN	ix
KATA PENGANTAR	x
DAFTAR ISI	xii
DAFTAR TABEL	xiv
DAFTAR GAMBAR	xv
DAFTAR LAMPIRAN	xvi
DAFTAR SINGKATAN	xvii
BAB I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	5
C. Tujuan Penelitian	5
D. Manfaat Penelitian	6
BAB II KAJIAN PUSTAKA DAN LANDASAN TEORI	7
A. Kajian Pustaka	7
B. Landasan Teori	12
1. Model-Driven Engineering (MDE).....	12
2. Executable and Translatable UML (xtUML).....	29
3. Domain-Specific Language (DSL)	45
4. Konsep Compiler	61

5.	ANTLR4	77
6.	Visitor Pattern	85
7.	Semantic Mapping	93
8.	Laravel MVC	99
9.	Design Science Research	101
10.	Kerangka Konseptual.....	104
BAB III METODE PENELITIAN.....		108
A.	Lingkungan Pengembangan (<i>Development Environment</i>)	113
B.	Arsitektur Modular Model <i>Compiler</i>	114
C.	Spesifikasi <i>Grammar DSL</i>	118
D.	Mekanisme Pemetaan Semantik (<i>Semantic Mapping</i>) ...	122
BAB IV HASIL DAN PEMBAHASAN		124
A.	Implementasi Komponen <i>Front-End Compiler</i>	124
B.	Implementasi Komponen <i>Back-End Compiler</i>	130
C.	Demonstrasi Proses Transpilasi	134
D.	Evaluasi Artefak Model <i>Compiler</i>	138
BAB V PENUTUP		149
A.	Kesimpulan.....	149
B.	Saran	149
DAFTAR PUSTAKA		150
LAMPIRAN		154

DAFTAR TABEL

Tabel 2.1 Perbandingan dengan Penelitian Terdahulu	10
Tabel 2.2 Matriks Perbandingan Karakteristik Penelitian	12
Tabel 3.1 Spesifikasi Perangkat Keras	113
Tabel 3.2 Spesifikasi Perangkat Lunak	113
Tabel 3.3 Cakupan grammar yang dikembangkan	121
Tabel 4.1 Daftar Tokenisasi	128
Tabel 4.2 Demonstrasi Transformasi Semantik DSL ke Artefak Laravel	135
Tabel 4.3 Skenario <i>Black-box</i> Testing	140
Tabel 4.4 Hasil <i>Black-box</i> Testing	143



DAFTAR GAMBAR

Gambar 2.1 alur kerangka konseptual	107
Gambar 3.1 Tahapan penelitian berdasarkan <i>Design Science Research Methodology</i> (DSRM)	109
Gambar 3.2 Alur pendekatan RAD	110
Gambar 3.3 Arsitektur modular <i>compiler</i> yang dikembangkan pada penelitian ini.....	114
Gambar 3.4 Alur proses kompilasi	118
Gambar 3.5 Potongan Rancangan Grammar DSL (OAL.g4).....	120
Gambar 4.1 Alur Implementasi <i>Front-End Compiler</i>	125
Gambar 4.2 Potongan Grammar OAL.g4.....	126
Gambar 4.3 Potongan Script library.oal	127
Gambar 4.4 Arsitektur Implementasi <i>Back-End Compiler</i>	131
Gambar 4.5 Hasil pengujian endpoint GET /books.....	144
Gambar 4.6 Hasil pengujian endpoint POST /books.....	144
Gambar 4.7 Hasil pengujian endpoint POST /books/borrow	145
Gambar 4.8 Hasil pengujian <i>middleware</i>	145

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR LAMPIRAN

Lampiran 1 Script OAL.g4.....	154
Lampiran 2 Script BaseCompiler.php	155
Lampiran 3 Script LaravelControllerCompiler.php	156
Lampiran 4 Script LaravelModelCompiler.php	157
Lampiran 5 Script LaravelMiddleWareCompiler.php.....	159
Lampiran 6 Script LaravelRouteCompiler.php	160
Lampiran 7 Script generate.php.....	162
Lampiran 8 Script library.oal.....	163



DAFTAR SINGKATAN

ANTLR	: <i>ANother Tool for Language Recognition</i>
API	: <i>Application Programming Interface</i>
AST	: <i>Abstract Syntax Tree</i>
CRUD	: <i>Create, Read, Update, Delete</i>
DSL	: <i>Domain-Specific Language</i>
MDE	: <i>Model-Driven Engineering</i>
MVC	: <i>Model-View-Controller</i>
OAL	: <i>Object Action Language</i>
ORM	: <i>Object-Relational Mapping</i>
PIM	: <i>Platform-Independent Model</i>
PSM	: <i>Platform-Specific Model</i>
RAD	: <i>Rapid Application Development</i>
SDLC	: <i>Software Development Life Cycle</i>
UML	: <i>Unified Modeling Language</i>
xtUML	: <i>Executable and Translatable UML</i>

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Perkembangan sistem informasi modern mendorong peningkatan kompleksitas dalam proses pengembangan perangkat lunak. Kompleksitas tersebut tidak hanya dipengaruhi oleh bertambahnya kebutuhan fungsional sistem, tetapi juga oleh semakin banyaknya komponen perangkat lunak yang harus saling berintegrasi. Kondisi ini menyebabkan proses implementasi perangkat lunak tetap sepenuhnya bergantung pada penulisan kode secara manual (*manual coding*), sehingga meningkatkan waktu pengembangan, biaya implementasi, serta risiko terjadinya kesalahan logika maupun inkonsistensi antara model perancangan dan implementasi sistem. Menurut Sommerville (2016), aktivitas implementasi dan pengujian merupakan komponen yang menyerap proporsi biaya terbesar dalam siklus pengembangan perangkat lunak. Selain itu, perubahan kode yang tidak diikuti dengan pembaruan dokumentasi menyebabkan rancangan arsitektur menjadi tidak lagi merepresentasikan kondisi implementasi yang sebenarnya.

Untuk mengatasi berbagai permasalahan tersebut, pendekatan *Model-Driven Engineering* (MDE) dikembangkan dengan menggeser fokus pengembangan perangkat lunak dari penulisan kode secara manual menuju pemodelan pada tingkat abstraksi yang lebih tinggi. Dalam paradigma ini, model diposisikan sebagai artefak utama yang merepresentasikan struktur, perilaku, dan kebutuhan sistem, sedangkan implementasi perangkat lunak dihasilkan melalui proses transformasi secara otomatis. Pendekatan ini bertujuan meningkatkan produktivitas pengembangan, mengurangi kesalahan implementasi, serta menjaga konsistensi antara model dan kode program (Ciccozzi, 2018; Starrett, 2016). Selain itu, berbagai penelitian

menunjukkan bahwa MDE memiliki tujuan yang sejalan dengan *platform low-code*, yaitu mengurangi kompleksitas pengembangan melalui abstraksi model, *Domain-Specific Language* (DSL), serta otomatisasi transformasi model menjadi artefak perangkat lunak (Di Ruscio et al., 2022; Neis et al., 2023).

Salah satu pendekatan yang berkembang dalam paradigma *Model-Driven Engineering* adalah *Executable and Translatable Unified Modeling Language* (xtUML), yang memperluas fungsi *Unified Modeling Language* (UML) dari sekadar dokumentasi menjadi model yang dapat digunakan sebagai dasar transformasi perangkat lunak. Berbeda dengan UML konvensional yang umumnya hanya digunakan sebagai representasi statis sistem, xtUML mengintegrasikan aspek struktural (*Class Diagram*), aspek perilaku (*Action Semantics*), dan aspek kontrol (*State Machine*) dalam satu model yang utuh. Dalam pendekatan ini, model compiler berperan mentransformasikan *Platform Independent Model* (PIM) menjadi *Platform Specific Model* (PSM) tanpa menghilangkan makna semantik model (Ciccozzi, 2018; Starrett, 2016). Konsep xtUML tidak diimplementasikan secara penuh, melainkan diadopsi sebagai landasan konseptual dalam perancangan sebuah *Domain-Specific Language* (DSL) yang merepresentasikan struktur dan perilaku sistem sebelum ditransformasikan menjadi artefak *backend* Laravel.

Meskipun paradigma MDE telah berkembang pesat, implementasi *model compiler* pada domain aplikasi web modern tetap menghadapi berbagai keterbatasan. Sebagian besar *code generator* yang tersedia tetap berorientasi pada transformasi model menjadi artefak berbasis skema basis data (*database-driven code generation*), seperti model, migrasi, maupun operasi *Create, Read, Update, dan Delete* (CRUD) dasar (Burden et al., 2011). Pendekatan tersebut efektif dalam menghasilkan struktur data dan komponen aplikasi yang bersifat statis, namun belum mampu

mentransformasikan aspek perilaku (*action semantics*) yang merepresentasikan logika bisnis aplikasi secara utuh.

Keterbatasan tersebut menyebabkan pengembang tetap dituntut mengimplementasikan logika bisnis, alur kendali (*control flow*), mekanisme validasi, serta komponen pendukung lainnya secara manual pada lapisan *controller* maupun *service*. Kondisi ini mengurangi manfaat utama pendekatan MDE, yaitu menjaga konsistensi antara model dan implementasi, sekaligus berpotensi memunculkan kembali *round-trip engineering gap* akibat perubahan kode yang tidak lagi berdasar pada model. Di sisi lain, pendekatan yang menggunakan interpreter pada saat runtime juga dilaporkan menimbulkan overhead memori dan penurunan performa dibandingkan pendekatan berbasis kompilasi yang menghasilkan kode *native* (Hammer et al., 2025).

Perkembangan penelitian terkini juga menunjukkan bahwa kebutuhan otomasi pengembangan perangkat lunak tidak lagi terbatas pada kemampuan menghasilkan kode secara otomatis, tetapi juga mencakup validasi model, skalabilitas *generator*, kemampuan adaptasi terhadap berbagai domain aplikasi, serta dukungan terhadap transformasi model yang lebih komprehensif (Buschhaus et al., 2024; Dalibor et al., 2022; Martinez-Lasaca et al., 2023). Temuan-temuan tersebut menunjukkan bahwa tetap terdapat peluang penelitian untuk mengembangkan model compiler yang tidak hanya menghasilkan struktur aplikasi, tetapi juga mampu mentransformasikan perilaku sistem secara lebih lengkap ke dalam *framework* aplikasi web modern.

Laravel merupakan *framework* berbasis PHP yang secara luas diadopsi dalam pengembangan aplikasi web karena menyediakan arsitektur *Model-View-Controller* (MVC), dukungan *Object Relational Mapping* (ORM), mekanisme *routing*, serta ekosistem pengembangan yang matang. Tingginya tingkat adopsi Laravel oleh komunitas pengembang menjadikannya sebagai

Salah satu platform yang representatif untuk implementasi pendekatan MDE pada pengembangan aplikasi web modern (JetBrains, 2025; Paolone et al., 2020). Selain itu, dominasi PHP sebagai bahasa pemrograman sisi server juga menunjukkan bahwa penelitian pada platform ini memiliki relevansi yang tinggi terhadap kebutuhan industri perangkat lunak (W3techs, 2026).

Berdasarkan kondisi tersebut, penelitian ini mengusulkan pengembangan *model compiler* berbasis DSL yang mengadopsi konsep *action semantics* pada xtUML untuk mentransformasikan spesifikasi model menjadi artefak *backend* Laravel secara otomatis. DSL yang dirancang dalam penelitian ini mengadaptasi konsep OAL dan dikembangkan secara khusus agar mampu merepresentasikan *model data*, *routing*, *middleware*, serta logika bisnis sesuai karakteristik *framework* Laravel. Berbeda dengan code generator yang umumnya hanya menghasilkan struktur data atau operasi CRUD dasar, *model compiler* yang dikembangkan pada penelitian ini dirancang untuk mentransformasikan aspek perilaku (*action semantics*) beserta komponen aplikasi seperti *routing* dan *middleware* dalam satu proses kompilasi, sehingga implementasi model dapat diterjemahkan menjadi artefak *backend* Laravel secara lebih lengkap dan konsisten.

Transformasi spesifikasi DSL menjadi artefak *backend* Laravel dilakukan melalui tahapan *parsing*, analisis struktur model, dan *code generation* secara otomatis menggunakan ANTLR4 sebagai *parser generator*. Representasi hasil *parsing* selanjutnya diproses menggunakan mekanisme *Visitor Pattern* untuk menghasilkan artefak *backend* Laravel yang sesuai dengan struktur arsitektur *framework*. Dengan pendekatan tersebut, penelitian ini diharapkan dapat memperluas penerapan MDE pada pengembangan aplikasi web sekaligus menyediakan alternatif implementasi *model compiler* berbasis DSL yang mampu mengurangi ketergantungan terhadap penulisan kode secara manual.

Temuan berbagai kajian menunjukkan bahwa pendekatan MDE dan *model compiler* telah berkembang pesat dalam mendukung otomatis pengembangan perangkat lunak. Namun, pada domain aplikasi web, implementasi yang tersedia tetap banyak berfokus pada proses transformasi struktur data dan operasi CRUD dasar, sehingga aspek perilaku seperti *action semantics* serta komponen pendukung aplikasi, seperti *routing* dan *middleware*, belum banyak ditangani secara terpadu. Selain itu, literatur yang secara khusus membahas *model compiler* untuk menghasilkan artefak *backend* Laravel masih terbatas. Kondisi tersebut menunjukkan adanya ruang pengembangan bagi penelitian ini untuk merancang *model compiler* berbasis *Domain-Specific Language* (DSL) yang mengadopsi konsep xtUML untuk transformasi artefak *backend* Laravel menggunakan ANTLR4.

B. Rumusan Masalah

Berdasarkan latar belakang dan *research gap* yang telah diuraikan, rumusan masalah dalam penelitian ini dirumuskan sebagai berikut:

1. Bagaimana merancang dan mengembangkan *model compiler* berbasis DSL yang mengadopsi konsep xtUML menggunakan ANTLR4 untuk mentransformasikan spesifikasi model menjadi artefak *backend* Laravel secara otomatis?
2. Bagaimana hasil validasi fungsional terhadap artefak *backend* Laravel yang dihasilkan oleh *model compiler* berdasarkan pengujian *black-box* pada skenario CRUD relasional, *routing*, dan *middleware*?

C. Tujuan Penelitian

Berdasarkan rumusan masalah tersebut, tujuan penelitian ini adalah sebagai berikut:

1. Mengembangkan *model compiler* berbasis *Domain-Specific Language* (DSL) yang mengadopsi xtUML menggunakan ANTLR4 untuk

- mentransformasikan spesifikasi model menjadi artefak *backend* Laravel secara otomatis.
2. Melakukan validasi terhadap artefak *backend* Laravel yang dihasilkan oleh *model compiler* menggunakan pengujian *black-box* pada skenario CRUD relasional, routing, dan *middleware*.

D. Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Manfaat Teoretis

Memberikan kontribusi terhadap pengembangan ilmu pengetahuan pada bidang MDE, khususnya dalam perancangan DSL yang mengadopsi konsep xtUML serta pengembangan *model compiler* untuk mentransformasikan model menjadi artefak *backend* Laravel secara otomatis.

2. Manfaat Praktis

Menyediakan alternatif *tooling* berupa *model compiler* berbasis DSL yang dapat membantu pengembang perangkat lunak menghasilkan artefak *backend* Laravel secara otomatis, sehingga dapat meningkatkan produktivitas pengembangan, menjaga konsistensi antara model dan implementasi, serta mengurangi kesalahan akibat penulisan kode secara manual.

BAB V

PENUTUP

A. Kesimpulan

Penelitian ini telah berhasil merancang dan membangun arsitektur model compiler berbasis ANTLR4 dengan pendekatan *Visitor Pattern* yang mampu memetakan spesifikasi tekstual OAL menjadi komponen arsitektur MVC Laravel secara valid (Parr, 2013).

Implementasi *compiler* ini menghasilkan artefak Laravel berupa model, *controller*, *route*, *middleware*, dan migrasi database yang dapat diuji secara fungsional melalui *Black-box Testing*. Hasil pengujian menunjukkan bahwa *endpoint* yang dihasilkan berjalan sesuai skenario uji, sehingga artefak yang dihasilkan *compiler* dapat dinyatakan valid pada ruang lingkup operasi CRUD relasional, validasi data, proteksi *middleware*, dan deklarasi *routing* yang ditetapkan dalam penelitian ini (Laravel, 2026; Myers et al., 2011).

B. Saran

Untuk pengembangan selanjutnya, disarankan agar ruang lingkup compiler ini diperluas untuk mendukung otomatisasi pembuatan komponen *User Interface* (*Blade views*), validasi model yang lebih kaya, serta integrasi *third-party library* secara modular. Arah ini selaras dengan tantangan MDE modern terkait kebutuhan *tooling* yang skalabel dan mampu menjembatani abstraksi model dengan implementasi nyata (Bucchiarone et al., 2020; Martinez-Lasaca et al., 2023).

DAFTAR PUSTAKA

- Aho, A. V, Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Addison-Wesley.
- Brambilla, M., Cabot, J., & Wimmer, M. (2017). *Model-Driven Software Engineering in Practice* (2nd Edition). Morgan & Claypool Publishers.
- Bucchiarone, A., Cabot, J., Paige, R. F., & Pierantonio, A. (2020). Grand Challenges in Model-Driven Engineering: An Analysis of the State of the Research. *Software and Systems Modeling*, 19, 5–13.
<https://doi.org/10.1007/s10270-019-00773-6>
- Burden, H., Heldal, R., & Siljamäki, T. (2011). Executable and Translatable UML - How Difficult Can it Be? *18th Asia-Pacific Software Engineering Conference*, 357–364.
- Buschhaus, C., Gerasimov, A., Kirchhof, J. C., Michael, J., Netz, L., Rumpe, B., & Stüber, S. (2024). Lessons Learned from Applying Model-Driven Engineering in 5 Domains: The Success Story of the MontiGem Generator Framework. *Science of Computer Programming*, 232, 103033. <https://doi.org/10.1016/j.scico.2023.103033>
- Ciccozzi, F. (2018). On the Automated Translational Execution of the Action Language for Foundational UML. *Software and Systems Modeling*, 17, 1311–1337. <https://doi.org/10.1007/s10270-016-0556-7>
- Dalibor, M., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Varga, S., & Wortmann, A. (2022). Generating Customized Low-Code Development Platforms for Digital Twins. *Journal of Computer Languages*, 70, 101117. <https://doi.org/10.1016/j.col.2022.101117>
- Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-Code Development and Model-Driven Engineering:

- Two Sides of the Same Coin? *Software and Systems Modeling*, 21, 437–446. <https://doi.org/10.1007/s10270-021-00970-2>
- Dijkstra, E. W. (1976). *A Discipline of Programming*. Prentice-Hall.
- Fauzi, M. A., Tribiakto, H., Moniva, A., Amir, F., Ilyas, I. K., & Utami, E. (2023). Systematic Literature Reviews on Rapid Application Development Information System. *Bulletin of Computer Science and Electrical Engineering*, 4(1), 57–64. <https://doi.org/10.25008/bcsee.v4i1.1181>
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.
- Hammer, Maschotta, & Zimmermann. (2025). Compiler-Like Code Generation for fUML: Reducing Overhead in Executable UML. *IEEE Transactions on Software Engineering*, 1–12.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–105. <https://doi.org/10.2307/25148625>
- JetBrains. (2025). *The State of PHP 2025*. <https://blog.jetbrains.com/phpstorm/2025/10/state-of-php-2025/>
- Laravel. (2026). *Laravel 12.x Documentation*. <https://laravel.com/docs/12.x>
- Martinez-Lasaca, F., Diez, P., Guerra, E., & de Lara, J. (2023). Dandelion: A Scalable, Cloud-Based Graphical Language Workbench for Industrial Low-Code Development. *Journal of Computer Languages*, 76, 101217. <https://doi.org/10.1016/j.cola.2023.101217>
- Mellor, S. J., & Balcer, M. J. (2002). *Executable UML: A Foundation for Model-Driven Architecture*. Addison-Wesley Professional.

- Mernik, M., Heering, J., & Sloane, A. M. (2005). When and How to Develop Domain-Specific Languages. *ACM Computing Surveys*, 37(4), 316–344. <https://doi.org/10.1145/1118890.1118892>
- Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). John Wiley & Sons.
- Neis, P., Wehrmeister, M. A., Mendes, M. F., & Pesente, J. R. (2023). Applying a Model-Driven Approach to the Development of Power Plant SCADA/EMS Software. *International Journal of Electrical Power & Energy Systems*, 153, 109336. <https://doi.org/10.1016/j.ijepes.2023.109336>
- Object Management Group. (2003). *MDA Guide Version 1.0.1* (J. Miller & J. Mukerji, Eds.). Object Management Group. <https://www.omg.org/cgi-bin/doc?omg/03-06-01>
- Paolone, G., Marinelli, M., Paesani, R., & Di Felice, P. (2020). Automatic Code Generation of MVC Web Applications. *Computers*, 9(3), 56. <https://doi.org/10.3390/computers9030056>
- Parr, T. (2013). *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf.
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Schmidt, D. C. (2006). *Model-driven engineering*. <http://mic.omg.org>
- Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
- Starrett, C. (2016). xtUML: Current and Next State of a Modeling Dialect. *CEUR Workshop Proceedings*, 1709, 44–51.

W3techs. (2026). *Usage Statistics of PHP for Websites*.

<https://w3techs.com/technologies/details/pl-php>

Whittle, J., Hutchinson, J., & Rouncefield, M. (2014). The State of Practice in Model-Driven Engineering. *IEEE Software*, 31(3), 79–85.

<https://doi.org/10.1109/MS.2013.65>

