# Rancang Bangun Sistem Analisis Dependensi Statis dengan Antarmuka Grafis Interaktif

**TUGAS AKHIR**


Disusun oleh:

Rama Audra Aji Pangestu

22106050037





**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA**

**YOGYAKARTA**


**2026**

# LEMBAR PENGESAHAN

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-1231/Un.02/DST/PP.00.9/06/2026

Tugas Akhir dengan judul : Rancang Bangun Sistem Analisis Dependensi Statis dengan Antarmuka Grafis Interaktif

yang dipersiapkan dan disusun oleh:

| | |
|---|---|
| Nama | : RAMA AUDRA AJI PANGESTU |
| Nomor Induk Mahasiswa | : 22106050037 |
| Telah diujikan pada | : Selasa, 26 Mei 2026 |
| Nilai ujian Tugas Akhir | : A |

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

## TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Ir. Aulia Faqih Rifa'i, M.Kom.
SIGNED

Valid ID: 6a2622d849cd5

Penguji I

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

Valid ID: 6a23aa45ae339

Penguji II

Eko Hadi Gunawan, M.Eng.
SIGNED

Valid ID: 6a25bcdb5574f

Yogyakarta, 26 Mei 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a2628b916ccd5

# LEMBAR PERNYATAAN

## SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

| | |
|---|---|
| Nama | : Rama Audra Aji Pangestu |
| NIM | : 22106050037 |
| Program Studi | : Informatika |
| Fakultas | : Sains dan Teknologi |

Dengan ini menyatakan bahwa isi skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi dan sesungguhnya skripsi ini merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta, 19 Mei 2026

Rama Audra Aji Pangestu

# LEMBAR PERSETUJUAN

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

## SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

*Assalamu'alaikum wr. wb.*

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Rama Audra Aji Pangestu
NIM : 22106050037
Judul Skripsi : Rancang Bangun Sistem Analisis Dependensi Statis dengan Antarmuka Grafis Interaktif

sudah dapat diajukan kembali kepada Program Studi Matematika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

*Wassalamu'alaikum wr. wb.*

Yogyakarta,
Pembimbing

Dr. Ir. Aulia Faqih Rifa'i, M.Kom.
NIP. 19860306 201101 1 009

## ABSTRAK

Pemahaman basis kode yang tidak familier menjadi tantangan penting dalam pengembangan perangkat lunak, terutama pada proses *onboarding* pengembang ke basis kode baru. Tugas Akhir ini bertujuan merancang dan membangun *Tauta*, yaitu sistem analisis dependensi statis dengan antarmuka grafis interaktif untuk mendukung *program comprehension* pada basis kode *JavaScript/TypeScript*. Sistem dikembangkan menggunakan pendekatan *Iterative and Incremental* melalui tiga fokus utama, yaitu pembangunan fondasi analisis *dependensi statis*, pengembangan metrik arsitektur, dan penyempurnaan antarmuka berbasis *decision-support*. *Tauta* mampu membentuk *dependency graph*, menampilkan relasi *dependencies* dan *dependents*, mendeteksi *circular dependency*, mengidentifikasi *orphan candidate*, serta menyajikan metrik dan sinyal analisis melalui pendekatan *diagnosis-first* untuk membantu prioritas tinjauan pengembang. Evaluasi program *comprehension* dilakukan dengan membandingkan pengerjaan manual dan pengerjaan menggunakan *Tauta* berdasarkan *Task Completion Time* (TCT) dan *Accuracy*. Hasil evaluasi menunjukkan *Accuracy* keseluruhan meningkat dari 25,00% menjadi 100,00%. Total TCT pada Tugas 1 dan Tugas 2 menurun dari 595,34 detik menjadi 202,70 detik, dengan penghematan waktu 65,95%. Pada Tugas 3, seluruh partisipan gagal menemukan *orphan candidate* secara manual, tetapi berhasil saat menggunakan *Tauta*. Hasil ini mengindikasikan bahwa *Tauta* mendukung pemahaman struktur dependensi secara lebih cepat dan tepat pada skenario uji yang digunakan.

Kata kunci: Analisis Dependensi Statis, *Program Comprehension, Dependency Graph, JavaScript/TypeScript*, Antarmuka Interaktif.

## ABSTRACT

*Understanding an unfamiliar codebase is an important challenge in software development, especially when developers are onboarded into a new codebase. This final project aims to design and develop Tauta, a static dependency analysis system with an interactive graphical interface to support program comprehension in JavaScript/TypeScript codebases. The system was developed using an Iterative and Incremental approach through three main focuses: building the foundation of static dependency analysis, developing architectural metrics, and refining a decision-support-oriented interface. Tauta can construct a dependency graph, display dependencies and dependents, detect circular dependencies, identify orphan candidates, and present analysis metrics and signals through a diagnosis-first approach to guide developers' review priorities. The program comprehension evaluation compared manual task completion with task completion using Tauta based on Task Completion Time (TCT) and Accuracy. The results show that overall Accuracy increased from 25.00% to 100.00%. The total TCT for Task 1 and Task 2 decreased from 595.34 seconds to 202.70 seconds, resulting in 65.95% time savings. In Task 3, all participants failed to identify orphan candidates manually but succeeded when using Tauta. These results indicate that Tauta supports faster and more accurate dependency-structure comprehension in the evaluated scenario.*

*Keywords: Static Dependency Analysis, Program Comprehension, Dependency Graph, JavaScript/TypeScript, Interactive Interface.*

# MOTTO

*"Everyone is fighting a battle you know nothing about. Be kind. Always."*

Brad Meltzer

*"Be kind, for everyone you meet is fighting a hard battle."*

Ian Maclaren

## KATA PENGANTAR

*Alhamdulillāhi Rabbil 'Ālamīn,* segala puji bagi Allah SWT atas rahmat dan pertolongan-Nya sehingga penulis dapat menyelesaikan Tugas Akhir berjudul "Rancang Bangun Sistem Analisis Dependensi Statis dengan Antarmuka Grafis Interaktif". Selawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad SAW. Dengan penuh kerendahan hati, penulis menyadari bahwa karya ini tidak terwujud tanpa doa, dukungan, nasihat, dan bantuan dari berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Prof. Noorhaidi Hasan S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta, yang telah menyediakan naungan ilmu tempat penulis bertumbuh dan belajar.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta, atas segala fasilitas dan ruang teduh bagi mahasiswa untuk terus berkarya.
3. Bapak Dr. Muhammad Mustakim, S.T. M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta, atas dedikasi dan arahannya yang senantiasa menjaga langkah para mahasiswa tetap berada di jalur yang terang.
4. Bapak Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng. selaku Dosen Penasihat Akademik. Terima kasih atas setiap nasihat yang menenangkan dan arahan yang membimbing penulis sejak hari pertama menapakkan kaki di kampus ini hingga menyentuh garis akhir.
5. Bapak Dr. Ir. Aulia Faqih Rifa'i, M.Kom. selaku Dosen Pembimbing. Terima kasih atas kesabaran seluas samudra dalam membimbing, meluangkan waktu, dan mengurai setiap kerumitan pemikiran penulis hingga karya ini menemukan bentuk terbaiknya.
6. Kedua orang tua penulis, Bapak dan ibu tercinta. Kepada dua pahlawan yang peluhnya tak pernah dihitung dan doanya tak pernah putus menembus

langit malam. Terima kasih telah menjadi rumah tempat penulis kembali, atas cinta yang tak bersyarat, dan pengorbanan yang tak akan pernah bisa terbalas hanya oleh untaian kata dalam lembaran kertas ini.

7. Teman-teman Grup Nahan Tawa, Terima kasih telah menjadi saksi perjalanan hidup sejak riuhnya masa abu-abu hingga kini. Kehadiran kalian adalah tawa yang selalu berhasil mengusir penat dan pengingat bahwa penulis tidak pernah berjalan sendirian.
8. Teman-teman Matdis dan Musang, Terima kasih telah menjadi keluarga pertama sejak awal masa mahasiswa baru. Bersama kalian, hiruk-pikuk perkuliahan, tumpukan tugas, hingga malam-malam panjang terasa lebih ringan dan penuh warna.
9. Teman-teman Neroform (informatika angkatan 2022), Terima kasih atas solidaritas, bahu-membahu, dan memori kebersamaan sebagai satu angkatan yang tangguh. Kita memulai ini bersama, dan semoga kesuksesan menyertai langkah kita semua di masa depan.
10. Teman-teman KKN Sangkrek, Terima kasih atas cerita singkat yang membekas selamanya, atas hangatnya kekeluargaan, dan pelajaran hidup di tempat pengabdian yang senantiasa dirindukan.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna. Kritik dan saran yang membangun akan penulis terima dengan tangan terbuka. Akhirnya, semoga karya kecil ini dapat menjadi setitik manfaat, betapa pun sederhananya, dan menjadi pijakan bagi pengembangan yang lebih bermakna di masa yang akan datang.



Yogyakarta, 26 Mei2026

Penyusun

Rama Audra Aji Pangestu

22106050037

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Perangkat lunak tidak berhenti pada tahap pembangunan awal, tetapi terus berevolusi melalui penambahan fitur, perbaikan cacat, dan penyesuaian terhadap kebutuhan baru [1]. Konsekuensinya, pemeliharaan menjadi bagian dominan dalam siklus hidup perangkat lunak [1], [2]. Dalam praktiknya, pengembang lebih banyak menghabiskan waktu untuk memahami kode yang sudah ada daripada menulis kode baru. Studi lapangan skala besar pada pengembang profesional melaporkan bahwa rata-rata pengembang menghabiskan sekitar 58% waktunya untuk aktivitas pemahaman program [2].

Tantangan tersebut menjadi semakin berat ketika pengembang berhadapan dengan basis kode yang tidak familier, misalnya pada saat *onboarding* (serah terima pengembangan) [3]. Dalam situasi seperti ini, pengembang tidak cukup hanya membaca berkas satu per satu, tetapi perlu membangun pemahaman mengenai hubungan antar komponen, dan memperkirakan dampak perubahan tanpa selalu memiliki akses pada pengetahuan implisit dari pengembang sebelumnya. Hal ini menegaskan bahwa memahami basis kode yang tidak familier merupakan persoalan nyata yang relevan secara praktis.

Dalam konteks tersebut, analisis dependensi penting karena membantu pengembang melihat struktur keterkaitan antar bagian sistem dan mendukung analisis dampak perubahan (*change impact analysis*) [4]. Sejumlah studi eksperimen terkontrol juga menunjukkan bahwa penyajian struktur perangkat lunak melalui visualisasi interaktif yang dirancang untuk tugas komprehensif dapat meningkatkan kebenaran jawaban dan mengurangi waktu penyelesaian tugas dibandingkan antarmuka konvensional berbasis *file tree*, atau editor tekstual [5]. Dengan demikian, penyajian struktur dependensi yang tepat berpotensi meningkatkan kemampuan pengembang dalam menavigasi basis kode yang kompleks.

Namun, *tool* yang tersedia saat ini seperti *dependency-cruiser, Madge*, *Skott,* dan *Knip* belum sepenuhnya menjawab kebutuhan tersebut secara menyeluruh. Meskipun beberapa *tool* tersebut sudah kuat dalam ekstraksi relasi, visualisasi graf dasar, deteksi siklus, dan identifikasi artefak yang tidak digunakan, penyajiannya masih cenderung berupa graf mentah atau laporan teknis yang padat. Kondisi tersebut dapat membuat pengembang kesulitan memahami struktur dependensi, menentukan area prioritas tinjauan, dan memperkirakan dampak perubahan. Oleh karena itu, dibutuhkan sistem yang tidak sekadar menampilkan relasi dependensi, tetapi juga mengintegrasikan metrik struktural, informasi perubahan, dan sinyal koordinasi perubahan lintas berkas ke dalam antarmuka interaktif yang lebih terarah guna mendukung *program comprehension*.

Berdasarkan uraian tersebut, *research gap* dalam Tugas Akhir ini terletak pada kebutuhan akan sistem yang tidak hanya mengekstraksi relasi dependensi, tetapi juga menyajikan hasil analisis dalam bentuk antarmuka grafis interaktif yang lebih mendukung *program comprehension* pada basis kode *JavaScript/TypeScrip*t yang tidak familier. Dengan posisi tersebut, arah tugas akhir ini adalah merancang dan membangun sistem analisis dependensi statis dengan antarmuka grafis interaktif yang memadukan relasi dependensi, metrik struktural, informasi perubahan, dan sinyal koordinasi perubahan lintas berkas untuk mendukung pengembang dalam memahami basis kode *JavaScript/TypeScript* yang tidak familier.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan beberapa masalah sebagai berikut:

1. Bagaimana merancang dan membangun sistem analisis dependensi statis pada basis kode *JavaScript/TypeScript* dengan antarmuka grafis interaktif?
2. Bagaimana pengaruh penggunaan sistem analisis dependensi statis interaktif terhadap kecepatan dan ketepatan penyelesaian tugas pengembang dalam memahami basis kode *JavaScript/TypeScript* yang tidak familier?

## 1.3 Batasan Masalah

Agar arah Tugas Akhir tetap konsisten dan sesuai dengan sasaran yang diharapkan, ruang lingkup Tugas Akhir ini dibatasi pada hal-hal berikut:

1. Tugas Akhir ini difokuskan pada rancang bangun sistem analisis dependensi statis untuk basis kode *JavaScript/TypeScript*, khususnya berkas internal berekstensi *.js*, *.jsx*, *.ts*, dan *.tsx*. Tugas Akhir ini tidak mencakup analisis multi-bahasa, serta tidak memodelkan dependensi dari *library* eksternal atau *package* pihak ketiga sebagai *node* utama dalam graf dependensi.
2. Pada proyek berbentuk *monorepo*, analisis dibatasi pada aplikasi, *package*, atau subdirektori tertentu yang dipilih sebagai target analisis. Tugas Akhir ini tidak mencakup analisis *root monorepo* secara penuh lintas seluruh *workspace* atau *package.*
3. Analisis yang dilakukan bersifat statis dan berfokus pada relasi dependensi antarberkas yang dapat dikenali dari struktur kode sumber, seperti relasi impor, relasi ekspor, *require*, dan pola dependensi statis lain yang didukung sistem. Dengan demikian, graf yang dibentuk pada Tugas Akhir ini merepresentasikan keterhubungan struktural antarberkas, bukan alur eksekusi program secara penuh.
4. Tugas Akhir ini tidak mencakup analisis perilaku runtime, alur data, alur kontrol, alur request-response API, pemetaan model ke basis data, pemetaan proses bisnis backend, atau hubungan semantik antarfungsi yang hanya dapat dipastikan melalui eksekusi program, konfigurasi framework tertentu, *dependency* injection, routing dinamis, atau konvensi runtime lainnya. Analisis semacam itu membutuhkan pendekatan berbeda, seperti control-flow analysis, data-flow analysis, call *graph* analysis, framework-specific analysis, atau dynamic analysis, sehingga berada di luar ruang lingkup Tugas Akhir ini.
5. Tugas Akhir ini tidak mencakup pengukuran kompleksitas kode pada level fungsi atau blok kode, seperti cyclomatic complexity, cognitive complexity,

Halstead metrics, atau ukuran kompleksitas algoritmik lainnya. Metrik yang digunakan difokuskan pada struktur dependensi dan koordinasi perubahan, seperti *Ca, Ce, Instability*, risiko penyebaran perubahan, *blast radius, hotspot* berbasis riwayat perubahan, serta sinyal koordinasi perubahan lintas berkas yang dapat dikenali secara statis.

6. Sistem tidak ditujukan untuk melakukan perbaikan kode otomatis, refactoring otomatis, atau pembuatan patch perubahan. Hasil analisis diposisikan sebagai alat bantu pemahaman, peninjauan, dan pertimbangan awal sebelum pengembang melakukan perubahan kode secara manual.
7. Informasi perubahan berbasis riwayat Git digunakan apabila riwayat tersebut tersedia pada repositori yang dianalisis. Apabila riwayat perubahan tidak tersedia atau tidak lengkap, maka indikator yang bergantung pada informasi tersebut berada di luar cakupan analisis penuh.
8. Metrik dan indikator, seperti relasi dependensi, metrik struktural, risiko penyebaran perubahan, *hotspot*, dan sinyal koordinasi perubahan lintas berkas, digunakan sebagai dukungan analisis dan prioritas tinjauan. Indikator tersebut tidak dimaksudkan sebagai ukuran mutlak untuk menilai kualitas perangkat lunak, memprediksi cacat, atau menentukan risiko perubahan secara universal.
9. Fokus Tugas Akhir adalah mendukung pemahaman struktur basis kode yang tidak familier dan analisis dampak perubahan melalui penyajian hasil analisis dependensi secara interaktif. Evaluasi dibatasi pada pengujian fungsional sistem dan evaluasi skenario *program comprehension* . Tugas Akhir ini tidak mengukur keberhasilan proses *onboarding* secara menyeluruh, dan hasil evaluasi tidak dimaksudkan untuk digeneralisasi bagi seluruh profil pengembang, seluruh jenis proyek *JavaScript/TypeScript*, atau seluruh konteks pengembangan perangkat lunak.

## 1.4 Tujuan Tugas Akhir

Berdasarkan rumusan masalah yang telah diuraikan, Tugas Akhir ini bertujuan untuk mencapai beberapa hal sebagai berikut:

1. Merancang dan membangun sistem analisis dependensi statis dengan antarmuka grafis interaktif pada basis kode *JavaScript/TypeScript*.
2. Mengevaluasi pengaruh penggunaan sistem analisis dependensi statis interaktif terhadap tingkat kecepatan dan keakuratan dalam penyelesaian tugas pengembang dalam memahami basis kode *JavaScript/TypeScript* yang tidak familier

## 1.5 Manfaat Tugas Akhir

Tugas Akhir ini diharapkan dapat memberikan manfaat teoretis bagi pengembangan keilmuan serta manfaat praktis sebagai bahan pertimbangan bagi pihak terkait.

### 1.5.1 Manfaat Teoritis

Tugas Akhir ini diharapkan dapat memberikan manfaat teoretis, di antaranya sebagai berikut:

1. Memberikan kontribusi terhadap pengembangan kajian rekayasa perangkat lunak, khususnya pada bidang analisis dependensi statis, visualisasi perangkat lunak interaktif, pemahaman basis kode yang tidak familier, dan analisis dampak perubahan.
2. Memperkaya pemahaman konseptual mengenai integrasi hasil analisis dependensi, metrik struktural, informasi perubahan, dan sinyal koordinasi perubahan lintas berkas ke dalam antarmuka grafis interaktif yang lebih mendukung program comprehension.
3. Menjadi referensi bagi Tugas Akhir selanjutnya yang membahas pengembangan alat analisis basis kode, visualisasi dependensi, penyajian

informasi arsitektural, serta dukungan terhadap analisis dampak perubahan dan pemahaman kontrak lintas berkas.

4. Memberikan sudut pandang bahwa analisis dependensi tidak hanya berfungsi untuk menampilkan relasi antarberkas, tetapi juga dapat diarahkan untuk mendukung interpretasi, pemahaman struktur perangkat lunak, peninjauan perubahan, dan pembacaan area kontrak lintas berkas yang sensitif terhadap perubahan secara lebih efektif.

### 1.5.2 Manfaat Praktis

Tugas Akhir ini diharapkan dapat memberikan manfaat praktis, di antaranya sebagai berikut:

1. Bagi pengembang perangkat lunak, Tugas Akhir ini diharapkan dapat menyediakan sistem yang membantu menelusuri hubungan dependensi antarberkas, memahami struktur basis kode, memperkirakan dampak perubahan, dan mengenali kontrak lintas berkas yang memerlukan perhatian lebih sebelum perubahan dilakukan.
2. Bagi tim pengembangan perangkat lunak, Tugas Akhir ini diharapkan dapat mendukung proses onboarding , handover, audit arsitektur, dan diskusi teknis melalui penyajian informasi struktur sistem yang lebih terarah dan lebih mudah dipahami.
3. Bagi pihak yang melakukan pemeliharaan dan evaluasi perangkat lunak, Tugas Akhir ini diharapkan dapat menjadi bahan pertimbangan dalam mengidentifikasi area yang memerlukan perhatian lebih, khususnya pada bagian sistem yang memiliki keterhubungan tinggi, berpotensi menimbulkan dampak perubahan yang luas, atau memerlukan koordinasi perubahan lintas berkas.
4. Bagi lingkungan akademik, Tugas Akhir ini diharapkan dapat dimanfaatkan sebagai media pembelajaran dalam memahami konsep dependensi, struktur

perangkat lunak, analisis dampak perubahan, serta penerapan visualisasi interaktif pada analisis basis kode JavaScript/TypeScript.

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian, dan evaluasi sistem yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut.

1. Tugas Akhir ini berhasil merancang dan membangun *Tauta* sebagai sistem analisis dependensi statis pada basis kode *JavaScript/TypeScript* dengan antarmuka grafis interaktif. Sistem dikembangkan secara *Iterative and Incremental* dan mampu membentuk *dependency graph*, menampilkan *dependencies* dan *dependents*, mendeteksi *circular dependency*, serta mengidentifikasi *orphan candidate*. *Tauta* juga menyajikan metrik dan sinyal analisis melalui pendekatan *diagnosis-first* sehingga pengguna dapat menentukan area kode yang perlu ditinjau.
2. Hasil evaluasi menunjukkan bahwa *Tauta* mendukung kecepatan dan ketepatan penyelesaian tugas *program comprehension* pada skenario uji yang digunakan. Ketepatan jawaban meningkat dari 25,00% menjadi 100,00%, sedangkan total *Task Completion Time* pada Tugas 1 dan Tugas 2 menurun dari 595,34 detik menjadi 202,70 detik, dengan penghematan waktu 392,64 detik atau 65,95%. Pada Tugas 3, seluruh partisipan gagal menemukan *orphan candidate* secara manual, tetapi seluruhnya berhasil saat menggunakan *Tauta*. Hasil ini tetap dibatasi oleh jumlah partisipan, penggunaan satu objek uji, cakupan tugas yang terbatas, dan kemungkinan *learning effect*.

## 5.2 Saran

Berdasarkan hasil pengembangan dan evaluasi, terdapat beberapa saran untuk pengembangan berikutnya.

1. Perluasan Variabel Evaluasi *Program Comprehension*

   Evaluasi berikutnya dapat mencakup metrik dan sinyal lain yang sudah tersedia, seperti *Blast Radius, Propagation Risk, Instability, churn, hotspot*, dan *connascence*.

2. Penambahan Jumlah dan Variasi Partisipan

   Evaluasi berikutnya perlu melibatkan lebih banyak partisipan dengan profil yang lebih beragam, seperti pengembang junior, pengembang berpengalaman, dan pengembang dari ekosistem teknologi lain.

3. Penggunaan Lebih dari Satu Objek Uji

   Evaluasi selanjutnya disarankan menggunakan beberapa proyek *JavaScript/TypeScript* dengan ukuran, struktur, kerangka kerja, dan gaya arsitektur yang berbeda.

4. Penyempurnaan Desain Evaluasi

   Desain evaluasi berikutnya dapat menggunakan *counterbalancing* untuk mengurangi *learning effect* dan memperkuat perbandingan antara kondisi manual dan kondisi menggunakan sistem.

5. Perluasan Dukungan Bahasa Pemrograman

   Pengembangan berikutnya dapat mempertimbangkan dukungan parser untuk bahasa lain, seperti *Python*, *Go*, atau *Java*, sebagai arah pengembangan jangka panjang.

# DAFTAR PUSTAKA

[1] K. H. Bennett dan V. T. Rajlich, "Software maintenance and evolution: a roadmap," dalam *Proceedings of the Conference on The Future of Software Engineering*, Limerick Ireland: ACM, Mei 2000, hlm. 73–87. doi: 10.1145/336512.336534.

[2] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, dan S. Li, "Measuring Program Comprehension: A Large-Scale Field Study with Professionals," *IEEE Trans. Softw. Eng.*, vol. 44, no. 10, hlm. 951–976, Okt 2018, doi: 10.1109/TSE.2017.2734091.

[3] I. Santos, K. R. Felizardo, M. A. Gerosa, dan I. Steinmacher, "Software Solutions for Newcomers' Onboarding in Software Projects: A Systematic Literature Review," 28 Agustus 2024, *arXiv*: arXiv:2408.15989. doi: 10.48550/arXiv.2408.15989.

[4] M. Kretsou, E.-M. Arvanitou, A. Ampatzoglou, I. Deligiannis, dan V. C. Gerogiannis, "*Change impact* analysis: A systematic mapping study," *J. Syst. Softw.*, vol. 174, hlm. 110892, Apr 2021, doi: 10.1016/j.jss.2020.110892.

[5] B. Cornelissen, A. Zaidman, dan A. Van Deursen, "A Controlled Experiment for Program Comprehension through Trace Visualization," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, hlm. 341–355, Mei 2011, doi: 10.1109/TSE.2010.47.

[6] A. Von Mayrhauser dan A. M. Vans, "Program comprehension during software maintenance and evolution," *Computer*, vol. 28, no. 8, hlm. 44–55, Agu 1995, doi: 10.1109/2.402076.

[7] M.-A. Storey, "Theories, methods and tools in program comprehension: past, present and future," dalam *13th International Workshop on Program Comprehension (IWPC'05)*, St. Louis, MO, USA: IEEE, 2005, hlm. 181–191. doi: 10.1109/WPC.2005.38.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, dan C. Stein, "Introduction to Algorithms".

[9] R. C. Martin, "OO Design Quality Metrics: An Analysis of *Dependencies*," 1994.

[10] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[11] R. S. Arnold dan S. A. Bohner, *Software Change Impact Analysis*. dalam IEEE Computer Society Press Tutorial. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996.

[12] M. Page-Jones, "Comparing techniques by means of encapsulation and *connascence*," *Commun. ACM*, vol. 35, no. 9, hlm. 147–151, Sep 1992, doi: 10.1145/130994.131004.

[13] N. Nagappan dan T. Ball, "Use of *Relative* Code Churn Measures to Predict System Defect Density," dalam *Proceedings of the 27th International Conference on Software Engineering (ICSE)*, St. Louis, MO, USA: ACM/IEEE, 2005, hlm. 284–292. doi: 10.1105/ICSE.2005.1553571.

[14] A. Tornhill, Your Code as a Crime Scene: Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs. Raleigh, NC, USA: Pragmatic Bookshelf, 2015.

[15] L. Merino, M. Ghafari, C. Anslow, dan O. Nierstrasz, "A systematic literature review of software visualization evaluation," *J. Syst. Softw.*, vol. 144, hlm. 165–180, Okt 2018, doi: 10.1016/j.jss.2018.06.027.

[16] M. Shahin, P. Liang, dan M. A. Babar, "A Systematic Review of Software *Architecture* Visualization Techniques," *J. Syst. Softw.*, vol. 94, hlm. 161–185, Agu 2014, doi: 10.1016/j.jss.2014.03.071.

[17] J. Sweller, "Cognitive Load During Problem Solving: Effects on Learning," *Cogn. Sci.*, vol. 12, no. 2, hlm. 257–285, Apr 1988, doi: 10.1207/s15516709cog1202_4.

[18] G. A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychol. Rev.*, vol. 63, no. 2,

hlm. 81–97, 1956, doi: 10.1037/h0043158.

[19] R. H. Sprague Jr., "A Framework for the Development of Decision Support Systems," *MIS Q.*, vol. 4, no. 4, hlm. 1–26, Des 1980, doi: 10.2307/248946.

[20] S. Sverweij, *dependency-cruiser*. (2024). GitHub. [Daring]. Tersedia pada: https://github.com/sverweij/*dependency*-cruiser

[21] pahen, *madge*. (2024). GitHub. [Daring]. Tersedia pada: https://github.com/pahen/madge

[22] antoine-coulon, *Skott*. (2026). GitHub. [Daring]. Tersedia pada: https://github.com/antoine-coulon/skott

[23] "Knip Documentation." [Daring]. Tersedia pada: https://github.com/antoine-coulon/skott

[24] C. Larman dan V. R. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, vol. 36, no. 6, hlm. 47–56, Jun 2003, doi: 10.1109/MC.2003.1204375.

[25] I. Sommerville, *Software Engineering*, 10 ed. Pearson, 2016.

[26] G. J. Myers, C. Sandler, dan T. Badgett, *The Art of Software Testing*, 3 ed. John Wiley & Sons, 2011.

[27] P. Ammann dan J. Offutt, *Introduction to Software Testing*, 2 ed. Cambridge University Press, 2016. doi: 10.1017/9781316771273.