Jurnal Ilmiah Sinus (JIS) Vol : 24, No. 2, Juli 2026
ISSN (Print) : 1693-1173 , ISSN (Online): 2548-4028

Efektivitas Requirements Traceability Matrix (RTM) sebagai Scaffolding terhadap Deteksi Defect pada Pengujian Perangkat Lunak Mahasiswa

Afrizal Meka Mulyana[1*)]
[1*)] Program Studi Sistem Informasi, Universitas Tiga Serangkai
[1*)] email: afrizal@tsu.ac.id

ABSTRACT

The Requirements Traceability Matrix (RTM) is widely used as scaffolding to guide students through thorough software testing, yet a rarely tested assumption holds that completing the procedure equates to detecting defects. This study analyzes planted-bug detection by defect category under uniform RTM completion, a condition not previously examined. A descriptive multiple-case study was applied to nine Information Systems student groups (24 students) testing five web applications containing 33 identically planted defects across seven a priori categories. Detection was assessed through strict matching between reported and planted defects, and analyzed using Wilson confidence intervals and Fisher's exact test. Although all groups completed the RTM at 100%, the overall detection rate was only 55.7% (95% CI 43.3-67.5%), ranging from 14.3% to 87.5% across groups. Detection declined sharply by category; observable defects were detected significantly more often than non-observable ones (67.4% vs 27.8%; odds ratio 5.39; p = 0.010). These findings confirm a dissociation between procedural completeness and detection effectiveness. The study contributes category-level empirical evidence that procedural scaffolding ensures completeness but not detection acuity, and recommends enriching scaffolding with explicit emphasis on negative-scenario and non-functional testing.

Keywords *: defect detection; planted bug; requirements traceability matrix; software testing; testing education.*

I. PENDAHULUAN

Kualitas perangkat lunak menentukan keberhasilan sebuah sistem informasi, dan pengujian perangkat lunak adalah mekanisme utama untuk memastikan perangkat lunak berperilaku sesuai spesifikasi serta bebas dari *defect* yang merugikan. Kemampuan menguji bukan semata keterampilan prosedural yang dapat dikuasai dengan mengikuti langkah baku, melainkan keterampilan kognitif yang menuntut penguji mengantisipasi cara sebuah sistem dapat gagal. Studi terkini menunjukkan bahwa banyak mahasiswa memperlakukan pengujian sebagai daftar centang prosedural alih-alih upaya memvalidasi perilaku program (Andleeb et al., 2025), sehingga membekali keterampilan ini menjadi tantangan pedagogis tersendiri.

Requirements Traceability Matrix (RTM) lazim digunakan untuk melatih pengujian sistematis dengan menautkan setiap kebutuhan ke artefak uji yang memverifikasinya (Ruiz et al., 2023). Dalam pembelajaran, RTM berperan sebagai *scaffolding*, yaitu sokongan terstruktur yang membantu pembelajar menyelesaikan tugas yang belum dapat dikerjakan secara mandiri, dan terbukti berkontribusi positif terhadap kemampuan berpikir mahasiswa (Chen et al., 2025). Pendekatan ini kerap dipadukan dengan metode planted bug, yang memungkinkan pengukuran objektif karena *defect* yang ditanam telah diketahui sebelumnya sehingga tingkat deteksi dapat dihitung secara terukur (Delgado-Pérez et al., 2021).

Namun tersimpan sebuah asumsi yang jarang diuji, yaitu bahwa menuntaskan prosedur RTM otomatis berarti mahasiswa mampu mendeteksi *defect*. Padahal kelengkapan prosedural dan efektivitas deteksi adalah dua hal berbeda. Penelitian terdahulu umumnya mengukur efektivitas pada tingkat agregat, baik melalui skor mutation testing (Delgado-Pérez et al., 2021), pemetaan bidang pendidikan pengujian (Garousi et al., 2020), maupun

karakterisasi kualitatif kesulitan mahasiswa (Andleeb et al., 2025) dan penilaian kualitas uji secara keseluruhan (Showler et al., 2025). Belum ada yang memilah tingkat deteksi menurut kategori *defect* pada kondisi penyelesaian prosedur yang seragam, padahal justru pada kondisi itulah variasi efektivitas paling informatif karena tidak lagi dapat dikaitkan dengan perbedaan kelengkapan prosedur. Kesenjangan ini semakin terasa dalam konteks pendidikan tinggi di Indonesia.

Penelitian ini menganalisis tingkat deteksi planted bug berdasarkan kategori *defect* pada sembilan kelompok mahasiswa yang seluruhnya menyelesaikan RTM secara tuntas dan menguji lima aplikasi web dengan himpunan *defect* identik, sehingga perbandingan antar kelompok dan antar kategori dapat dilakukan secara setara. Dua pertanyaan diajukan: pertama, sejauh mana tingkat deteksi bervariasi antar kategori *defect* ketika penyelesaian RTM seragam; dan kedua, kategori *defect* mana yang paling sulit dideteksi serta bagaimana pola tersebut dijelaskan. Kebaruan penelitian terletak pada analisis deteksi yang terpilah menurut kategori pada kondisi RTM seragam, sudut pandang yang belum digarap penelitian terdahulu yang umumnya berhenti pada ukuran agregat. Kontribusinya bersifat ganda: secara teoretis memberikan bukti empiris terpilah bahwa kelengkapan prosedur tidak menjamin efektivitas deteksi pada pembelajar pemula; secara praktis mengarahkan perbaikan desain *scaffolding* agar menekankan kategori *defect* yang sulit dideteksi.

II. TINJAUAN PUSTAKA

2.1 Pengujian Perangkat Lunak dan Klasifikasi Cacat

Pengujian perangkat lunak adalah proses menjalankan program dengan maksud menemukan kesalahan, dan efektivitasnya sangat bergantung pada kemampuan penguji merancang kasus uji yang mampu memicu kegagalan. Pada tataran fungsional, pengujian black box banyak digunakan, termasuk oleh penguji pemula, karena tidak menuntut penguasaan struktur kode internal dan berfokus pada kesesuaian keluaran terhadap spesifikasi (Mujahidin Haqqoni et al., 2024). Kesalahan yang ditemukan tidak seragam sifatnya; untuk dapat dianalisis secara sistematis, *defect* perlu diklasifikasikan menurut jenisnya berdasarkan karakteristik seperti modul, bentuk kesalahan, dan cara kesalahan itu termanifestasi. Klasifikasi semacam ini penting karena tiap jenis *defect* menuntut strategi deteksi yang berbeda: *defect* yang termanifestasi langsung pada keluaran fungsional relatif mudah dikenali, sedangkan *defect* yang hanya muncul pada kondisi batas, skenario negatif, atau aspek non-fungsional menuntut kejelian yang lebih tinggi (Andleeb et al., 2025).

2.2 *Requirements Traceability Matrix* sebagai *Scaffolding* Pembelajaran

Requirements Traceability Matrix (RTM) adalah dokumen yang menautkan setiap kebutuhan ke artefak uji yang memverifikasinya, sehingga menjamin bahwa tidak ada kebutuhan yang luput dari pengujian; *traceability* semacam ini memungkinkan penelusuran dua arah antara kebutuhan dan kasus uji sekaligus mempermudah deteksi kebutuhan yang belum teruji (Ruiz et al., 2023). Ditinjau dari perspektif pedagogis, RTM dapat diposisikan sebagai *scaffolding*, yaitu sokongan terstruktur yang memungkinkan pembelajar menyelesaikan tugas yang belum dapat mereka kerjakan secara mandiri. Kajian mutakhir menunjukkan bahwa *scaffolding*, baik yang bersifat kognitif maupun metakognitif, berkontribusi positif terhadap kemampuan berpikir mahasiswa dalam pembelajaran pemrograman (Chen et al., 2025). Dengan memandu mahasiswa menelusuri setiap kebutuhan menuju kasus uji, RTM diharapkan menumbuhkan kebiasaan pengujian yang menyeluruh dan terlacak.

2.3 Metode Planted Bug untuk Asesmen Deteksi

Untuk mengukur kemampuan deteksi secara objektif, diperlukan tolok ukur *defect* yang telah diketahui sebelumnya. Metode planted bug menanamkan sejumlah *defect* yang telah didefinisikan ke dalam perangkat lunak, sehingga tingkat deteksi dapat dihitung sebagai rasio *defect* yang berhasil ditemukan terhadap *defect* yang ditanam. Pendekatan sejenis, termasuk mutation testing, telah dimanfaatkan dalam pendidikan pengujian untuk menilai sekaligus melatih kemampuan mahasiswa (Delgado-Pérez et al., 2021). Keunggulan pendekatan ini terletak pada objektivitasnya karena himpunan *defect* diketahui, penilaian deteksi tidak bergantung pada interpretasi subjektif.

2.4 Kesenjangan antara Kelengkapan Prosedural dan Efektivitas Deteksi

Meskipun instrumen seperti RTM mendorong kelengkapan prosedur, literatur terbaru menunjukkan bahwa menyelesaikan prosedur tidak dengan sendirinya menjamin efektivitas deteksi. Studi terhadap praktik pengujian mahasiswa menemukan bahwa pembelajar kerap memperlakukan pengujian sebagai daftar centang prosedural serta menunjukkan happy-path bias, yakni fokus pada masukan valid sembari mengabaikan skenario kegagalan dan eksepsi (Andleeb et al., 2025). Untuk memperjelas posisi penelitian ini terhadap literatur yang ada, Tabel 1 merangkum sejumlah penelitian terdahulu beserta metode, temuan utama, dan keterbatasannya.

Tabel 1. Perbandingan penelitian terdahulu

Penulis (tahun)	Fokus & Metode	Temuan Utama	Keterbatasan
Garousi et al. (2020)	Systematic literature mapping pendidikan pengujian	Pengajaran cenderung teoretis; mahasiswa kurang terdorong menemukan *defect*	Tingkat pemetaan; tidak mengukur deteksi per kategori
Delgado-Pérez et al. (2021)	Eksperimen mutation testing + penilaian sejawat	Mahasiswa menguji operasi dasar, mengabaikan fitur rumit	Ukuran agregat; tidak memilah per kategori *defect*
Ruiz et al. (2023)	Wawancara hambatan *traceability* di industri	*Traceability* jarang dijalankan; manfaat deteksi diakui	Konteks industri, bukan pembelajaran mahasiswa
Andleeb et al. (2025)	Studi kualitatif kesulitan mahasiswa	Pengujian sebagai daftar centang; happy-path bias	Kualitatif; tidak menghitung detection rate per kategori
Showler et al. (2025)	Analisis 1014 uji dari 12 kelompok	Efektivitas uji mahasiswa bervariasi	Fokus kualitas uji; bukan deteksi per kategori
Penelitian ini (2026)	**Studi kasus, 9 kelompok, planted bug, matching ketat**	**Detection 55,7% meski RTM 100%; *observable* vs *non-observable* signifikan**	**Sampel kecil satu institusi**

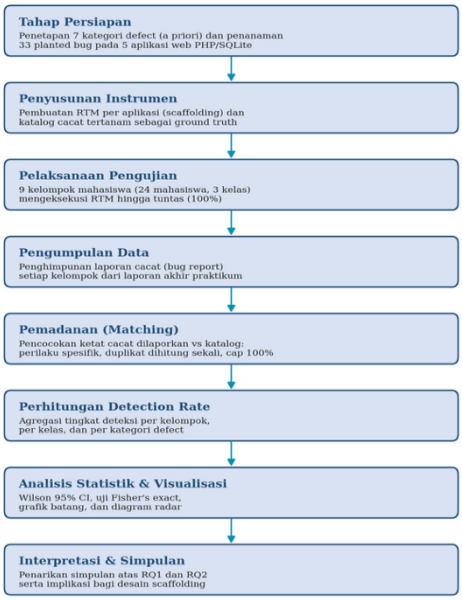
Sintesis atas kelima studi tersebut memperlihatkan satu pola yang konsisten. Kajian terdahulu mengukur efektivitas pengujian mahasiswa pada tingkat agregat, baik berupa skor mutation (Delgado-Pérez et al., 2021), pemetaan tematik atas bidang (Garousi et al., 2020), karakterisasi kualitatif kesulitan mahasiswa (Andleeb et al., 2025), maupun penilaian kualitas uji secara keseluruhan (Showler et al., 2025), sementara riset *traceability* lebih menyoroti praktik industri daripada konteks pembelajaran (Ruiz et al., 2023). Belum ada yang mengukur tingkat deteksi *defect* secara terpilah menurut kategori pada kondisi ketika penyelesaian prosedur pengujian ditahan seragam. Justru pada kondisi seragam itulah variasi efektivitas deteksi menjadi paling informatif, karena perbedaan hasil tidak dapat lagi dikaitkan dengan perbedaan kelengkapan prosedur. Kesenjangan inilah yang diisi penelitian ini.

III. METODE PENELITIAN

3.1 Desain Penelitian

Penelitian ini menggunakan pendekatan studi kasus deskriptif dengan rancangan banyak-kasus (multiple-case design). Pendekatan studi kasus dipilih karena penelitian ini mengamati fenomena dalam konteks pembelajaran nyata tanpa melakukan manipulasi

terhadap variabel, dan bertujuan memahami pola deteksi secara mendalam alih-alih menggeneralisasi secara statistik (Yin, 2018; Runeson & Höst, 2009). Fenomena yang diamati adalah efektivitas deteksi *defect* oleh mahasiswa dalam kondisi ketika prosedur pengujian, yang direpresentasikan melalui penyelesaian RTM, telah dituntaskan secara seragam oleh seluruh unit analisis. Dengan menahan penyelesaian prosedur pada kondisi konstan, rancangan ini memungkinkan variasi pada efektivitas deteksi menjadi teramati secara jelas. Alur penelitian secara keseluruhan disajikan pada Gambar 1.



Gambar 1. Diagram alur penelitian.

3.2 Subjek dan Konteks

Unit analisis penelitian ini adalah sembilan kelompok mahasiswa program studi Sistem Informasi jenjang sarjana pada sebuah perguruan tinggi di Indonesia, yang mengikuti kegiatan praktikum pengujian perangkat lunak pada semester genap tahun akademik 2025/2026. Kesembilan kelompok tersebut beranggotakan total 24 mahasiswa dan tersebar pada tiga kelas paralel dengan karakteristik berbeda, yaitu kelas reguler, kelas pekerja, dan satu kelas berukuran kecil. Rincian jumlah mahasiswa tiap kelompok beserta objek uji yang dikerjakan disajikan pada Tabel 2. Perbedaan karakteristik kelas tidak diperlakukan sebagai variabel bebas yang dikontrol, melainkan sebagai konteks alami yang menyertai tiap kasus. Dua kelompok tambahan yang semula ada dikeluarkan dari analisis karena laporannya tidak lengkap sehingga tidak memenuhi syarat keterandalan data.

Tabel 2. Karakteristik subjek dan objek uji

Kelompok	Kelas	Karakteristik	Jumlah Mahasiswa	Aplikasi (planted bug)
A-K1	A	Reguler	3	TokoKu (7)
A-K2	A	Reguler	3	KosKu (8)
A-K3	A	Reguler	3	KliniKu (6)
A-K5	A	Reguler	3	KantinKu (6)
A-K6	A	Reguler	2	AbsenKu (6)
B-K1	B	Pekerja	3	TokoKu (7)
B-K2	B	Pekerja	2	KosKu (8)
B-K3	B	Pekerja	3	KliniKu (6)
C-K1	C	Kelas kecil	2	TokoKu (7)
Total	**3 kelas**		**24 mahasiswa**	**5 aplikasi, 33 *defect***

3.3 Objek Uji dan Cacat Tertanam

Objek uji berupa lima aplikasi web sederhana berbasis PHP dan SQLite yang mewakili domain berbeda. Tujuh kategori *defect* ditetapkan terlebih dahulu (a priori) berdasarkan jenis kesalahan yang lazim ditemui pada aplikasi web, yaitu validasi formulir, otentikasi/sesi, logika kueri basis data, aturan bisnis, logika perhitungan, umpan balik antarmuka, dan pengalihan halaman. Selanjutnya, untuk setiap aplikasi, sejumlah *defect* dirancang dan ditanamkan sedemikian rupa sehingga mewakili kategori-kategori tersebut, dengan lokasi berkas, bentuk kesalahan, dan prosedur verifikasi yang seluruhnya didokumentasikan dalam sebuah katalog *defect*. Himpunan *defect* yang ditanamkan identik untuk setiap aplikasi lintas kelas, sehingga perbandingan antar kelompok yang menguji aplikasi yang sama menjadi setara. Secara keseluruhan terdapat 33 *defect* unik pada kelima aplikasi. Distribusi *defect* per kategori disajikan pada Tabel 3.

Tabel 3. Distribusi planted bug per kategori *defect*

Kategori *Defect*	Jumlah *Defect* Unik
Validasi Formulir	10
Otentikasi/Sesi	6
Logika Kueri Basis Data	6
Aturan Bisnis	3
Logika Perhitungan	3
Umpan Balik Antarmuka	3
Pengalihan Halaman	2
Total	**33**

3.4 Instrumen

Dua instrumen utama digunakan. Pertama, RTM yang disediakan untuk setiap aplikasi, memuat kasus uji yang menautkan tiap kebutuhan fungsional ke skenario pengujiannya; RTM ini berfungsi sebagai *scaffolding* prosedural yang memandu mahasiswa mengeksekusi pengujian secara menyeluruh. Validitas isi RTM dijaga dengan menurunkan setiap kasus uji langsung dari kebutuhan fungsional aplikasi dan memastikan setiap kebutuhan tertaut ke sekurang-kurangnya satu kasus uji. Kedua, katalog *defect* tertanam (answer key) yang memuat seluruh planted bug beserta kategori, lokasi berkas, dan prosedur verifikasinya. Katalog inilah yang menjadi rujukan kebenaran (ground truth) dalam menilai apakah suatu *defect* yang dilaporkan mahasiswa benar-benar merupakan *defect* yang ditanamkan.

3.5 Pengumpulan Data

Data deteksi *defect* diperoleh dari laporan akhir praktikum tiap kelompok, khususnya bagian laporan *defect* (bug report) yang mencantumkan modul, deskripsi, dan klasifikasi tiap *defect* yang ditemukan. Data ini kemudian dipadankan dengan katalog *defect* tertanam untuk menentukan status deteksi setiap planted bug pada tiap kelompok.

3.6 Analisis Data

Analisis dilakukan melalui pemadanan (matching) antara *defect* yang dilaporkan tiap kelompok dan katalog *defect* tertanam, dengan aturan yang ditetapkan di awal untuk menjaga keterukuran (construct validity). Aturan tersebut bersifat ketat: sebuah *defect* yang dilaporkan dinyatakan mendeteksi suatu planted bug hanya apabila perilaku kesalahan yang dideskripsikan bersesuaian secara spesifik dengan entri pada katalog; laporan ganda atas planted bug yang sama dihitung satu kali; dan *defect* yang nyata tetapi berada di luar

himpunan *defect* tertanam (temuan organik) tidak dihitung sebagai deteksi planted bug. Pada kasus ketika jumlah *defect* yang dilaporkan melampaui jumlah *defect* yang ditanamkan akibat temuan organik, nilai deteksi dibatasi pada 100% untuk menjaga konsistensi metrik antar unit. Pemadanan dilakukan oleh satu penilai, yaitu peneliti; untuk mengurangi subjektivitas, setiap keputusan pemadanan yang ambigu diselesaikan secara konservatif dan seluruh keputusan didokumentasikan dalam jejak audit yang dapat ditelusuri ulang.

Tingkat deteksi dihitung sebagai rasio planted bug yang berhasil dideteksi terhadap total planted bug, dan diagregasi pada tiga tingkat: per kelompok, per kelas, dan per kategori *defect*. Untuk mengukur ketidakpastian estimasi pada ukuran sampel yang kecil, digunakan interval kepercayaan (IK) Wilson 95%, yang lebih andal daripada interval normal untuk proporsi dengan jumlah pengamatan kecil (Brown et al., 2001). Selanjutnya, perbedaan tingkat deteksi antara *defect* yang bersifat *observable* dan *non-observable* diuji menggunakan uji Fisher's exact, yang sahih untuk tabel kontingensi dengan frekuensi sel kecil. Uji inferensial berbasis distribusi seperti chi-square maupun ANOVA tidak digunakan karena asumsi ukuran sel dan sampelnya tidak terpenuhi pada data ini, sehingga penggunaannya justru berpotensi menghasilkan simpulan yang menyesatkan.

3.7 Ancaman terhadap Validitas

Beberapa ancaman terhadap validitas perlu dinyatakan secara terbuka (Runeson & Höst, 2009). Pada validitas konstruk, risiko kekeliruan memadankan *defect* dimitigasi melalui aturan pemadanan ketat dan pembatasan nilai deteksi; karena pemadanan dilakukan satu penilai, potensi subjektivitas dikelola lewat aturan awal dan jejak audit, namun tetap diakui sebagai ruang perbaikan. Pada validitas internal, terdapat kemungkinan pembauran antara karakteristik kelas dan aplikasi karena aplikasi tidak tersebar merata pada ketiga kelas. Pada validitas eksternal, jumlah unit analisis yang kecil dan berasal dari satu institusi membatasi generalisasi, sehingga hasil bersifat kontekstual. Pada keterandalan, satu kelompok pada kelas kecil memiliki kelengkapan laporan terbatas dan sejumlah kategori memiliki instance sedikit, sehingga interpretasinya dilakukan berhati-hati.

IV. HASIL DAN PEMBAHASAN

4.1 Tingkat Deteksi per Kelompok dan per Kelas

Kesembilan kelompok menyelesaikan seluruh kasus uji dalam RTM yang disediakan, sehingga tingkat penyelesaian prosedur pengujian seragam sebesar 100% pada semua unit analisis. Meskipun demikian, tingkat deteksi *defect* yang sebenarnya, yaitu proporsi planted bug yang berhasil dikenali, bervariasi secara mencolok antar kelompok, berkisar dari 14,3% hingga 87,5%. Rincian tingkat deteksi tiap kelompok disajikan pada Tabel 4.

Tabel 4. Tingkat deteksi planted bug per kelompok

Kelompok	Aplikasi	Kelas	Terdeteksi	Planted	Detection Rate
A-K1	TokoKu	A	4	7	57,1%
B-K1	TokoKu	B	5	7	71,4%
C-K1	TokoKu	C	1	7	14,3%
A-K2	KosKu	A	6	8	75,0%
B-K2	KosKu	B	7	8	87,5%
A-K3	KliniKu	A	2	6	33,3%
B-K3	KliniKu	B	2	6	33,3%
A-K5	KantinKu	A	4	6	66,7%
A-K6	AbsenKu	A	3	6	50,0%
Total			**34**	**61**	**55,7%**

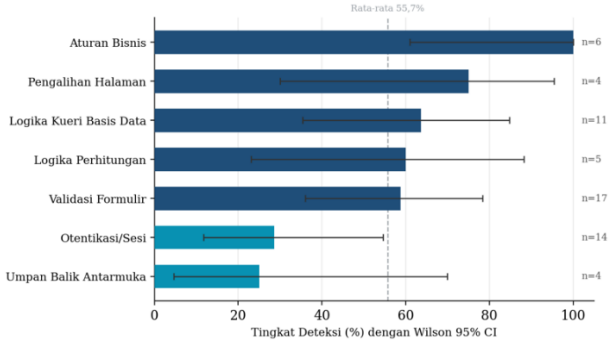
Secara agregat, dari 61 instance planted bug yang diuji lintas seluruh kelompok, sebanyak 34 berhasil dideteksi, menghasilkan tingkat deteksi keseluruhan sebesar 55,7% (IK 95% Wilson: 43,3%-67,5%). Bila diagregasi menurut kelas, kelas reguler memperoleh 57,6% (19 dari 33) dan kelas pekerja 66,7% (14 dari 21), sedangkan kelas berukuran kecil 14,3% (1 dari 7). Sekilas kelas pekerja tampak lebih unggul daripada kelas reguler, tetapi perbedaan ini tidak signifikan secara statistik (uji Fisher's exact, p = 0,575) dan interval kepercayaannya sangat bertumpang tindih (pekerja 45,4%-82,8%; reguler 40,8%-72,8%). Dengan demikian, selisih tersebut lebih mungkin mencerminkan variasi acak daripada perbedaan kemampuan yang nyata. Dugaan bahwa pengalaman kerja mahasiswa kelas pekerja turut berperan tetap terbuka untuk diteliti, tetapi tidak dapat disimpulkan dari data ini. Temuan ini mengarahkan perhatian dari perbedaan antar kelas menuju sumber variasi yang lebih menjelaskan, yaitu jenis *defect*-nya.

4.2 Tingkat Deteksi per Kategori Cacat

Ketika data yang sama diagregasi menurut kategori *defect*, muncul pola yang jauh lebih teratur dibandingkan agregasi per kelas. Tingkat deteksi bergradien tajam dari kategori yang hampir selalu terdeteksi hingga kategori yang sebagian besar terlewat, sebagaimana disajikan pada Tabel 5 dan divisualisasikan pada Gambar 2. Setiap tingkat deteksi disertai interval kepercayaan Wilson 95% untuk menunjukkan derajat ketidakpastiannya.

Tabel 5. Tingkat deteksi per kategori *defect* dengan interval kepercayaan Wilson 95%

Kategori *Defect*	Terdeteksi	Total	Detection Rate	IK 95% Wilson
Aturan Bisnis	6	6	100,0%	61,0%-100,0%
Pengalihan Halaman	3	4	75,0%	30,1%-95,4%
Logika Kueri Basis Data	7	11	63,6%	35,4%-84,8%
Logika Perhitungan	3	5	60,0%	23,1%-88,2%
Validasi Formulir	10	17	58,8%	36,0%-78,4%
Otentikasi/Sesi	4	14	28,6%	11,7%-54,6%
Umpan Balik Antarmuka	1	4	25,0%	4,6%-69,9%
Total	**34**	**61**	**55,7%**	**43,3%-67,5%**

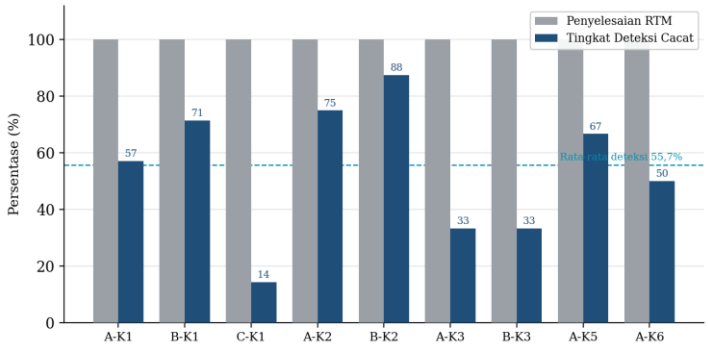


Gambar 2. Tingkat deteksi per kategori *defect* dengan interval kepercayaan Wilson 95%

Kategori Aturan Bisnis, Pengalihan Halaman, Logika Kueri Basis Data, Logika Perhitungan, dan Validasi Formulir berada pada atau di atas kisaran tengah (58,8%-100%), sedangkan kategori Otentikasi/Sesi (28,6%) dan Umpan Balik Antarmuka (25,0%) berada jauh di bawahnya. Lebar interval kepercayaan memperjelas bahwa sebagian kategori dengan jumlah instance kecil, seperti Umpan Balik Antarmuka (IK 4,6%-69,9%), memiliki estimasi yang sangat tidak pasti sehingga harus dimaknai dengan hati-hati. Sebaliknya, kategori dengan jumlah instance memadai, yaitu Validasi Formulir (17), Otentikasi/Sesi (14), dan Logika Kueri Basis Data (11), memiliki interval yang lebih sempit dan memberikan dasar yang lebih kokoh untuk penarikan simpulan.

4.3 Disosiasi antara Penyelesaian Prosedur dan Efektivitas Deteksi

Menjawab pertanyaan penelitian pertama, hasil ini menunjukkan bahwa tingkat deteksi *defect* tetap bervariasi secara substansial meskipun penyelesaian RTM seragam sempurna pada seluruh kelompok. Penyelesaian prosedur sebesar 100% berdampingan dengan tingkat deteksi keseluruhan yang hanya 55,7%, sebuah selisih yang memperlihatkan bahwa menuntaskan langkah-langkah pengujian tidak identik dengan menemukan *defect*. Disosiasi ini divisualisasikan pada Gambar 3, yang memperlihatkan garis penyelesaian RTM yang datar sempurna berlawanan dengan batang deteksi yang naik-turun antar kelompok.



Gambar 3. Perbandingan penyelesaian RTM dan tingkat deteksi *defect* per kelompok

Temuan ini sejalan dengan pengamatan bahwa pembelajar cenderung memperlakukan pengujian sebagai daftar centang prosedural alih-alih upaya memvalidasi perilaku program (Andleeb et al., 2025). RTM sebagai *scaffolding* berhasil menjamin bahwa mahasiswa mengeksekusi seluruh kasus uji, namun keberhasilan prosedural itu tidak dengan sendirinya berpindah menjadi kejelian mengenali penyimpangan perilaku. Dengan kata lain, *scaffolding* prosedural memenuhi fungsinya pada tataran kelengkapan, tetapi tidak memadai untuk menumbuhkan efektivitas deteksi secara merata di seluruh jenis *defect*.

4.4 Perbedaan Deteksi *Defect Observable* dan *Non-observable*

Untuk menguji apakah gradien antar kategori mencerminkan pola yang lebih mendasar, kategori dikelompokkan menjadi dua berdasarkan cara *defect* termanifestasi. Kelompok *observable* mencakup *defect* yang penyimpangannya kasat mata pada alur normal (Aturan Bisnis, Pengalihan Halaman, Logika Kueri Basis Data, Logika Perhitungan, dan Validasi Formulir), sedangkan kelompok *non-observable* mencakup *defect* yang umumnya hanya tersingkap melalui pengujian negatif atau perhatian pada aspek non-fungsional (Otentikasi/Sesi dan Umpan Balik Antarmuka). *Defect observable* terdeteksi pada 67,4% kasus (29 dari 43; IK 95% 52,5%-79,5%), jauh lebih tinggi daripada *defect non-observable* yang hanya 27,8% (5 dari 18; IK 95% 12,5%-50,9%). Uji Fisher's exact menunjukkan perbedaan ini signifikan secara statistik (odds ratio = 5,39; p = 0,010), dan kedua interval kepercayaan tidak saling bertumpang tindih. Dengan demikian, *defect non-observable* sekitar lima kali lebih kecil kemungkinannya untuk terdeteksi dibandingkan *defect* observable, dan perbedaan ini tidak dapat dijelaskan sekadar sebagai kebetulan.

4.5 Pola Kategori dan Penjelasannya

Menjawab pertanyaan penelitian kedua, kategori *defect* yang paling sulit dideteksi adalah Otentikasi/Sesi dan Umpan Balik Antarmuka. Pola ini dapat dijelaskan melalui karakteristik cara *defect* termanifestasi. *Defect* pada kategori Aturan Bisnis dan Pengalihan Halaman umumnya menghasilkan penyimpangan perilaku yang kasat mata ketika aplikasi dijalankan pada alur normal, sehingga mudah dikenali bahkan tanpa perancangan kasus uji yang canggih. Sebaliknya, *defect* Otentikasi/Sesi umumnya hanya tersingkap melalui

pengujian skenario negatif, seperti mengakses halaman terproteksi tanpa sesi yang sah atau memanipulasi identitas pengguna, jenis pengujian yang menuntut penguji membayangkan cara sistem disalahgunakan, bukan sekadar cara sistem digunakan. Demikian pula, *defect* Umpan Balik Antarmuka bersifat non-fungsional dan tidak menghentikan jalannya aplikasi, sehingga mudah diabaikan oleh penguji yang berfokus pada keberhasilan fungsi utama.

Temuan ini memperkuat sekaligus memperinci hasil terdahulu. Kecenderungan mengabaikan skenario kegagalan konsisten dengan happy-path bias (Andleeb et al., 2025), dan pola menguji operasi dasar sembari mengabaikan fitur rumit juga dilaporkan pada penilaian berbasis mutation testing (Delgado-Pérez et al., 2021). Namun, berbeda dengan kedua studi yang berhenti pada karakterisasi kualitatif atau skor agregat, penelitian ini menunjukkan secara terkuantifikasi bahwa kelemahan itu terkonsentrasi pada kategori *defect non-observable* dan signifikan secara statistik, sehingga mencerminkan keterbatasan kognitif yang tidak teratasi oleh *scaffolding* prosedural semata.

4.6 Kasus Kelompok dengan Deteksi Terendah

Satu kelompok pada kelas berukuran kecil mencatat tingkat deteksi terendah, yaitu 14,3% (1 dari 7). Temuan ini perlu dimaknai dengan kehati-hatian ganda. Pertama, unit ini hanya terdiri atas satu kelompok dengan dua mahasiswa, sehingga tidak dapat ditafsirkan sebagai karakteristik kelas kecil secara umum. Kedua, laporan kelompok ini memiliki kelengkapan terbatas, dengan sebagian tabel eksekusi uji tidak terisi, sehingga sebagian *defect* yang mungkin ditemukan tidak terdokumentasi dan angka deteksinya berpotensi menjadi batas bawah, bukan nilai pasti. Kelompok ini karenanya diperlakukan sebagai kasus ilustratif mengenai keterbatasan data, bukan sebagai bukti pola.

4.7 Implikasi

Secara teoretis, penelitian ini memberikan bukti empiris terpilah bahwa kelengkapan prosedur tidak menjamin efektivitas deteksi pada pembelajar pemula, dan kesenjangan itu terkonsentrasi sistematis pada *defect non-observable*. Secara praktis, desain *scaffolding* perlu diperkaya dengan penekanan eksplisit pada kategori sulit: RTM sebaiknya dilengkapi panduan atau kasus uji yang mengarahkan mahasiswa pada pengujian negatif, manipulasi sesi, dan verifikasi umpan balik non-fungsional, yakni jenis pengujian yang tidak muncul alamiah dari alur normal. Dengan demikian *scaffolding* tidak hanya berorientasi pada penyelesaian prosedur, tetapi juga pada penumbuhan kemampuan berpikir kritis mahasiswa dalam mendeteksi beragam kategori *defect*.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini menganalisis tingkat deteksi planted bug menurut kategori *defect* pada sembilan kelompok mahasiswa yang seluruhnya menyelesaikan RTM secara tuntas. Menjawab pertanyaan penelitian pertama, ditemukan bahwa tingkat deteksi *defect* tetap bervariasi secara substansial, dari 14,3% hingga 87,5% antar kelompok, dengan rata-rata keseluruhan 55,7%, meskipun penyelesaian prosedur pengujian seragam sempurna sebesar 100%. Temuan ini menegaskan adanya disosiasi antara kelengkapan prosedural dan efektivitas deteksi. Menjawab pertanyaan penelitian kedua, kategori *defect* yang paling sulit dideteksi adalah Otentikasi/Sesi (28,6%) dan Umpan Balik Antarmuka (25,0%); secara lebih umum, *defect non-observable* secara signifikan lebih sulit dideteksi daripada *defect observable* (67,4% berbanding 27,8%; $p = 0,010$). Kontribusi teoretis penelitian ini adalah bukti empiris terpilah menurut kategori bahwa *scaffolding* prosedural seperti RTM memadai untuk menjamin kelengkapan pengujian tetapi belum cukup untuk menumbuhkan kejelian

deteksi secara merata, sedangkan kontribusi praktisnya adalah arahan bagi perancangan *scaffolding* yang menekankan kategori *defect* sulit.

5.2 Saran

Berdasarkan temuan tersebut, disarankan agar desain *scaffolding* pengujian tidak berhenti pada penjaminan kelengkapan prosedur, melainkan diperkaya dengan panduan dan kasus uji yang secara eksplisit mengarahkan mahasiswa pada pengujian negatif, manipulasi sesi, dan verifikasi umpan balik non-fungsional. Penelitian ini memiliki keterbatasan yang membuka arah kajian lanjutan: jumlah unit analisis yang kecil dan berasal dari satu institusi membatasi generalisasi, pemadanan dilakukan oleh satu penilai, dan sebagian kategori *defect* memiliki jumlah instance yang sedikit. Penelitian selanjutnya disarankan melibatkan jumlah kelompok dan institusi yang lebih besar dengan sebaran aplikasi yang seimbang, menerapkan pemadanan oleh lebih dari satu penilai disertai pengukuran kesepakatan antar-penilai seperti koefisien Cohen's kappa, serta menambahkan data proses seperti rekaman aktivitas pengujian untuk menelaah bukan hanya *defect* mana yang terlewat, tetapi juga mengapa *defect* tersebut terlewat pada tingkat perilaku penguji.

DAFTAR PUSTAKA

Andleeb, S., Mendoza, T., Cordova, L., Walia, G., & Carver, J. (2025). Enhancing software testing education: Understanding where students struggle. Proceedings of the 26th ACM Annual Conference on Cybersecurity & Information Technology Education, 154–160. https://doi.org/10.1145/3769694.3771127

Chen, D., Zhang, Y., Luo, H., Gao, Z., Yu, L., & Lin, Y. (2025). Exploring the impact of *scaffolding* in programming on students' computational thinking: Evidence from a three-level meta-analysis. Journal of Educational Computing Research. https://doi.org/10.1177/07356331251386618

Delgado-Pérez, P., Medina-Bulo, I., Álvarez-García, M. Á., & Valle-Gómez, K. J. (2021). Mutation testing and self/peer assessment: Analyzing their effect on students in a software testing course. 2021 IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).

Garousi, V., Felderer, M., & Mäntylä, M. V. (2020). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, 106, 101–121. https://doi.org/10.1016/j.infsof.2018.09.006

Mujahidin Haqqoni, B., Winarno, I., Naufal Musthofa, M., Sakdi, M., & Saifudin, A. (2024). Pengujian fungsional perangkat lunak sistem informasi perpustakaan dengan metode blackbox testing bagi pemula. LOGIC: Jurnal Ilmu Komputer dan Pendidikan, 2(4), 696–704.

Ruiz, M., Hu, J. Y., & Dalpiaz, F. (2023). Why don't we trace? A study on the barriers to software *traceability* in practice. Requirements Engineering, 28(4), 619–637. https://doi.org/10.1007/s00766-023-00408-9

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Showler, A., Miljanovic, M. A., & Bradbury, J. S. (2025). How effective and efficient are student-written software tests? Proceedings of the 56th ACM Technical Symposium on Computer Science Education V.1, 1057–1063. https://doi.org/10.1145/3641554.3701809

Yin, R. K. (2018). Case study research and applications: Design and methods (6th ed.). SAGE.