

Pengaruh Refactoring Extract Method terhadap Pengembangan Aplikasi menggunakan Test Driven Development

1st Fauzi Hazim Wibowo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia

fauzihzam@students.telkomuniversity.ac.id

2nd Dawam Dwi Jatmiko Suwawi
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia

dawamdjs@telkomuniversity.ac.id

3rd Anisa Herdiani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia

anisaherdiani@telkomuniversity.ac.id

Abstrak — Tingginya kompleksitas dan rendahnya maintainability pada kode menyebabkan maintain sebuah program sulit untuk dilakukan. Maintainability dan readability saling berkaitan karena rendahnya maintainability menyebabkan kode sulit untuk dibaca dan dimodifikasi. Menurunkan kompleksitas, meningkatkan maintainability, dan meningkatkan readability merupakan tujuan refactoring pada test driven development. Refactoring dengan extract method dipilih karena dapat meningkatkan readability dan mengurangi duplikasi pada kode. Pengembangan website pada penelitian ini menggunakan paradigma pemrograman functional programming dan mengalami permasalahan long method. Metode refactoring ini dapat menghilangkan long method pada paradigma pemrograman functional programming sehingga sesuai diterapkan pada penelitian ini. Test driven development merupakan pengembangan perangkat lunak yang didasari oleh pembuatan program pengujian iteratif otomatis kecil, penulisan kode untuk lolos testing, dan refactoring code. Penelitian ini membuat website penilaian e-learning readiness Hung model berdasarkan requirement dari kaprodi S1 PJJ Informatika menggunakan test driven development. Pengembangan website ini dikerjakan oleh satu tim dan memiliki anggaran yang kecil. Oleh karena itu, penelitian ini sesuai dengan metode pengembangan perangkat lunak test driven development yang memungkinkan pengembangan perangkat lunak dengan satu tim dan anggaran yang kecil. Website ini diteliti dan dianalisis terkait pengaruh extract method terhadap cyclomatic complexity, halstead volume, maintainability index, dan code readability prediction pada pengembangan menggunakan test driven development.

Kata kunci — test driven development, extract method, cyclomatic complexity, halstead volume, maintainability index, code readability prediction.

I. PENDAHULUAN

A. Latar Belakang

E-learning readiness merupakan faktor-faktor yang harus diperhatikan oleh institusi, pengajar, dan mahasiswa agar implementasi *e-learning* berhasil [1]. Selain itu, *e-learning readiness* juga bertujuan untuk mengetahui faktor-faktor penghambat kesuksesan dalam pelaksanaan *e-learning* [2]. Salah satu model pada *e-learning readiness* pada mahasiswa adalah model Hung. Model ini membagi *e-learning readiness* ke beberapa

matrik, antara lain : *computer/internet self-efficacy*, *self-directed learning*, *learner control*, *motivation for learning*, dan *online communication self-efficacy* [3].

Test driven development merupakan pengembangan perangkat lunak yang didasari oleh pembuatan program pengujian iteratif otomatis kecil, penulisan kode untuk lolos testing, dan *refactoring code* [4]. Metode ini cocok digunakan pada penelitian ini karena dapat dikerjakan dengan satu tim dan anggaran yang kecil. Tahap *refactoring* pada *test driven development* akan menyempurnakan kode dengan mengurangi duplikasi sehingga dapat mengurangi kompleksitas, meningkatkan readability, dan meningkatkan maintainability pada kode [5]. *Extract method* mengubah bagian kode menjadi *method* [6]. Metode *extract method* dipilih karena memiliki keunggulan dapat meningkatkan readability dan mengurangi duplikasi pada kode [7]. Rendahnya readability pada penelitian ini karena menggabungkan banyak logika pada satu skrip (long method) sehingga refactoring extract method digunakan pada penelitian ini. Metode refactoring ini dapat menghilangkan *long method* pada paradigma pemrograman *functional programming*. *Long method* terjadi karena menggabungkan banyak logika pada satu skrip [8].

Pengaruh extract method pada pengembangan *test driven development* diujikan dengan membandingkan hasil penghitungan *cyclomatic complexity*, *halstead volume*, *maintainability index*, dan *code readability prediction* sebelum maupun sesudah *refactoring* dengan studi kasus *website e-learning readiness model Hung* pada S1 PJJ Informatika. *Cyclomatic complexity* dan *halstead volume* digunakan untuk menguji matrik kompleksitas. *Maintainability index* sebagai matrik pengukuran *maintainability* pada kode. Sedangkan, *code readability prediction* untuk matrik readability pada kode. Pengaruh *extract method* ini diperoleh dengan menghitung persentase penurunan *cyclomatic complexity*, penurunan *halstead volume*, kenaikan *maintainability index*, dan kenaikan *code readability prediction* pada *function* sebelum dan sesudah *refactoring*. Hasil perhitungan tersebut kemudian dihitung rata-ratanya. Apabila mendapatkan persentase rata-rata yang positif pada penurunan *cyclomatic*

complexity, penurunan *halstead volume*, kenaikan *maintainability index*, dan kenaikan *code readability prediction* maka penerapan *refactoring extract method* sesuai terhadap pengembangan *test driven development*. Pengaruh *extract method* terhadap *cyclomatic complexity*, *halstead volume*, *maintainability index*, dan *code readability prediction* terhadap *test driven development* belum pernah dibahas di jurnal sehingga dibahas pada penelitian ini.

B. Topik dan Batasannya

Belum terdapat jurnal yang membahas tentang pengaruh *extract method* terhadap *cyclomatic complexity*, *halstead volume*, *maintainability index*, dan *code readability prediction* pada pengembangan perangkat lunak menggunakan metode *test driven development* merupakan alasan diambilnya topik ini dengan studi kasus program *Telkom University E-Readiness Survey For S1 PJJ Informatika (TUNERS)*. Penelitian ini menghitung 4 hal tersebut pada program baik sebelum maupun sesudah *refactoring* dengan metode *extract method*. *Refactoring* pada *test driven development* berfungsi untuk menyempurnakan kode dengan mengurangi duplikasi sehingga dapat mengurangi kompleksitas, meningkatkan *readability*, dan meningkatkan *maintainability* pada kode sehingga 4 hal tersebut digunakan.

Batasan masalah dari penelitian ini adalah hanya menguji dan menyimpulkan dari 4 hal tersebut berdasarkan studi kasus TUNERS. Selain itu, juga membuat *website e-learning readiness model Hung* yang sesuai dengan program studi S1 PJJ Informatika.

C. Tujuan

Tujuan dari penelitian ini adalah membangun TUNERS menggunakan *test driven development* berdasarkan *requirement* yang diberikan oleh Kepala Program Studi S1 PJJ Informatika. *Website* ini digunakan sebagai studi kasus untuk diteliti dan mengidentifikasi pengaruh *extract method* terhadap *cyclomatic complexity*, *halstead volume*, *maintainability index*, dan *code readability prediction* pada pengembangan perangkat lunak dengan *test driven development*.

II. KAJIAN TEORI

A. E-learning Readiness Model Hung

Menurut Odunaike, dkk [1] mendefinisikan *e-learning readiness* sebagai faktor-faktor yang harus diperhatikan agar implementasi *e-learning* berhasil. Faktor-faktor yang terdapat dari model tersebut dapat digunakan untuk mengukur tingkat keberhasilan implementasi *e-learning*. Selain itu, *e-learning readiness* juga bertujuan untuk mengetahui faktor penghambat kesuksesan pelaksanaan *e-learning* [2]. Faktor yang menghambat implementasi *e-learning* dapat diketahui dari hasil penilaian sehingga diketahui faktor yang menghambat implementasinya. Setelah, diketahui faktor penghambat implementasi *e-learning*, institusi dapat

menyusun keputusan untuk meningkatkan *e-learning readiness*.

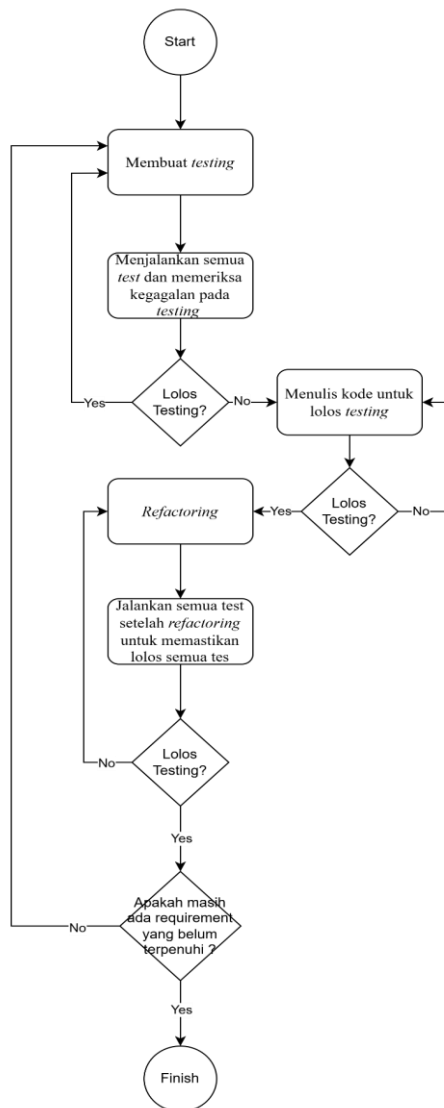
Model Hung merupakan salah satu model pada *e-learning readiness*. Model tersebut digunakan oleh Cesario [3] pada penelitiannya terhadap Perancangan Sistem dan Analisis *e-Learning Readiness* Mahasiswa pada Mahasiswa S1 PJJ Informatika Universitas Telkom dapat dilihat pada tabel 1.

TABEL 1.
Tabel model Hung [3]

Komponen	Deskripsi
<i>Computer/internet self-efficacy</i>	Penilaian terhadap kemampuan seseorang dalam menggunakan komputer [9].
<i>Self-directed learning</i>	Merupakan kemampuan individu menentukan kebutuhan belajar, merumuskan tujuan, menentukan strategi belajar, dan mengevaluasi hasil pembelajarannya [10].
<i>Learner control</i>	Merupakan tingkat kemampuan individu untuk mengarahkan proses pembelajarannya sendiri [11].
<i>Motivation for learning</i>	Kesiapan mahasiswa untuk terbuka terhadap ide-ide baru, motivasi untuk belajar, memperbaiki kesalahan, dan mendiskusikannya dengan orang lain [3].
<i>Online communication Self-efficacy</i>	Kemampuan individu dalam berkomunikasi dengan dosen dan teman, serta mengunggah pertanyaan, dan juga mengekspresikan perasaan dan pikiran secara <i>online</i> [12].

B. Test Driven Development

Test driven development (TDD) merupakan metode didasari oleh pembuatan program pengujian iteratif otomatis kecil, penulisan kode untuk lolos *testing*, dan peningkatan kode [4]. Metode ini terdiri dari 3 fase pada TDD, yaitu : menulis *test case* (red), memastikan lolos *test* (green), *refactoring* (blue) [13]. Berikut tahapan dari TDD.



GAMBAR 1. Proses TDD [14]

Tahapan refactoring pada TDD memiliki keunggulan menyempurnakan kode dengan mengurangi duplikasi sehingga dapat mengurangi kompleksitas, meningkatkan readability, dan meningkatkan maintainability pada kode [5].

C. Extract Method

Extract Method (XM) memiliki keunggulan dapat meningkatkan readability dan mengurangi duplikasi pada kode [7]. XM mengubah bagian kode menjadi method [6]. Hal tersebut dilakukan dengan memindahkan kode dari method lama ke method baru [7].

Metode ini dapat digunakan ketika program dengan paradigma pemrograman functional programming mengalami masalah long method. Long method terjadi karena menggabungkan banyak logika pada satu skrip [8]. XM membagi kode pada method baru.

D. Software Complexity

Software Complexity (SC) merupakan tingkat kesulitan memahami dan bekerja pada suatu program [15]. Terdapat penelitian yang menyatakan korelasi antara SC dengan Software Maintainability dan Cost. Penelitian ini menyatakan semakin tinggi tingkat SC maka semakin sulit untuk memahami kode tersebut sehingga membutuhkan pegawai yang lebih banyak yang menyebabkan meningkatnya cost [16].

E. Cyclomatic Complexity

Cyclomatic Complexity (CC) merupakan sebuah matrik untuk mengukur kompleksitas dari sebuah program [17]. CC mengukur kompleksitas dengan menggunakan Control Flow Graph (CFG). CFG tersebut diperoleh dengan membuat node dari setiap statement dan menjumlahkan jumlah node tersebut, kemudian menyambungkan setiap node dengan edge berdasarkan alur atau flow tersebut [17]. Persamaan dari CC dapat dilihat pada persamaan 1.

$$CC = E - n + 2 \quad (1)$$

Keterangan :

CC = Cyclomatic complexity

E = Edge

n = Node

Hasil dari persamaan tersebut kemudian menghasilkan nilai yang diklasifikasikan. Berikut pengklasifikasiannya dapat dilihat pada tabel 2.

TABEL 2. Klasifikasi CC [17,18]

CC	Rating
1 - 4	Very low
5 - 10	Low
11 - 20	Nominal
21 - 40	High
41 - 50	Very high
> 50	Extra high

Pada penelitian tentang klasifikasi CC di atas pada studi kasus COCOMO II membandingkan CC terhadap judgment dari 3 expert. Menghasil nilai kappa statistic 0,78 yang menghasilkan nilai sedang sehingga dapat digunakan sebagai alternatif pendekatan terhadap complexity [17].

F. Halstead Volume

Halstead metrics merupakan pengukuran complexity metrics berdasarkan jumlah dari operator dan operand [18]. Halstead volume menghitung functional point atau pengukuran dari program size [19]. Halstead volume (HV) merupakan salah satu bagian dari halstead metrics. Berikut [15] persamaan dari HV dapat dilihat pada persamaan 2.

$$HV = N \times \log_2 n \quad (2)$$

Keterangan :

HV = *Halstead volume*

N = Nilai kalkulasi *length of program*

n = Nilai kalkulasi *vocabulary of the program*

Length of program mengkalkulasikan total operator dan operand pada program [16]. Persamaan [15] dari *length of program* dapat dilihat pada persamaan 3.

$$N = N1 + N2 \quad (3)$$

Keterangan :

N = *Length of program*

N1 = Total semua operator yang muncul

N2 = Total semua operand yang muncul

Vocabulary of the program adalah kalkulasi dari jumlah operator dan operand unik yang muncul dalam program [16]. Berikut [15] persamaan dari *vocabulary of the program* dapat dilihat pada persamaan 4.

$$n = n1 + n2 \quad (4)$$

Keterangan :

n = *Vocabulary of program*

n1 = Jumlah operator unik

n2 = Jumlah operand unik

G. Maintainability Index

MI digunakan untuk mengukur tingkat kesulitan dari perawatan atau perubahan sebuah perangkat lunak di masa mendatang dengan mengkalkulasi berdasarkan LoC (*Line of Code*), CC, dan HV [20]. Persamaan [20] dari MI dapat dilihat pada persamaan 5.

$$MI = 171 - 5.2 \times \ln(HV) - 0.23 \times CC - 16.2 \times \ln(Loc) + (50 \times \sin(\sqrt{2.46 \times perCM})) \quad (5)$$

Keterangan :

HV = *Halstead Volume*

CC = *Cyclomatic Complexity*

LoC = *Lines of Code*

perCM = *Percent line of comment*

Hasil dari persamaan tersebut kemudian menghasilkan nilai yang diklasifikasikan. Berikut pengklasifikasiannya dapat dilihat pada tabel 3.

TABEL 3. Klasifikasi MI [20]

Nilai Maintainability Index	Klasifikasi
MI > 85	<i>Highly maintainability</i>
65 < MI ≤ 85	<i>Moderately maintainability</i>
MI ≤ 65	<i>Difficult to maintain</i>

Metode pengukuran MI tersebut telah diujikan dengan mengambil 50 *method* pada aplikasi AsciiDocFX dan mendapatkan akurasi 100% dari 50 *method* tersebut [20].Code Readability Prediction

CRP ini dihasilkan setelah korelasi antara struktur kode dan *code readability* diperoleh [21]. Berikut [21] persamaan dari CRP dapat dilihat pada persamaan 6.

$$CRP = 4.317 + LoC \times (-0.037) + BL \times (0.015) + NOP \times (-0.005) + IL \times (-0.093) \quad (6)$$

Keterangan :

CRP = *Code readability prediction*

LoC = *Line of code*

BL = *Number of blank lines*

IL = *Length of identifier*

Sedangkan untuk persamaan dari *length of identifier* yang merupakan rata-rata panjang identifier diperoleh melalui persamaan 7 [21].

$$IL = \frac{LOI}{TOI} \quad (7)$$

Keterangan :

IL = *Length of identifier*

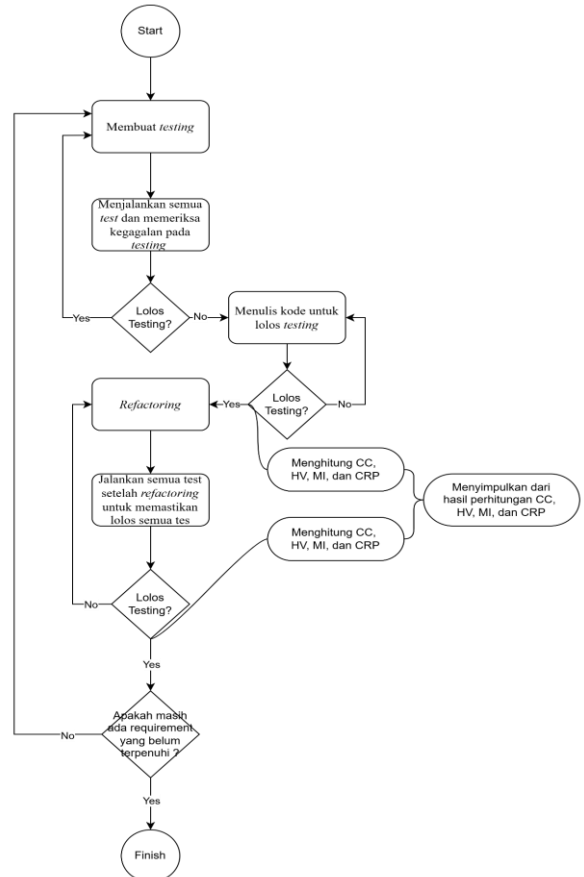
LOI = *Total length of identifier*

TOI = *Total identifier*

Persamaan 7 tersebut diujikan pada 100 kode dan menghasilkan akurasi dari prediksi sebesar 87.02% dan *average deviation* sebesar 0.414 [21].

III. METODE

Metode penelitian dilakukan sesuai dengan diagram alir pada Gambar X. Diagram Alir Metodologi Penelitian.



GAMBAR 2. Diagram alir metodologi penelitian

Diagram tersebut digunakan sebagai dasar pengembangan website pada penelitian ini. Membuat testing

Testing tersebut dibuat berdasarkan requirement yang diberikan oleh Kepala Program Studi S1 PJJ Informatika dapat dilihat pada lampiran 1 sebagai pengembangan website penilaian e-learning readiness model Hung. Testing tersebut ditulis pada laravel dusk.

A. Menjalankan semua test dan memeriksa kegagalan pada testing

Testing tersebut dijalankan untuk menguji kode. Apabila terjadi kegagalan maka diperiksa penyebab kegagalan tersebut. Kegagalan tersebut diperbaiki pada tahap penulisan kode.

B. Penulisan kode

Penulisan kode ini bertujuan agar lolos testing. Apabila masih terjadi kegagalan pada testing maka dilakukan penulisan hingga lolos *testing*. Setelah lolos testing maka kode tersebut dihitung CC, HV, MI, dan CRP untuk selanjutnya dibandingkan hasilnya dengan perhitungan setelah refactoring menggunakan XM. Kode tersebut ditulis menggunakan bahasa php dengan framework laravel. Beberapa sistem juga dibangun sesuai dengan requirement dari Kepala Program Studi S1 PJJ Informatika, Berikut sistem yang dibangun pada tabel 4

TABEL 4.
Sistem yang dibangun

No	Sistem	Keterangan	User
1	Register	User dapat mendaftar pada Pengujian ini	Mahasiswa
2	Login	User dapat login agar bisa mengakses website	Admin, mahasiswa
3	Forget Password	Fitur ini memungkinkan user untuk mengganti password, link akan dikirim pada email	Admin, mahasiswa
4	Input survey e-learning readiness	User dapat memasukkan survey sesuai dengan pilihan mereka. User dapat mengisi kuesioner lebih dari sekali.	Mahasiswa
5	Quiz Computer Self-efficacy	User dapat memasukkan quiz sesuai dengan pilihan mereka. User dapat mengisi quiz ini lebih dari sekali.	Mahasiswa
6	Quiz Computer Learner Control	User dapat memasukkan quiz sesuai dengan pilihan mereka. User dapat mengisi quiz ini lebih dari sekali.	Mahasiswa
7	Quiz Computer Motivation for Learning	User dapat memasukkan quiz sesuai dengan pilihan mereka. User dapat mengisi quiz ini lebih dari sekali.	Mahasiswa
8	Quiz Online Communication Self Efficacy	User dapat memasukkan quiz sesuai dengan pilihan mereka. User dapat mengisi quiz ini lebih dari sekali.	Mahasiswa
9	Quiz Self directed learning	User dapat memasukkan quiz sesuai dengan pilihan mereka. User dapat mengisi quiz ini lebih dari sekali.	Mahasiswa
10	Guidance Page	Guidance page memiliki tampilan berdasarkan hasil survey. Jika hasil survey di	Mahasiswa

		bawah 4.2 maka akan menampilkan tips untuk meningkatkan matriks tersebut. Sedangkan, jika di atas 4.2 akan halaman tersebut akan menunjukkan pengetahuan umum seputar matriks tersebut.	
11	Admin show survey result	Admin dapat melihat hasil survei dari mahasiswa. Admin dapat melakukan searching berdasarkan nama dan nim. Selain itu, admin juga bisa melakukan filter berdasarkan kelas dan waktu pengisian.	Admin
12	Admin show quiz result	Admin dapat melihat hasil quiz dari mahasiswa. Admin dapat melakukan searching berdasarkan nama dan nim. Selain itu, admin juga bisa melakukan filter berdasarkan kelas dan waktu pengisian.	Admin

C. Refactoring

Refactoring pada TDD bertujuan untuk mengurangi kompleksitas, meningkatkan maintainability, dan readability pada kode. Refactoring pada penelitian ini menggunakan XM. XM dipilih karena dapat mengurangi duplikasi pada kode dan meningkatkan readability pada kode. XM dilakukan dengan memindahkan kode dari *method* lama ke *method* baru. Selain itu, XM sesuai untuk mengurangi long method pada paradigma pemrograman functional programming. Berikut contoh penerapan pada function `guidanceLearnerControl` :

```
public function
guidanceLearnerControl() {
    $user = Auth::User();
    if ($user->nim == NULL) {
        return to_route('login');
    }
    $lastResults =
    Answer::where('nim', $user->nim)-
    >orderBy('id', 'desc')->first();
    if ($lastResults->avgLC >= 4.2)
    {
        return
        view('passedLearningControl');
    } else {
        return
        view('unpassedLearningControl');
    }
}
```

GAMBAR 3.

Sebelum refactoring XM

```
public function
guidanceLearnerControl() {
    if (checkLogin()) {
        return checkLogin();
    }
    $nim = getNim();
    $lastResults =
    lastExists($nim);
    return
    viewGuidanceLC($lastResults-
    >avgLC);
}
```

GAMBAR 4.

Sebelum refactoring XM

Refactoring tersebut dilakukan secara berulang pada kode yang ditulis. Refactoring ini dilakukan sebanyak 27 iterasi pada 27 function. Iterasi meliputi proses refactoring hingga lolos pengujian. Lebih detail iterasi refactoring tersebut dapat dilihat pada tabel iterasi refactoring di lampiran 3.

D. Testing setelah refactoring

Testing ini digunakan untuk menjamin kode setelah refactoring berjalan sesuai dengan fungsinya. Apabila terjadi kegagalan pada testing maka dilakukan perbaikan pada refactoring ini. Setelah lolos pengujian maka dihitung CC, HV, MI, dan CRP. Hasil perhitungan tersebut selanjutnya dibandingkan dan diperoleh

kesimpulan dari perbandingan CC, HV, MI, dan CRP sebelum dan sesudah refactoring XM. Selain itu, juga dilanjutkan pada siklus yang baru dimulai dari pembuatan testing. Terakhir setelah semua requirement terpenuhi maka website diserahkan kepada pihak S1 PJJ Informatika. Surat keterangan dari tempat studi dapat dilihat pada lampiran 4.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian

Pengujian ini dilakukan dengan menghitung CC, HV, MI, dan CRP pada setiap *function* yang dilakukan *refactoring* sehingga diperoleh hasil penghitungan sebelum dan sesudah *refactoring* terhadap komponen tersebut. Hasil dari penghitungan tersebut dianalisis untuk mendapatkan pengaruh XM terhadap CC, HV, MI, dan CRP pada penelitian ini. Berikut rata-rata penurunan CC, penurunan HV, peningkatan MI, dan peningkatan CRP yang dapat dilihat pada tabel 5.

Tabel 5.
Peningkatan nilai CC, HV, MI, dan CRP

Persentase rata-rata			
Penurunan CC	Penurunan HV	Peningkatan MI	Peningkatan CRP
31%	68%	28%	4%

Tabel di atas menunjukkan implementasi XM untuk studi kasus ini menunjukkan hasil rata-rata yang lebih baik pada CC, HV, MI, dan CRP. Lebih detail untuk hasil pengujian terhadap CC, HV, MI, dan CRP dapat dilihat pada lampiran 5.

B. Analisis Hasil Pengujian

Penerapan XM sesuai terhadap pengembangan TDD terlebih untuk paradigma pemrograman *functional programming* yang memiliki permasalahan *long method*. *Long method* tersebut terjadi karena menggabungkan beberapa logika pada satu skrip. Hal tersebut dapat dilihat melalui hasil pengujian di atas diperoleh hasil rata-rata yang lebih baik pada CC, HV, MI, dan CRP. Namun, terdapat penurunan nilai dari CRP setelah dilakukan *refactoring*. Hal ini disebabkan karena tingginya nilai IL setelah dilakukan *refactoring* XM. Walaupun diperoleh penurunan LOC pada kode namun IL meningkat.

Selain itu, beberapa *function* mengalami *error* pada penghitungan CRP karena memiliki nilai di bawah 1 pada kode sebelum XM. Hal tersebut dikarenakan jumlah LOC yang banyak. *Function* *homeView* pada file *homeController* yang memiliki *line* terbanyak memiliki nilai CRP terendah yaitu -0,97. Hal tersebut juga didukung dengan seluruh kode yang memiliki nilai CRP di bawah 1 mempunyai 48 LOC. Selain itu, juga NOP berpengaruh *error* tersebut. Hal ini didukung dengan semua *function* yang mengalami *error* pada perhitungan CRP memiliki nilai NOP di atas 13,51 dan LOC di atas 38.

Selain itu juga, terdapat beberapa *function* yang mengalami penurunan CRP setelah *refactoring* XM

dibandingkan sebelum *refactoring* dikarenakan nilai IL yang meningkat setelah *refactoring*. Hal ini dapat diatasi dengan *refactoring rename parameter* dan *rename method* agar menghasilkan nilai IL yang lebih rendah sehingga menghasilkan CRP yang lebih tinggi. Berikut contoh penerapannya :

```
public function viewQuizComputerSelfEfficacy() {
    if (checkLogin()) {
        return checkLogin();
    }
    return view('quizComputerSelfEfficacy');
}
```

GAMBAR 5.

Sebelum *refactoring rename parameter* dan *rename method*

```
public function viewQuizCSE() {
    if (checkLogin()) {
        return checkLogin();
    }
    return view('quizCSE');
}
```

GAMBAR 6.

Setelah *refactoring rename parameter* dan *rename method*

Pada contoh di atas setelah dilakukan *rename parameter* dan *rename method* menghasilkan nilai CRP sebesar 3,29 dan nilai IL sebesar 8,4. Terdapat peningkatan nilai CRP dibandingkan sebelum *refactoring* XM yang bernilai 3,04 dan setelah *refactoring* XM yang bernilai 2,66. Selain itu, terdapat penurunan nilai IL dibandingkan sebelum *refactoring* XM yang bernilai 10,67 dan setelah *refactoring* XM yang bernilai 15,2. Nilai CRP dan IL saling berbanding terbalik, karena nilai CRP yang tinggi memiliki nilai IL yang rendah.

V. KESIMPULAN

TUNERS telah dibuat berdasarkan requirement dari Kepala Program Studi S1 PJJ Informatika. Website ini akan digunakan oleh program studi S1 PJJ Informatika sebagai penilaian *e-learning readiness model Hung*. Penerapan XM pada TDD terhadap pembuatan TUNERS memberi persentase rata-rata penurunan CC dan HV yang positif, serta persentase rata-rata peningkatan MI dan CRP yang positif. Oleh karena itu, tujuan *refactoring* pada TDD tercapai dengan penerapan XM. Namun, terdapat error pada nilai CRP karena LOC dan IL yang tinggi. Selain itu, terdapat penurunan nilai CRP setelah *refactoring* XM jika dibandingkan sebelum *refactoring* XM. Hal ini dapat diatasi dengan melakukan *refactoring rename parameter* dan *rename method*. Oleh karena itu, pada penelitian selanjutnya peneliti menyarankan untuk dilakukan *refactoring rename parameter* dan *rename method* untuk meningkatkan CRP pada pengembangan menggunakan TDD

REFERENSI

[1] Mosa, A. A., Mohd. Naz'ri bin Mahrin, & Ibrarrahim, R. 2016. Technological Aspects of E-Learning

Readiness in Higher Education: A Review of the Literature. *Comput. Inf. Sci.*, 9(1), 113–127.

[2] Rohayani, A. H. 2015. A literature review: Readiness factors to measuring e-learning readiness in higher education. *Procedia Computer Science*, 59, 230–234. Elsevier

[3] Gunawan, A. C., Herdiani, A., and G. A. A. Wisudiawan. 2021. Perancangan Sistem dan Analisis e-Learning Readiness Mahasiswa Studi Kasus: Mahasiswa S1 PJJ Informatika Universitas Telkom. Bandung. Universitas Telkom

[4] Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. 2020. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 14(11).

[5] Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. 2017. Agile software development models tdd, fdd, dsdm, and crystal methods: A survey. *International Journal of Multidisciplinary Sciences and Engineering*, 8(2), 1–10.

[6] Mariani, T., & Vergilio, S. R. 2017. A systematic review on search-based refactoring. *Information and Software Technology*, 83, 14–34. Elsevier

[7] Agnihotri, M., & Chug, A. 2020. A systematic literature survey of software metrics, code smells and refactoring techniques. *Journal of Information Processing Systems*, 16(4), 915–934. Korea Information Processing Society

[8] Hermans, F., & Aivaloglou, E. 2016. Do code smells hamper novice programming? A controlled experiment on Scratch programs. *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, 1–10. IEEE

[9] Compeau, D. R., & Higgins, C. A. 1995. Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly*, 189–211. JSTOR

[10] Loyens, S. M., Magda, J., & Rikers, R. M. 2008. Self-directed learning in problem-based learning and its relationships with self-regulated learning. *Educational Psychology Review*, 20, 411–427. Springer

[11] Shyu, H.-Y., & Brown, S. W. 1992. Learner control versus program control in interactive videodisc instruction: What are the effects in procedural learning. *International Journal of Instructional Media*, 19(2), 85–95.

[12] Chung, E., Noor, N. M., & Mathew, V. N. 2020. Are you ready? An assessment of online learning readiness among university students. *International Journal of Academic Research in Progressive Education and Development*, 9(1), 301–317.

[13] Pervez, M. U., Eman, L., & Abbas, B. D. 2022. *Test Driven Development: A Review*. Nov.

[14] Bissi, W., Neto, A. G. S. S., & Emer, M. C. F. P. 2016. The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology*, 74, 45–54. Elsevier

[15] Debbarma, M. K., Debbarma, S., Debbarma, N., Chakma, K., & Jamatia, A. 2013. A review and analysis of software complexity metrics in structural testing.

International Journal of Computer and Communication Engineering, 2(2), 129–133. IACSIT Press

[16] Abd Jader, M. N., & Mahmood, R. Z. 2018. Calculating McCabe's Cyclomatic Complexity Metric and Its Effect on the Quality Aspects of Software. *IJRCT*

[17] Subandri, M. A., & Sarno, R. 2017. Cyclomatic complexity for determining product complexity level in COCOMO II. *Procedia Computer Science*, 124, 478–486. Elsevier

[18] Laird, L. M., & Brennan, M. C. 2006. *Software measurement and estimation: A practical approach*. John Wiley & Sons. John Wiley & Sons

[19] Thirumalai, C., Shridharshan, R. R., & Reynold, L. R. 2017. An assessment of halstead and COCOMO model for effort estimation. *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 1–4. IEEE

[20] Atmaja, R. G., Priyambadha, B., & Pradana, F. 2019. Pembangunan Kakas Bantu Untuk Mengukur Maintainability Index Pada Perangkat Lunak Berdasarkan Nilai Halstead Metrics dan McCabe's Cyclomatic Complexity: English. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 3(3), 2167–2172. Universitas Brawijaya

[21] Batool, A., Bashir, M. B., Babar, M., Sohail, A., & Ejaz, N. 2021. Effect or Program Constructs on Code Readability and Predicting Code Readability Using Statistical Modeling. *Foundations of Computing and Decision Sciences*, 46(2), 127–145.