LOGIC: Jurnal Penelitian Informatika
Vol. 3, Issue 1, pp. 20–26 (2025)
doi: http://doi.org/10.25124/logic.v3i1.6467

RESEARCH ARTICLE

Pengaruh *Refactoring Extract Method* terhadap Pengembangan Aplikasi menggunakan *Test Driven Development*

Fauzi Hazim Wibowo, Dawam Dwi Jatmiko Suwawi* and Anisa Herdiani

Fakultas Informatika, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: dawamdjs@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*.

Key words: *test driven development, extract method, cyclomatic complexity, halstead volume, maintainability index, code readability prediction*.

Pendahuluan

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 x, dan *refactoring codie* [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 *readabilitas* 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 *readabilitas* 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.

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 EReadiness 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.

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 mengidentifikasikan pengaruh *extract method* terhadap *cyclomatic complexity*, *halstead volume*, *maintainability index*, dan *code readability prediction* pada pengembangan perangkat lunak dengan *test driven development*.

Tinjauan Pustaka

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.

Table 1. Tabel model Hung [3]

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

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

Extract Method

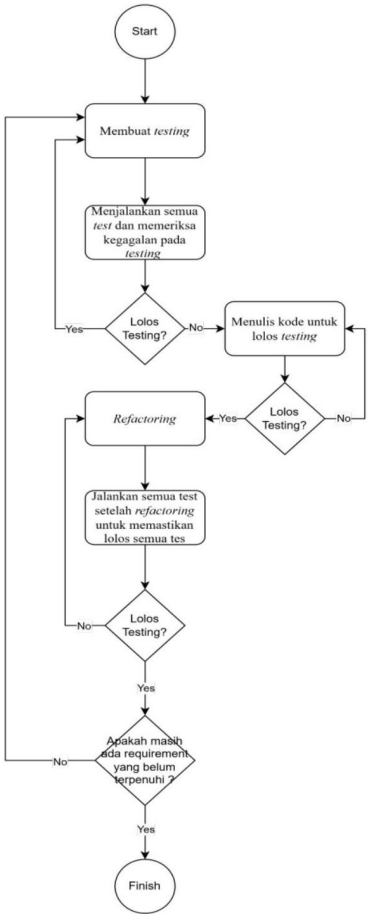
Extract Method (XM) memiliki keunggulan dapat meningkatkan *readability* dan mengurangi duplikasi pada kode [7]. XM mengubah bagian kode menjadi *method* [6]. Hal tersebut dilakukan dengan memindahkan kode dari *method* lama ke *method* baru [7]. Metode ini dapat digunakan ketika program dengan paradigma pemrograman *functional programming* mengalami masalah *long method*. *Long method* terjadi karena menggabungkan banyak logika pada satu skrip [8]. XM membagi kode pada *method* baru.

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].

Cyclomatic Complexity

Cyclomatic Complexity (CC) merupakan sebuah matrik untuk mengukur kompleksitas dari sebuah program [17]. CC mengukur kompleksitas



Gambar 1. Proses TDD [14]

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.

$$CC = E - n + 2 \tag{1}$$

Keterangan :

$\mathrm{CC} =$ *Cyclomatic complexity*

$\mathrm{E} =$ *Edge*

$\mathrm{n} =$ *Node*

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

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.

$$\mathrm{HV} = \mathrm{N} \times \log_2 \mathrm{n} \tag{2}$$

Keterangan :
HV = *Halstead volume*
$\mathrm{N} =$ Nilai kalkulasi length of program
$\mathrm{n} =$ 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.

$$N = N1 + N2 \tag{3}$$

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.

$$n = n1 + n2 \tag{4}$$

Keterangan :
$\mathrm{n} =$ *Vocabulary of program*
n1 = Jumlah operator unik
n2 = Jumlah *operand* unik

Table 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

Maintainability Index

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

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

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. Metode pengukuran MI tersebut telah diujikan dengan mengambil 50 *method* pada aplikasi AsciidocFX dan mendapatkan akurasi 100% dari

Table 3. Klasifikasi MI [20]

Maintainability Index	Klasifikasi
MI > 85	*Highly maintainability*
$65 <$ MI ≤ 85	*Moderately maintainability*
MI ≤ 65	*Difficult to maintain*

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.

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

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].

Metodologi Penelitian

Metode penelitian dilakukan sesuai dengan diagram alir pada Gambar X. 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.

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.

Penulisan kode

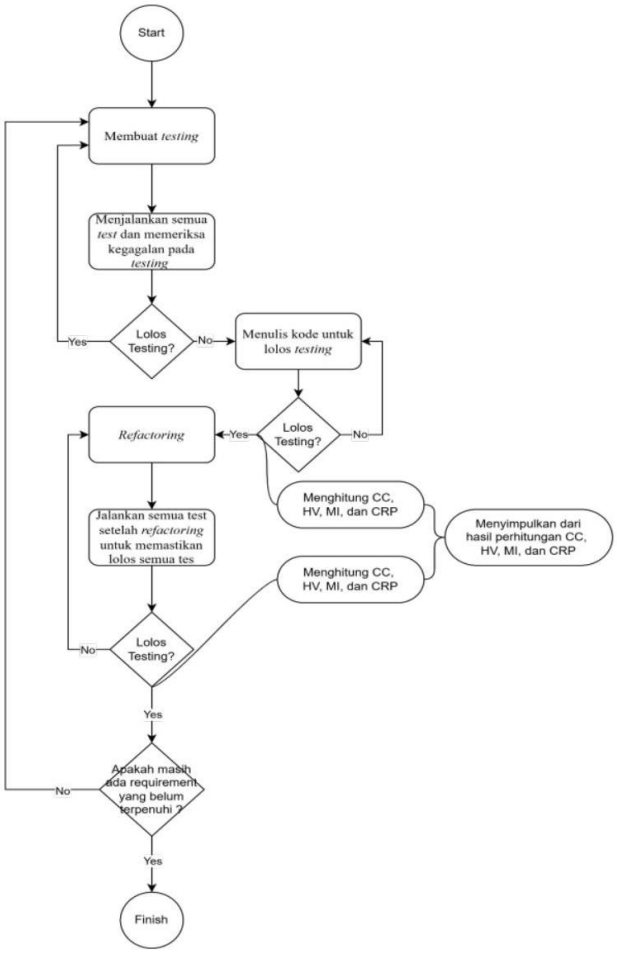
Penulisan kode ini bertujuan agar lolos testing. Apabila masih terjadi kegagalan pada testing maka dilakukan penulisan hingga lolos testing. Setelah lolos testing maka kode tersebut dihitung CC, HV, MI, dan CRP untuk selanjutnya dibandingkan hasilnya dengan perhitungan setelah *refactoring* menggunakan 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.

Refactoring

Refactoring pada TDD bertujuan untuk mengurangi kompleksitas, meningkatkan *maintainability*, dan *readability* pada kode. *Refactoring* pada penelitian ini menggunakan XM. XM dipilih karena dapat

Table 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 elearning readiness	User dapat memasukkan survey sesuai dengan pilihan mereka. User dapat mengisi kuesioner lebih dari sekali.	Mahasiswa
5	Quiz textitComputer Selfefficacy	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 Commun ication 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 dibawah 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.	Mahasiswa
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



Gambar 2. Diagram alir metodologi penelitian

```
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

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* 3 4: *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.

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.

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

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

Hasil dan Pembahasan

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 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.

```
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*

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 5: Setelah *refactoring rename parameter* dan *rename method* Pada contoh di atas 6 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.

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.

Daftar Pustaka

1. Mosa AA, Mahrin MNb, Ibrrahim R. Technological Aspects of E-Learning Readiness in Higher Education: A Review of the Literature. Computer and Information Science. 2016;9(1):113-27.
2. Rohayani AH. A Literature Review: Readiness Factors to Measuring E-Learning Readiness in Higher Education. Procedia Computer Science. 2015;59:230-4.
3. Gunawan AC, Herdiani A, Wisudiawan GAA. Perancangan Sistem dan Analisis eLearning Readiness Mahasiswa Studi Kasus: Mahasiswa S1 PJJ Informatika Universitas Telkom. Bandung; 2021. Bachelor's thesis, Universitas Telkom.
4. Al-Saqqa S, Sawalha S, AbdelNabi H. Agile Software Development: Methodologies and Trends. International Journal of Interactive Mobile Technologies. 2020;14(11).
5. Anwer F, Aftab S, Waheed U, Muhammad SS. Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey. International Journal of Multidisciplinary Sciences and Engineering. 2017;8(2):1-10.
6. Mariani T, Vergilio SR. A Systematic Review on Search-Based Refactoring. Information and Software Technology. 2017;83:14-34.
7. Agnihotri M, Chug A. A Systematic Literature Survey of Software Metrics, Code Smells and Refactoring Techniques. Journal of Information Processing Systems. 2020;16(4):915-34.
8. Hermans F, Aivaloglou E. Do Code Smells Hamper Novice Programming? A Controlled Experiment on Scratch Programs. In: 2016 IEEE 24th International Conference on Program Comprehension (ICPC). IEEE; 2016. p. 1-10.
9. Compeau DR, Higgins CA. Computer Self-Efficacy: Development of a Measure and Initial Test. MIS Quarterly. 1995:189-211.
10. Loyens SM, Magda J, Rikers RM. Self-Directed Learning in Problem-Based Learning and Its Relationships with Self-Regulated Learning. Educational Psychology Review. 2008;20:411-27.
11. Shyu HY, Brown SW. Learner Control versus Program Control in Interactive Videodisc Instruction: What are the Effects in Procedural Learning. International Journal of Instructional Media. 1992;19(2):85-95.
12. Chung E, Noor NM, Mathew VN. Are You Ready? An Assessment of Online Learning Readiness Among University Students. International Journal of Academic Research in Progressive Education and Development. 2020;9(1):301-17.
13. Pervez MU, Eman L, Abbas BD. Test Driven Development: A Review; 2022. No journal info provided.
14. Bissi W, Neto AGSS, Emer MCFP. The Effects of Test Driven Development on Internal Quality, External Quality and Productivity: A Systematic Review. Information and Software Technology. 2016;74:45-54.
15. Debbarma MK, Debbarma S, Debbarma N, Chakma K, Jamatia A. A Review and Analysis of Software Complexity Metrics in Structural Testing. International Journal of Computer and Communication Engineering. 2013;2(2):129-33.
16. Abd Jader MN, Mahmood RZ. Calculating McCabe's Cyclomatic Complexity Metric and Its Effect on the Quality Aspects of Software. International Journal of Innovative Research in Computer Technology (IJIRCT). 2018. No volume or page numbers provided.

17. Subandri MA, Sarno R. Cyclomatic Complexity for Determining Product Complexity Level in COCOMO II. Procedia Computer Science. 2017;124:478-86.
18. Laird LM, Brennan MC. Software Measurement and Estimation: A Practical Approach. John Wiley & Sons; 2006.
19. Thirumalai C, Shridharshan RR, Reynold LR. An Assessment of Halstead and COCOMO Model for Effort Estimation. In: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT). IEEE; 2017. p. 1-4.
20. Atmaja RG, Priyambadha B, Pradana F. 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. 2019;3(3):2167-72.
21. Batool A, Bashir MB, Babar M, Sohail A, Ejaz N. Effect of Program Constructs on Code Readability and Predicting Code Readability Using Statistical Modeling. Foundations of Computing and Decision Sciences. 2021;46(2):127-45.