Tugas I Makassar, 22 September 2025

**MAKALAH**

**Test – Driven Development (TDD)**



Nama : Rofifah Gina Tamala (13020230091)

Nur azizah (13020230088)

Aqilah Zahra (13020230012)

Sitti Hadijah (13020230014)

Aiska Suci Rahmadani S. (13020230092)

Kelas : B1

Mata Kuliah : Rekayasa Perangkat Lunak Objek

Dosen : Mardiyyah Hasnawi, S. Kom., M.T.

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MUSLIM INDONESIA**

**MAKASSAR**

**2025**

**Daftar Isi**

**BAB I PENDAHULUAN ............................................................................1**

**1.1 Latar Belakang ..................................................................................1**

**1.2 Rumusan Masalah .............................................................................2**

**1.3 Tujuan Penulisan ...............................................................................3**

**1.4 Manfaat Penulisan .............................................................................4**

**BAB II KAJIAN TEORITIS ......................................................................5**

**2.1 Definisi dan Sejarah Test Driven Development (TDD) .................5**

**2.2 Penerapan Model (Aktivitas/Fase TDD) .........................................6**

**2.3 Karakteristik Model TDD ................................................................7**

**2.4 Kelebihan dan Kelemahan TDD ......................................................8**

**2.5 Perbandingan dengan Model atau Metodologi Lain .....................9**

**2.6 Alat Bantu Model atau Metode TDD .............................................10**

**BAB III PENUTUP ....................................................................................11**

**3.1 Kesimpulan .......................................................................................11**

**DAFTAR PUSTAKA .................................................................................17**

# BAB I

# PENDAHULUAN

## **1.1 Latar Belakang**

Perangkat lunak kini menjadi kebutuhan utama di berbagai bidang sehingga menuntut kualitas tinggi, cepat, aman, dan efisien. Metode tradisional seperti Waterfall dianggap kaku karena pengujian baru dilakukan di akhir, yang berakibat pada meningkatnya bug, biaya perbaikan, dan keterlambatan rilis. Test Driven Development (TDD) hadir sebagai solusi dengan prinsip test-first approach, yaitu menulis tes sebelum kode dibuat. [1]

Sejumlah penelitian membuktikan efektivitas TDD. Rafiadly dkk. (2023) menunjukkan aplikasi Naviku berhasil diuji 100% menggunakan TDD. Siraj & Setiani (2025) menemukan TDD menurunkan kompleksitas kode hingga lebih dari 80% pada aplikasi Sedonor. Moe (2019) menegaskan keunggulan TDD dibanding metode lain seperti BDD dan ATDD, meski masing-masing punya kekuatan tersendiri. [2]

Dengan demikian, TDD dipandang bukan hanya metode teknis, tetapi paradigma baru dalam pengembangan perangkat lunak yang menekankan kualitas, modularitas, dan keberlanjutan.

## **1.2 Rumusan Masalah**

Berdasarkan uraian latar belakang, maka rumusan masalah dalam makalah ini adalah:

1. Apa definisi dan sejarah dari Test Driven Development (TDD)?
2. Bagaimana penerapan fase-fase TDD dalam pengembangan perangkat lunak?
3. Apa karakteristik utama yang membedakan TDD dari metodologi pengembangan perangkat lunak lainnya?
4. Apa kelebihan dan kelemahan TDD dalam praktik nyata?
5. Bagaimana perbandingan TDD dengan metode lain seperti Waterfall, BDD, dan ATDD? [1]
6. Apa saja alat bantu yang mendukung penerapan TDD dalam pengembangan perangkat lunak modern?
7. Bagaimana hasil penerapan TDD berdasarkan penelitian empiris yang telah dilakukan?

## **1.3 Tujuan Penulisan**

Tujuan dari penulisan makalah ini adalah:

1. Menjelaskan definisi, sejarah, serta konsep dasar Test Driven Development (TDD).
2. Mendeskripsikan aktivitas dan fase utama dalam siklus TDD.
3. Mengidentifikasi karakteristik, kelebihan, serta kelemahan TDD.
4. Membandingkan TDD dengan metodologi lain, khususnya Waterfall, BDD, dan ATDD. [1]
5. Menjelaskan alat bantu yang dapat digunakan dalam penerapan TDD.
6. Menganalisis hasil penelitian empiris yang menggunakan TDD dalam pengembangan perangkat lunak.
7. Memberikan pemahaman menyeluruh tentang peran TDD dalam meningkatkan kualitas perangkat lunak.

## **1.4 Manfaat Penulisan**

Manfaat dari penulisan makalah ini dapat dilihat dari dua aspek, yaitu:

1. Manfaat Teoretis:
   * Memberikan kontribusi terhadap pemahaman mengenai TDD sebagai salah satu metodologi pengembangan perangkat lunak modern.
   * Menjadi referensi bagi mahasiswa dan peneliti dalam bidang rekayasa perangkat lunak mengenai efektivitas TDD.
2. Manfaat Praktis:
   * Memberikan gambaran nyata tentang penerapan TDD dalam pengembangan aplikasi berbasis Android.
   * Menjadi bahan pertimbangan bagi pengembang perangkat lunak dalam memilih metodologi yang sesuai dengan kebutuhan proyek.
   * Membantu tim pengembang memahami kelebihan dan keterbatasan TDD, sehingga dapat menggunakannya secara tepat.

# BAB II

# KAJIAN TEORITIS

## **2.1 Definisi dan Sejarah Test Driven Development (TDD)**

Test Driven Development (TDD) adalah salah satu pendekatan dalam pengembangan perangkat lunak yang menekankan pembuatan *unit test* terlebih dahulu sebelum menulis kode produksi. Prinsip utama TDD dikenal dengan istilah *test-first approach*, artinya pengujian menjadi dasar untuk membangun implementasi kode. Dengan cara ini, perangkat lunak dikembangkan melalui siklus pendek yang berulang-ulang sehingga kualitas kode dapat terjaga sejak awal.

TDD pertama kali diperkenalkan oleh Kent Beck (2002) dalam bukunya *Test Driven Development: By Example*. Konsep ini kemudian menjadi bagian penting dari Extreme Programming (XP) yang termasuk ke dalam kerangka metodologi Agile. Agile menekankan kolaborasi, fleksibilitas, dan iterasi cepat, sementara TDD hadir sebagai praktik teknis untuk memastikan kualitas produk dalam kerangka tersebut.

Penelitian terkini semakin memperkuat relevansi TDD. Rafiadly dkk. (2023) membuktikan efektivitas TDD dalam aplikasi Naviku, dengan hasil seluruh fitur berhasil diuji 100%. Hal ini menegaskan bahwa TDD bukan hanya teori, tetapi telah diaplikasikan secara nyata dalam pengembangan aplikasi modern [2]

## **2.2 Penerapan Model (Aktivitas/Fase TDD)**

Penerapan TDD biasanya mengikuti siklus Red – Green – Refactor:

1. Red (Test Fails): Pengembang menulis tes terlebih dahulu, hasil awal pasti gagal karena kode belum dibuat.
2. Green (Test Passes): Menulis kode minimum agar tes lulus. Tujuannya hanya membuat tes berjalan, tanpa memperhatikan efisiensi.
3. Refactor: Setelah tes berhasil, kode disempurnakan agar lebih rapi, efisien, dan mudah dipelihara.

Siklus ini berulang secara iteratif untuk setiap fitur kecil hingga membentuk sistem lengkap.

Pada penelitian Siraj & Setiani (2025), TDD diterapkan pada aplikasi Sedonor untuk meningkatkan kualitas kode. Hasilnya menunjukkan penurunan signifikan pada metrik kompleksitas seperti Lines of Code (LOC) dan Response for Class (RFC), membuktikan bahwa siklus TDD mampu menghasilkan kode yang modular dan mudah diuji [3]

## **2.3 Karakteristik Model TDD**

Beberapa karakteristik utama TDD antara lain:

* Iteratif dan Incremental: pengembangan dilakukan dalam potongan kecil yang diuji secara bertahap.
* Berorientasi pada Unit Test: setiap fungsi diuji secara individual.
* Refactoring Berkelanjutan: kode terus diperbaiki agar tetap bersih tanpa mengubah fungsionalitas.
* Dokumentasi Otomatis: unit test berfungsi sebagai dokumentasi perilaku sistem.
* Meningkatkan Modularitas: kode lebih mudah dipelihara dan diuji kembali ketika ada perubahan.

Karakteristik ini terbukti dalam penelitian Sedonor, di mana penerapan TDD membuat kode lebih modular dan efisien sehingga lebih mudah dikembangkan ke tahap selanjutnya [3]

## **2.4 Kelebihan dan Kelemahan TDD**

Kelebihan TDD:

1. Mengurangi Bug Sejak Awal: Karena pengujian ditulis lebih dulu, kesalahan dapat terdeteksi sebelum kode berkembang lebih jauh
2. Kode Lebih Modular: Penelitian Sedonor menunjukkan modularitas meningkat dengan turunnya metrik LOC hingga 91,9% [3]
3. Mendukung Refactoring Aman: Perubahan kode dapat dilakukan tanpa khawatir merusak fungsionalitas.
4. Dokumentasi Otomatis: Unit test menjadi dokumentasi yang jelas tentang cara kerja kode.
5. Meningkatkan Kepercayaan Diri Programmer: Setiap perubahan langsung bisa diuji dengan cepat.

Kelemahan TDD:

1. Membutuhkan Waktu di Awal: Penulisan tes sebelum kode menambah waktu pengerjaan awal proyek
2. Tidak Mudah pada Legacy Code: Sulit diterapkan pada kode lama yang tidak modular.
3. Butuh Keahlian Tinggi: Pengembang harus paham prinsip pengujian dengan baik.
4. Tidak Selalu Cocok untuk Deadline Ketat: Karena waktu awal pengerjaan bisa lebih lama.

## **2.5 Perbandingan dengan Model atau Metodologi Lain**

1. TDD vs Waterfall
   * Waterfall bersifat linear dan kaku, pengujian dilakukan di akhir.
   * TDD bersifat iteratif, pengujian dilakukan sejak awal.
   * TDD lebih fleksibel terhadap perubahan kebutuhan dibanding Waterfall.
2. TDD vs BDD (Behavior Driven Development)
   * TDD berfokus pada implementasi kode dan unit test yang ditulis oleh developer.
   * BDD menekankan perilaku sistem dalam bahasa alami, mudah dipahami oleh stakeholder.
   * BDD meningkatkan komunikasi tim, sementara TDD lebih teknis
3. TDD vs ATDD (Acceptance Test Driven Development)
   * TDD fokus pada level unit test.
   * ATDD menekankan kolaborasi tim (developer, tester, user) untuk mendefinisikan tes penerimaan.
   * ATDD lebih berorientasi pada kebutuhan pengguna

Dari perbandingan ini, jelas bahwa TDD unggul dalam deteksi bug sejak awal, tetapi BDD dan ATDD unggul dalam aspek kolaborasi dan komunikasi. [1]

## **2.6 Alat Bantu Model atau Metode TDD**

Dalam praktiknya, TDD memerlukan dukungan alat bantu untuk menulis dan menjalankan tes otomatis. Beberapa *framework* populer antara lain:

* JUnit (Java) – digunakan luas pada pengembangan berbasis Java dan Android
* Mockito & MockK – untuk membuat objek tiruan dalam pengujian.
* PyTest, Unittest – untuk Python.
* Jest, Mocha – untuk JavaScript/Node.js.
* NUnit – untuk C#.

Selain itu, integrasi dengan Continuous Integration (CI) tools seperti Jenkins, GitHub Actions, atau GitLab CI semakin memperkuat praktik TDD karena setiap perubahan kode langsung diuji secara otomatis.

# ****BAB III****

# ****PENUTUP****

**3.1 Kesimpulan**  
Berdasarkan pembahasan mengenai Test Driven Development (TDD), dapat disimpulkan bahwa TDD merupakan pendekatan pengembangan perangkat lunak dengan prinsip test-first approach, yaitu menulis tes sebelum kode dibuat. TDD berjalan dalam siklus Red – Green – Refactor yang memungkinkan pengembang mendeteksi bug lebih awal, menjaga modularitas, dan meningkatkan kualitas perangkat lunak.

TDD memiliki sejumlah keunggulan, seperti kode lebih bersih, aman untuk dilakukan refactoring, serta unit test yang sekaligus berfungsi sebagai dokumentasi. Namun, TDD juga memiliki kelemahan, terutama dari sisi kebutuhan waktu dan keterampilan tinggi pada tahap awal pengembangan.

Hasil kajian empiris menunjukkan bahwa TDD dapat meningkatkan kualitas perangkat lunak secara signifikan, misalnya dengan menurunkan kompleksitas kode dan meningkatkan keandalan aplikasi. Dibandingkan metode lain seperti Waterfall, BDD, dan ATDD, TDD unggul dalam aspek teknis, meskipun BDD dan ATDD lebih menonjol dalam kolaborasi antar-stakeholder.

Dengan demikian, TDD bukan hanya sekadar teknik, melainkan paradigma baru dalam rekayasa perangkat lunak yang menekankan kualitas, efisiensi, dan keberlanjutan kode.

**DAFTAR PUSTAKA**

[1] J. Internasional, P. Ilmiah, M. M. Moe, U. S. Komputer, N. B. Kayin, and M. Myint, “Studi Perbandingan antara Pengembangan Berbasis Uji Coba (TDD), Pengembangan Berbasis Perilaku (BDD), dan Pengembangan Berbasis Uji Penerimaan (ATDD),” pp. 231–234, 2019.

[2] M. Rafiadly, R. Fauzi, and A. Musnansyah, “Perancangan Aplikasi Naviku untuk Memberikan Informasi Navigasi Kepada Tunanetra Menggunakan Metode Test Driven Development,” *J. Inf. Syst. Res.*, vol. 4, no. 4, pp. 1455–1463, 2023, doi: 10.47065/josh.v4i4.3948.

[3] M. R. Siraj and N. Setiani, “Pengaruh Test-Driven Development terhadap Metrik Testabilitas Kode dalam Pengembangan Aplikasi Android,” *J. Pendidik. dan Teknol. Indones.*, vol. 5, no. 2, pp. 277–287, 2025, doi: 10.52436/1.jpti.648.