BAB II

Pembahasan

1. **Pengertian test driven development**

Test Driven Development (TDD) adalah skema pengembangan perangkat lunak / aplikasi dengan melakukan unit test terlebih dahulu sebelum melakukan pemrograman lebih lanjut atau production. Metode pengembangan yang “dilandasi” proses pengujian.

1. **Tujuan test driven development**

Tujuan test driven development( TDD )adalah untuk membuat kode lebih jelas, sederhana dan bebas bug.

Pengembangan Test-Driven dimulai dengan merancang dan mengembangkan tes untuk setiap fungsionalitas kecil aplikasi.

Test Driven Development menginstruksikan pengembang untuk menulis kode baru hanya jika tes otomatis gagal. Ini menghindari duplikasi kode. Bentuk penuh TDD adalah pengembangan yang digerakkan oleh tes.

1. **Jenis- jenis test driven development**

* Buat Kode Pengujian (***Add Test***).
* Jalankan test, hasilnya pasti *fail* (***Watch Test Fail***).
* Tulis Kode Fitur Sistem (***Write Code***).
* Jalankan test (***Run Tests***) hingga testing *passed*.
* *Refactor* Kode Fitur Sistem (***Refactor***).

1. **Manfaat penggunaan test driven development**

Manfaat memakai test driven development(**TDD**) Membuat kode yang mudah untuk di-maintain Membuat kode yang modular karena **tes** biasanya cenderung kecil. Dalam membuat **tes** tidak perlu memikirkan arsitektur, jadi pengembangannya bisa sangat cepat

1. **Kelebihan dan kekuarang test driven development**
2. Kelebihan TDD

* Simplicity artinya bahwa kode program yang dibuat harus sederhana, tidak perlu membuat kode panjang jika masih bisa ditulis lebih singkat serta tidak terjadi pengulangan kode program yang sama.
* Readabilityartinya bahwa kode program yang dibuat dapat dibaca dan dimengerti orang lain yang membaca kode program tersebut.
* Modularityartinya kode program ditulis dengan membagi-baginya ke dalam bentuk yang lebih kecil sehingga tidak menggabungkan semua kode dalam satu *page*. Kode program dapat dibagi berdasarkan *model, view, controller*.
* Design artinya bahwa tim pembangun harus memikirkan atau membayangkan terlebih dahulu kode program yang akan dibangun sebelum mulai membangunnya. Tim pembangun harus memikirkan spesifikasi fungsi yang akan dibangun. *Test-Driven Development*menuntut tim pembangun memahami setiap fungsi yang akan dibangun karena tim pembangun tidak bisa membuat kode program tanpa mengerti bagaimana hasil yang akan dicapai dan bagaimana membuat *unit test*-nya[8]. *Test-Driven Development* menuntun tim pembangun untuk mendapatkan desain yang lebih baik, karena tim pembangunbebas membuat desain seiring dilakukannya proses *refactoring* karena tidak dituntut untuk melakukan perancangan desain di awal pembangunan aplikasi[8].
* Efficiencyartinya bahwa performansi program harus baik contohnya adalah
* Clarityartinya kode program, desain, dan tujuan kode program tersebut harus jelas dan benar-benar dimengerti apa yang dilakukan pada setiap level.

1. Kekurangan TDD

* Sulit menentukan *unit test* yang benar dalam sebuah studi kasus karena tidak ada ukuran yang menunjukkan kebenaran dari suatu *unit test*yang dibangun. \*soalnya kita kan buat test dulu baru buat code nya
* Tim pembangun sulit membuat abstraksi dari sebuah *interface* karena tidak ada *physical design* yang dilakukan di awal proses pembangunan aplikasi. Tim pembangun hanya membangun aplikasi dengan *design thinking*.
* Dalam pembangunan aplikasi dengan metodologi ini dibutuhkan tim pembangun yang sudah berpengalaman dan memahami bagaimana menulis *test* yang baik dan juga mengerti sedikit tentang arsitektur yang baik sehingga tidak cocok bagi tim pembangun pemula.

1. **Prinsip test driven development**

Prinsip dari test driven development adalah merupakan sebuah proses pada pembuatan perangkat lunak yang sangat bergantung kepada uji coba yang berubah-ubah kasusnya sesuai kebutuhan dengan harapan program yang di bangun dapat memenuhi uji coba