METODE AGILE TDD

PERBANDINGAN PENGEMBANGAN SISTEM INFORMASI MENGGUNAKAN METODE AGILE DENGAN METODE LAINNYA

Disusun Oleh :

5200411025 - Khrisna Yudha Pratama

5200411140 - Whitnu Nastain

5200411175 - Al-haydar Rizaldy

5200411521 - Eldyan Wasis Aristo Kana

Daftar Isi

Kata Pengantar

Puji Syukur ke hadirat Tuhan yang telah memberikan kami semua rahmat dan hidayah serta dilimpahkannya kesehatan, Kami mampu menyelesaikan makalah tentang Metode “Test Driven Development”

Kami Berharap kedepannya makalah ini dapat dipergunakan oleh adik adik kami dalam menempuh pendidikan Teknik informatika terkhusus mata kuliah Metodologi Desain Perangkat Lunak guna memenuhi kebutuhan ilmu mereka dalam metode pengembangan aplikasi yang efisien dan mudah dipahami

Terima kasih kami ucapkan kepada Dosen kami dan juga teman seperjuangan kelompok kami, berkat mereka makalah ini bisa diselesaikan tepat pada waktunya dan mampu menginspirasi sekaligus memberikan ilmu yang bermanfaat

Makalah ini berisi tentang pengenalan metode agile, perbandingannya dengan metode lain beserta contoh penerapannya pada suatu project yang tidak terlalu kompleks, makalah ini juga jauh dari kata sempurna oleh karena itu Kritik dan Saran sangat kami terima guna memberikan makalah yang berkualitas dan bermanfaat bagi segala generasi

Online, 26 December 2021

Kelompok 4

BAB I

Pendahuluan

Latar Belakang

Pengembangan Sistem Informasi ataupun Aplikasi Memiliki banyak cara dan proses yang harus dilalui, semakin banyaknya Aplikasi kompleks mempengaruhi terhadap proses pengembangan yang harus dilalui untuk menyelesaikan sebuah aplikasi, dalam proses pembuatan Aplikasi diperlukan beberapa cara dan metode yang bisa digunakan untuk menyelesaikan sebuah proses pembangunan aplikasi metode pengembangan Aplikasi yang dapat digunakan bermacam – macam seperti ***Waterfall Model***, ***Incremental Model***, ***RAD Model***, ***JAD Model***, ***Prototyping Model***, ***Component Based Model*** dan ***Agile Model***. Dari model tersebut Agile merupakan salah satu model yang terbaru dan memiliki langkah yang berbeda dengan metode pengembangan perangkat lunak lainnya. Perbedaan tersebut meliputi cara kerja dan langkah – langkah yang pada Agile model namun dalam pembahasan pada makalah kali ini adalah menjelaskan dan mengenalkan bagian dari model agile yaitu Test Driven Development.

Rumusan Masalah

Sesuai dengan latar belakang yang telah dijelaskan di atas, maka identifikasi masalah yang dapat dilihat adalah :

1. Mengenal *Test Driven Development* (TDD)
2. Perbandingan TDD dengan metode lainnya
3. Bagaimana jika menggunakan TDD pada sebuah project

Tujuan

Tujuan dari pembuatan makalah ini adalah untuk mengetahui pengaruh dari menerapkan proses pengujian dalam mengembangkan sebuah sistem informasi atau perangkat lunak dengan menggunakan metode TDD. Adapun beberapa tujuan lain dari pembuatan makalah ini adalah :

1. Mengetahui bagaimana TDD digunakan pada sebuah Project.
2. Memenuhi nilai mata kuliah Metodologi Desain Perangkat Lunak Praktik (MDPLP).
3. Mengenal Test Driven Development
4. Mengetahui Perbedaan Test Driven Development dengan Metode Pengembangan lainnya

BAB II

Pembahasan

Pengertian Test Driven Development

Test driven development (TDD) adalah Proses pengembangan Aplikasi yang lebih mengutamakan testing atau uji coba kode program terlebih dahulu, kemudian di jadikan atau di develop ke program utama dan melacak seluruh proses pengembangan kode program secara menyeluruh dengan terus melakukan proses testing yang berulang, hal ini merupakan kebalikan dari proses pengembangan aplikasi yang membuat aplikasinya terlebih dahulu, kemudian melakukan testing dimana bisa saja baru akan muncul bug yang tidak kita harapkan, dengan menggunakan Test Driven Development yang melakukan Testing code secara menyeluruh, developer dapat meminimalisir terjadinya Bug yang tidak diharapkan.

Pencipta Dari metode development ini adalah **Kent Beck**, seorang Software Engineer asal Amerika yang Mengembangkan atau Menemukan kembali metode ini, Metode Test Driven Development telah dikembangkan dari tahun 2003 dan sempat dilakukan Konferensi Pers Test Driven Development oleh beberapa pengembang yang berhasil membuat para pengembang merasa takjub dengan desain pemodelan yang tidak rumit dan para developer lebih percaya diri dengan adanya metode testing terlebih dahulu coding kemudian

Metode ini sangat berkaitan dengan metode pendahulu nya yaitu metode Extreme Programming yang dimana Kent Beck juga adalah penemunya namun dengan penemuannya terhadap Test Driven Development membuat metode ini memiliki Hal yang unik dan berbeda dari metode pendahulunya dan mampu membuat metode ini bisa bertahan hingga saat ini

Siklus Test Driven Development

Alur atau Siklus dari TDD sangat mudah dilakukan dan bisa terus di ulang hingga seluruh testing yang dilakukan berhasil, dengan metode Test Driven Development yang berarti pengembangan diatur atau di setir oleh Testing, maka proses development program akan terus menerus melakukan testing kemudian melakukan coding, lalu bagaimana siklus TDD secara menyeluruh

1. Membuat sebuah Test

Sebelum Membuat fitur baru dalam sebuah program utama, kita harus melakukan testing dan menulis beberapa baris kode yang berkaitan dengan fitur yang akan dibuat, dengan melakukan Testing kita dapat mengetahui Siklus atau alur kerja dari kode program yang kita buat, dengan menggunakan test driven development, developer bisa memenuhi spesifikasi kebutuhan klien sebelum menulis kode program utamanya, ini merupakan perbedaan yang sangat jelas yang dimana kita terbiasa dengan membuat kode programnya terlebih dahulu baru kemudian kita menemukan Bug sehingga dapat membutuhkan waktu lebih dalam pengembangannya

2. Melakukan Seluruh Testing Terlebih Dahulu

Sebelum menuliskan Kode program yang ingin kita test, kita harus memeriksa terlebih dahulu apakah Unit testing yang kita pakai berjalan dengan baik, biasanya dalam testing pertama kali, kita akan menemukan sebuah Error yang dikarenakan kita belum menulis satupun kode yang digunakan untuk Proses testing

3. Menulis kode yang bisa melewati tahap Test

Barulah kita menuliskan sebuah kode, pastikan Kode yang kita buat benar dan mudah dipahami, dengan tetap menggunakan Konsep KISS (keep it simple, Stupid!) sehingga kode program tidak akan kompleks dan mudah dipahami sistem maupun pengembang lainnya

4. Seluruh Test Berjalan dan Terlewatkan

Dengan kode program yang kita tuliskan mampu melewati tahap uji coba, dengan demikian kita bisa melanjutkannya ke tahap berikutnya

5. Refactor sesuai kebutuhan, dengan mengikuti kode testing dan tetap memastikan fungsionalitasnya terjamin

Kode yang sudah berhasil melalui test kemudian kita lakukan Restrukturisasi dan tetap memastikannya mudah dipahami dan mudah dimodifikasi, Pastikan setelah melakukan restrukturisasi Codingan, kita harus melakukan testing ulang agar menjamin fitur dapat berjalan sesuai kehendak dan keinginan Client dan Developer

Lakukan Berulang

Setelah Berhasil, tentu saja kita akan terus melakukan siklus tadi untuk mencoba atau membuat sebuah fitur baru, proses ini bisa memudahkan kita dalam pengembangan aplikasi dan bisa memangkas Waktu yang digunakan, ketimbang menggunakan metode debugging kemudian mencari asal muasal Error secara manual, hal ini sangat menghabiskan waktu yang sangat banyak dan merugikan kita sebagai developer dalam memenuhi kebutuhan Client

Manfaat TDD

- **Desain yang lebih baik**
  TDD dianggap dalam banyak kasus sebagai proses desain daripada proses testing. Hal ini dikarenakan kode test yang ditulis di awal cenderung lebih kohesif serta mengurangi coupling.

- **Efisiensi**
  Penggunaan TDD akan mengidentifikasikan kesalahan atau error dengan cepat ketika kode baru ditambahkan ke sistem. Selain itu, dengan menerapkan TDD, kita tidak perlu lagi membuat test setelah implementasi kode selesai. Hal ini memberikan keuntungan sendiri, karena membuat test setelah implementasi kode sudah selesai akan jauh lebih susah dibandingkan menulis tes di awal.

- **Test assets**
  Test case yang ditulis secara otomatis dalam TDD adalah aset yang berharga bagi proyek. Ketika ada sebuah peningkatan atau modifikasi pada kode, maka menjalankan unit test secara otomatis dapat mengidentifikasi terjadinya cacat/defect baru pada kode.

- **Mengurangi defect injection**
  Kesalahan lebih rentan terjadi pada saat pemeliharaan kode atau perubahan kode daripada implementasi kode baru. Seringkali, defect injection terjadi pada saat dilakukan pemeliharaan kode. Dengan penggunaan TDD, developer dapat mengetahui apakah perubahan tersebut mengakibatkan kesalahan pada kode yang sudah jadi atau kode yang baru.

BAB III

Contoh Penggunaan TDD

Dalam sebuah Jurnal berjudul "***Test-Driven Development pada Pengembangan Aplikasi Android untuk Memantau COVID-19***" selengkapnya dapat dilihat pada link berikut ini : https://ejournal.bsi.ac.id/ejurnal/index.php/ijcit/article/view/9502/pdf.
Pada jurnal tersebut, pengembangan aplikasi dilakukan dengan metode TDD. Metode TDD menuntut pengembang untuk menentukan terlebih dahulu proses bisnis dan desain perangkat lunak yang dibutuhkan, setelah itu baru menulis kode.

Hal pertama yang dilakukan oleh penulis jurnal atau developer tersebut adalah membuat Test Code menggunakan aplikasi untuk testing fitur yang ingin dibuat olehnya, kemudian melakukan tahap uji kodingan dengan cara menuliskan code tanpa parameter atau bersifat hanya testing sehingga dapat mengetahui apakah fitur dapat berjalan atau tidak.

Selama Proses Testing, ternyata codingannya mengalami kegagalan pada saat proses, sehingga dilakukan pengecekan logic code oleh nya hingga codingan yang dilakukan berhasil melewati tahap testing dan bisa di implementasikan lebih lanjut ke program utama karena proses yang dilakukan selama testing merupakan proses sementara bagaimana fitur tersebut bisa berjalan sesuai spesifikasi yang dibutuhkan client

Setelah dilakukan uji testing dan berhasil, maka proses selanjutnya adalah **Refactoring,** proses ini bertujuan untuk membuat kode yang lebih mudah di maintenance dan dikembangkan lebih lanjut. Developer melakukannya dengan cara merapikan kode yang telah diimplementasikan, serta mengurangi kompleksitas kode sehingga terbentuk kode yang rapi dan terstruktur.

Hasil akhir implementasi dan perancangan dengan menerapkan metode TDD, dapat disimpulkan bahwa fitur-fitur dari perangkat lunak mobile COVID-19 Monitor yang dibangun berjalan dengan baik. Dengan menerapkan metode TDD, ***Developer*** dimudahkan dalam menemukan error ataupun bug dalam proses implementasi kode.

BAB IV

Perbedaan Agile TDD dengan Waterfall

- Menurut model *Waterfall*, pengembangan perangkat lunak harus diselesaikan sebagai satu proyek. Ini kemudian dibagi menjadi beberapa fase yang berbeda, dengan setiap fase hanya terjadi satu kali selama proyek. Metodologi *Agile*, di sisi lain, dapat dilihat sebagai kumpulan dari banyak proyek kecil yang berbeda. Proyek yang tidak lain adalah iterasi fase yang berbeda yang bertujuan untuk meningkatkan kualitas perangkat lunak secara keseluruhan dengan umpan balik dari pengguna atau tim QA.
- Semua fase pengembangan proyek seperti desain, pengembangan, pengujian, dan lain-lain selesai satu kali dalam model *Waterfall*, sambil menerapkan pendekatan pengembangan berulang sebagai bagian dari metodologi *Agile*. Perencanaan, pengembangan, pembuatan prototipe, dan fase pengembangan perangkat lunak lainnya dapat terjadi lebih dari satu kali selama proyek *Agile*.
- Jika Anda ingin menggunakan model *Waterfall* untuk pengembangan perangkat lunak, Anda harus mengetahui dengan jelas semua persyaratan sebelumnya. Tidak ada ruang untuk mengubah persyaratan begitu pengembangan proyek dimulai. Metodologi *Agile* cukup fleksibel dan memungkinkan untuk membuat perubahan pada persyaratan, bahkan setelah perencanaan awal selesai.
- Salah satu perbedaan paling penting antara metodologi pengembangan *Agile* dan *Waterfall* adalah pendekatan mereka sendiri terhadap kualitas dan pengujian. Dalam model *Waterfall*, fase “Pengujian” muncul setelah fase “Bangunan”, tetapi dalam metode *Agile*, pengujian biasanya dilakukan bersamaan dengan pemrograman atau setidaknya selama iterasi yang sama dengan pemrograman.

Perbedaan Agile TDD dengan Prototype

- Dengan metode prototype ini , kita bisa memberikan klien experience yang lebih awal untuk software yang akan digunakan dan memperbaiki serta melengkapinya dengan feedback yang diberikan klien , sedangkan metode agile Scope pengerjaan yang bisa berubah kapanpun dapat menyebabkan kurangnya fokus dari tim development dan menyebabkan isu jika brief yang diberikan tidak jelas.
- Karena dengan metode prototype kita telah mengidentifikasi risiko dan isu yang mungkin terjadi di awal, kita juga dapat mengurangi risiko kegagalan dan di metode agile lebih mengutamakan Komunikasi yang berkelanjutan meningkatkan transparansi antara klien dan tim development.
- Jika di metode agile komunikasi antara klien dan tim pengembang yang intens akan memperkuat hubungan antara kedua belah pihak begitupun di metode prototype juga hampir sama begitu juga namun di metode prototype klien lebih awal ikut terlibat dalam proses pengerjaan yang mungkin bisa saja terlalu ikut campur yang akan mengganggu proses pengerjaan proyek.

- Di metode prototype sudah tersusun jelas pendataan tentang dana dan memang cukup mahal sedangkan di metode agile kurangnya dokumentasi dana meningkatkan risiko miscommunication.

Perbedaan Agile TDD dengan RAD :

- Dengan metode RAD pada kondisi user tidak memahami kebutuhan kebutuhan apa saja yang digunakan pada proses pengembangan perangkat lunak. Sedangkan dengan metode Agile developer lebih mengutamakan rasio kepuasan klien.
- Metode RAD mengikuti tahapan pengembangan sistem seperti umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada (reusable object) sehingga pengembang tidak perlu membuat dari awal lagi dan waktu lebih singkat berkisar antara 60 hari 90 hari , dengan waktu yang cukup lama metode RAD sangat berbeda dengan Metode agile justru lebih memakan waktu yang lebih lama dan tidak pasti jika itu adalah proyek besar.
- Karena dengan metode RAD mempunyai kemampuan untuk menggunakan komponen yang sudah ada dan waktu yang lebih singkat maka membuat biaya menjadi lebih rendah ,dalam menggunakan RAD , dan karena metode agile itu sendiri kurang begitu detail mendata biaya pengerjaan maka akan bisa terjadinya pembengkakan budget atau biaya.

Daftar Pustaka

1. Beck, Kent (2002-11-08). *Test-Driven Development by Example*. Vaseem: Addison Wesley.

2. https://en.wikipedia.org/wiki/Test-driven_development#Individual_best_practices