BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kondisi bisnis dewasa ini beroperasi dalam lingkungan global yang dengan cepat mengalami perubahan. Agar dapat bersaing, mengubah model bisnis untuk merespon *requirement* pasar mutlak diperlukan. Sementara itu, perangkat lunak sebagai bagian tak terpisahkan dari aktivitas bisnis juga harus dapat menyesuaikan. Sering kali model bisnis mengalami perubahan lebih cepat dari lama pengembangan perangkat lunak, ketika perangkat lunak telah selesai dikembangkan, model bisnis sudah mengalami perubahan yang menyebabkan perangkat lunak menjadi kadaluarsa, sehingga tidak memungkinkan untuk menggunakan metode pengembangan formal seperti *waterfall*.

Agile software development merupakan metode pengembangan perangkat lunak yang memungkinkan pengembang untuk terfokus pada perangkat lunak itu sendiri lebih dari desain dan dokumentasinya. Metode pengembangan ini cocok digunakan untuk membangun perangkat lunak yang sering kali mengalami perubahan *requirement* ketika pengembangan sedang berlangsung.

Salah satu praktek *agile software developement* yang dikenal luas adalah *eXtreme Programming* (XP). Dalam XP semua pihak yang terlibat dalam pengembangan perangkat lunak duduk bersama sebagai sebuah tim. Tim akan terdiri dari *customer* yang mendefinisikan *requirement* dan mengatur prioritas. *Customer* akan membutuhkan bantuan dari *system analys* untuk mendefinisikan kebutuhannya. Kebutuhan yang didapatkan dari custumer akan digunakan oleh

Tester untuk mendefinisikan *acceptance test*, dimana *acceptance test* akan dijadikan sebagai sebuah acuan pengembangan bagi *programmer*. Dalam tim XP, diperlukan juga sesesorang *manager* untuk membantu tim agar selalu berada dalam jalur. Manager juga diperlukan untuk menyediakan kebutuhan sumber daya, menangani komuikasi ke pihak eksternal, dan melakukan koordinasi terhadap segala aktivitas dalam tim. Setiap anggota tim XP berkontribusi maksimal sesuai dengan kemampuan yang dimilki tanpa spesialiasi khusus.

Tim XP akan melakukan perencanaan dan rilis berulang-ulang yang mengacu pada tingkat kesulitan *requirement* dan skala prioritas yang diberikan oleh *customer*. Untuk memastikan perangkat lunak selalu sesuai dengan *requirement*, maka pengujian harus selalu dilakukan secara berkelanjutan pada setiap *source code* baru yang diberikan oleh *programmer*. Agar efektif, pengujian haruslah dilakukan secara otomatis, yaitu dengan mendefinisikan unit pengujian sebelum *source code* program dibuat oleh programmer. Unit pengujian berupa *acceptence test* dan *developer test. Acceptance test* untuk menguji *behavior* perangkat perangkat lunak, sedangkan *developer test* untuk menguji *source code* perangkat lunak. Jika pengujian berhasil dilakukan, maka *source code* tersebut akan diintegrasikan pada *source code* utama.

Pendefinisikan unit pengujian sebelum *source code* program dibuat disebut sebagai t*est-first development. Source code* yang ditulis sebelumnya perlu ditinjau ulang kembali untuk menghindari *duplicate code* dan menjaga konsistensi. Penghapusan, penambahan atau pengubahan baris *source code* disebut sebagai *refactoring.* Dalam melakukan *refactoring*, status pengujian haruslah selalu dalam keadaaan *pass,* hal ini disebut sebagai *test-driven development.*

Integrasi secara berkelanjutan yakni dengan melakukan pengujian yang berkelanjutan, memeriksa konsistensi *source code style,* melakukan integrasi *source code* pada setiap skenario dalam sebuah fitur dalam perangkat berhasil dilakukan dan *feedback* yang secara *realtime* dari kondisi pengembangan perangkat lunak yang dikembangkan, disebut sebagai *Continous Integration.*

Melakukan praktek *Continous Integration* tidak harus dilakukan secara manual. Jika perangkat lunak dibangun menggunakan *test-driven development*, maka seharusnya dapat dibangun sistem yang dapat melakukan pengujian otomatis dan langsung melakukan *feedback* kepada tim terhadap status pengembangan perangkat lunak.

Berdasarkan uraian yang disebutkan sebelumnya, sehingga penelitan ini akan mencoba menjelaskan bagaimana membangun sistem yang mampu melakukan integrasi kali source code mengalami perubahan perubahan. Dengan begitu, rutinitas dalam praktek *continous integration* dapat dilakukan secara otomatis. Dengan adanya sistem yang melakukan eksekusi pengujian otomatis, maka akan menjaga fitur-fitur yang telah selesai dikembangkan sebelumnya terbebas dari *bug*.

1.2 Rumusan Masalah

Rumusan masalah dari penilitian ini adalah sebagai berikut:

1. Bagaimana praktek *Continous Integration* dapat mendeteksi kesalahan lebih dini dan menyeluruh melalui integrasi sistem dan pengujian secara berkelanjutan?
2. Bagaimana membangun *Continous Integration Server* yang mampu melakukan menjalankan pengujian secara otomatis dan berkelanjutan?

1.3 Batasan Masalah

Karena luasnya tema yang diangkat, maka dalam penelitian ini, ditetapkan beberapa batasan masalah sebagai berikut:

1. Sistem digunakan untuk melakukan verifikasi dan validasi terhadap aplikasi verbasis web.
2. Sistem operasi yang digunakan untuk menjalankan keseluruhan sistem ini adalah sistem operasi sistem operasi berbasis Unix.
3. Sistem yang dibangun tidak berdiri sendiri, namun dibutuh server virtualisasi server, aplikasi *Source Code Management* , dan juga *enviroment* Unix.
4. Sistem yang dibangun hanya mensupport perangkat lunak yang dibangun menggunakan *test-driven development*

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Menjelaskan bagaimana praktek *Continous Integration* dapat mendeteksi kesalahan lebih dini dan menyeluruh melalui integrasi sistem dan pengujian secara berkelanjutan.
2. Menjelaskan tahapan pengembangan sistem yang dapat melakukan praktek *Continous Integration*.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Bagi peneliti

   Manfaat untuk peneliti yaitu bisa dijadikan acuan dan pertimbangan untuk

penelitian berikutnya yang berkaitan dengan rekayasa perangkat lunak.

2. Bagi pemilik industri perangkat lunak

   Manfaat untuk pemilik industri perangkat lunak yakni mempemudah dalam pengadopsi *eXtreme Programming* dan pengujian perangkat lunak.

1.6 Metode Penelitian

Metode yang digunakan dalam penelitan ini adalah sebagai berikut:

1. Studi Literatur

   Pengumpulan data dengan mengumpulkan referensi yang harus dipelajari yang berkaitan dengan Rekayasa perangkat lunak, *Agile Methods, eXtreme Programming*, *Test-driven Development*, dan lingkungan pengembangan yang akan didukung oleh sistem.

2. Analisis dan Perancangan

   Analisis dalam penelitian ini dimulai dengan menentukan keperluan dan batasan untuk sistem yang akan dibangun dan menganalisis lingkungan pengembangan yang akan di dukung oleh sistem kemudian akan dilakukan perancangan yang bersifat global sebagai fondasi awal pengembangan sistem.

3. Implementasi dan Pengujian

   Implimentasi dan pengujian dilakukan secara berulang-ulang, dan setiap akan dilakukan implementasi akan dibuat kode pengujuan terlebih dahulu, sehingga ketika implementasi kode sudah dilakukan, pengujian sistem akan dilakukan secara otomatis pada setiap fitur yang dibuat.

4. Validasi

   Validasi dilakukan dengan langsung menguji kemampuan sistem yang

dibangun pada industri perangkat lunak yang sebenarnya.

1.7 Sistematika Penulisan

Penulisan skripsi ini tersusun dalam 5 (lima) bab dengan sistematika penulisan sebagai berikut:

BAB I Pendahuluan

Bab ini membahas latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II Tinjauan Pustaka

Bab ini membahas tentang teori-teori yang menjadi acuan untuk pelaksanaan penelitian yang meliputi teori *Test-driven Development, Continous Integration* dan beberapa teori yang berkaitan.

BAB III Metodologi Penelitian

Bab ini membahas tahap-tahap pembangunan perangkat lunak. Secara garis besar terdiri dari tahap pengumpulan data dan tahap pembangunan perangkat lunak.

BAB IV Hasil Penelitian dan Pembahasan

Bab ini membahas tentang hasil penelitian dan membahas masalah-masalah yang telah dirumuskan pada Bab Pendahuluan.

BAB V Kesimpulan dan Saran

Bab ini berisi kesimpulan dan hasil dari penelitian dan saran-saran yang didasarkan pada hasil penelitian yang diperoleh.