Testing Aplikasi Dengan Unit Test Untuk Mengurangi Komplain User Dengan Mengurangi Bug dan Error

Rosa Lujeng Duiri
yaampun_luj@yahoo.com

ABSTAK

Dalam membuat software kualitas juga perlu dijaga. Software yang berkualitas tentu harus tanpa bug dan error. Penulis mengambil contoh aplikasi E-Office yang dikembangkan di PT. Pindo Deli Pulp and Paper Mills. Bug dan error pada aplikasi ini disebabkan karena kurangnya testing secara white box (white box testing).Dalam penulisan ini penulis dengan menggunakan metodologi penelitian yang mengadopsi 7 langkah pemecahan masalah ingin membuktikan bahwa software yang telah dites dengan menggunakan Unit Test, dapat mencegah dan mengurangi bug dan error pada saat software diimplementasikan ke user, yang kemudian berdampak pada menurunnya jumlah komplain user ke divisi Teknologi Informasi.

Kata Kunci : Total Quality Manajemen, Bug dan Error, 7 Langkah Pemecahan Masalah, Unit Test, E-Office

PENDAHULUAN

Untuk menghasilkan suatu sistem aplikasi atau sistem software berkualitas yang sesuai dengan kebutuhan user atau pengguna aplikasi sudah merupakan kewajiban para sistem developer. Sistem software dikatakan berkualitas apabila sesuai dengan harapan user, sesuai dengan bisnis proses, mempermudah kerja user dan memiliki sedikit bug bahkan tanpa bug sama sekali. Ini merupakan suatu tantangan untuk para software developer untuk menghasilkan software tersebut. Demikian juga dialami para pengembang sistem atau para analis sistem dan programmer di PT. Pindo Deli Pulp and Paper Mills.

Sistem dan software developer di divisi Teknologi Informasi PT. Pindo Deli memiliki user yang merupakan orang-orang dari divisi-divisi lain di PT. Pindo Deli. Divisi-divisi yang berada di PT. Pindo Deli antara lain Divisi Finance dan Accounting, Divisi Human Resource, Divisi Electric and Engineering, Divisi Marketing, Divisi MBOS (Management by Olimpic System), Divisi ISO, Divisi EDD, Divisi CAD, Divisi Coustic Soda, Divisi Cast Coat, Divisi NCR, dan lain-lain.

Setelah aplikasi selesai dibuat kemudian aplikasi diimplementasikan ke user yang tidak lain adalah divisi-divisi tersebut diatas. Aplikasi ini diharapkan dapat sesuai dengan kebutuhan user, membantu dan mempermudah pekerjaan mereka. Tetapi pada kenyataannya aplikasi ini masih memiliki bug, output atau hasil yang dikeluarkan tidak sesuai dengan harapan user. Tentu saja hal ini akan menyebabkan komplain user masuk ke divisi Information Technology.

Apabila ada komplain masuk ke divisi IT kemudian akan dianalisa, dicari tahu apa penyebab masalahnya untuk kemudian dicari solusinya. Tetapi tidak jarang error dan bug yang ditemukan sulit untuk dilacak penyebabnya karena apabila dilihat dari logika

dan algoritma program sudah benar. Akhirnya bug dan error ini malah menyulitkan programmer, bahkan memerlukan waktu yang lama untuk memperbaikinya.

Tidak jarang bug dan error tidak bersolusi karena penyebabnya sulit dilacak. Malahan menimbulkan collateral damage yang semakin menyulitkan programmer. Sementara user ingin cepat-cepat error pada aplikasinya segera diperbaiki karena akan memperngaruhi pekerjaannya. Permasalahan seperti inilah yang menyebabkan suatu bug dan error tidak bersolusi, akhirnya untuk membantu user, terpaksa data kemudian diganti atau dimanipulasi sesuai dengan harapan user sementara programmer memperbaiki bug dan error pada aplikasi.

Melihat dari masalah bug dan error yang sulit dilacak maka Penulis tertarik untuk meneliti dan mencari solusi untuk permasalahan tersebut diatas. Penulis beranggapan bahwa sebelum suatu sistem berbasis komputer diimplementasikan ke user diperlukan testing oleh programmer-nya. Tujuan dari pengetesan ini adalah untuk mengetahui apakah alur program sudah berjalan sesuai dengan bisnis prosesnya atau belum, apakah program sudah menghasilkan output sesuai dengan yang diharapkan, dan mempermudah untuk memaintain aplikasi di kemudian hari.

Ada beberapa tool atau alat untuk testing suatu aplikasi, salah satunya adalah NUnit. Nunit adalah tool untuk testing program yang bebas digunakan atau produk yang open source .

Adapun tujuan dari penelitian ini adalah mengurangi komplain user dikarenakan bug dan error pada saat aplikasi diimplementasikan ke user, mendeteksi adanya bug dan error sebelum aplikasi diimplementasikan. memudahkan programmer untuk memperbaiki aplikasi pada saat maintenance, membuat aplikasi sebagai alat penunjang sistem yang sesuai dengan bisnis prosesnya.

TINJAUAN PUSTAKA

Pengertian Kualitas

Menurut Juran, kualitas diartikan sebagai cocok untuk digunakan (fitness for use) dan definisi ini memiliki 2 aspek utama, yaitu :

1. Ciri-ciri yang memenuhi permintaan pelanggan
   Kualitas yang lebih tinggi memungkinkan perusahaan meningkatkan kepuasan pelanggan, membuat produk laku terjual, dapat bersaing dengan pesaing, meningkatkan pangsa pasar dan volume penjualan, serta dapat dijual dengan harga jual yang lebih tinggi.
2. Bebas dari kekurangan
   Kualitas yang tinggi menyebabkan perusahaan dapat mengurangi tingkat kesalahan dan produk gagal, mengurangi inspeksi dan pengujian, mengurangi ketidakpuasan pelanggan, sehingga akhirnya dapat menekan pemborosan.

Meskipun tidak ada definisi mengenai kualitas yang dapat diterima secara universal, dari definisi-definisi yang ada terdapat beberapa persamaan, yaitu :

- Kualitas melebihi usaha memenuhi atau melebihi harapan pelanggan.

- Kualitas mencakup produk, jasa, manusia, proses dan lingkungan.
- Kualitas merupakan kondisi yang selalu berubah (misalnya apa yang dianggap merupakan kualitas saat ini mungkin dianggap kurang berkualitas di masa mendatang).

Berdasarkan elemen-elemen tersebut Goetsch dan Davis mendefinisikan kualitas sebagai berikut :

Kualitas merupakan suatu kondisi dinamis yang berhubungan dengan produk, jasa, manusia, proses dan lingkungan yang memenuhi atau melebihi harapan.

Definisi Total Quality Management

Menurut Ishikawa, TQM adalah perpaduan semua fungsi dari perusahaan ke dalam suatu falsafah yang dibangun berdasarkan konsep kualitas, teamwork, produktivitas, dan pengertian serta kepuasan pelanggan.

Untuk memudahkan pemahamannya, pengertian TQM dapat dibedakan dalam aspek. Aspek pertama menguraikan apa itu TQM dan aspek kedua membahas bagaimana mencapainya.

TQM hanya dapat dicapai dengan memperhatikan beberapa karakteristik berikut ini :

- Fokus pada pelanggan, baik pelanggan internal maupun eksternal.
- Memiliki obsesi yang tinggi terhadap kualitas.
- Menggunakan pendekatan ilmiah dalam pengambilan keputusan dan pemecahan masalah.
- Memiliki komitmen jangka panjang.
- Membutuhkan kerjasama tim (team work).
- Memperbaiki proses secara kontinu.
- Menyelenggarakan pendidikan dan pelatihan.
- Memberikan kebebasan yang terkendali.
- Memiliki kesatuan tujuan.

Adanya keterlibatan dan pemberdayaan karyawan.

Langkah Pemecahan Masalah

Sejak awal tahun 1980-an, teknik pemecahan masalah dengan pendekatan proses Plan Do Check Action sudah mulai dikenal oleh berbagai kelompok peningkatan mutu di perusahaan-perusahaan/organisasi/instansi di Indonesia, terutama yang menjalin hubungan kerjasama dengan perusahaan Jepang.

Pada mulanya Jepang memperkenalkan teknik pemecahan masalah bagi kalangan karyawan pelaksana, dengan proses Delapan Langkah PDCA. Delapan Langkah PDCA merupakan proses kegiatan *continuous improvement*. Perubahan tersebut dimaksudkan agar sistem manajemen mutu lebih luwes dalam menghadapi kecepatan perubahan dunia usaha yang sangat tinggi dengan memperhatikan kebutuhan pelanggan.
Berikut ini adalah langkah dalam 7 langkah pemecahan masalah.

1. Penentuan Judul
2. Menganalisa Penyebab
3. Menguji dan Menentukan Penyebab Dominan

4. Membuat Rencana dan Melakasanakan Perbaikan
5. Meneliti Hasil
6. Membuat Standar Baru
7. Mengumpulkan Data Baru dan Menentukan Rencana Berikutnya

METODOLOGI PENELITIAN

Penelitian dilakukan di PT. Pindo Deli Pulp and Paper Mills, penelitian dengan melibatkan programmer dan support dari divisi Teknologi Informasi, user atau pengguna aplikasi yaitu karyawan dari divisi lain. Penelitian dilakukan dengan membentuk *Small Group* yang terdiri dari *programmer*, *support* dan *user*. Penelitian dilakukan dalam kurun waktu 6 bulan.

Langkah-langkah yang diambil Penulis dalam penelitian ini adalah sebagai berikut :

1. Mengumpulkan data-data komplain user tentang aplikasi E-Office yang masuk ke bagian Teknik Support baik melalui telepon, email, maupun memo tertulis.
2. Mengelompokkan jenis komplain berdasarkan kasusnya.
3. Memilih kasus yang paling sering terjadi.
4. Menganalisa kasus yang perlu ditangani terlebih dahulu.

Dalam menentukan kasus mana yang memiliki prioritas penangan tertinggi penulis menggunakan *Check Sheet* yang berisi beberapa kriteria. Kriteria-kriteria tersebut antara lain :

1. Personal Involvement : Jumlah orang yang terlibat dalam masalah ini.
2. Frequency : Banyaknya masalah yang terjadi.
3. Measureable : Apakah masalah ini dapat dirumuskan.
4. Urgency : Tingkat proritas dari setiap masalah.
5. Dificulties : Tingkat kesulitan penanganan masalah.
6. Solving Time : Berapa lama waktu yang dibutuhkan untuk menyelesaikan masalah.
7. Data Availability : Ketersediaan data pendukung untuk penyelesaian masalah.
8. Problem Definition : Apakah masalah dapat dideskipsikan.
9. Historical Data : Ketersediaan data historical untuk penyelesaian masalah.

Setiap kriteria diberi bobot 1 untuk nilai positif dan 0 untuk nilai negatif, kemudian bobot tersebut dijumlah per jenis masalahnya kemudian digunakan untuk mencari berapa persen prioritasnya.

Menganalisa Penyebab

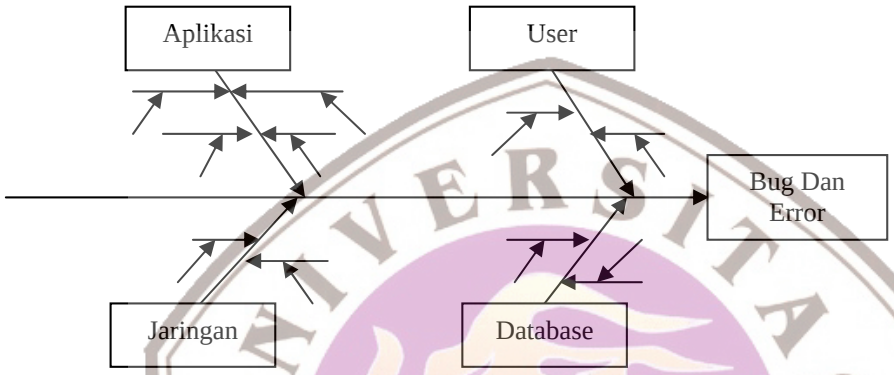
Kegiatan pada langkah ini adalah mencari faktor-faktor yang diduga dapat menjadi penyebab timbulnya masalah, sesuai dengan judul yang ditetapkan pada langkah sebelumnya.

Penelusuran penyebab ini menggunakan *Fishbone* Diagram atau Diagram sebab akibat agar memperoleh gambaran menyeluruh dari suatu susunan hubungan sebab akibat.

Diagram fishbone atau diagram sebab akibat dibuat berdasarkan survei yang dilakukan terhadap user. Masalah yang dihadapi dan ingin diperbaiki dalam hal ini *Bug* dan *Error*

menjadi kepala ikan, kemudian dari hasil survei didapatkan beberapa faktor yang menjadi penyebab bug dan error, faktor-faktor tersebut kemudian menjadi duri-duri utama dari diagram ini. Faktor-faktor tersebut antara lain :

1. User
2. Aplikasi
3. Database
4. Jaringan



Gambar 1. Diagram FishBone

Dari keempat faktor diatas kemudian dicari lagi penyebab-penyebab keempat faktor tadi yang kemudian digambarkan menjadi duri-duri kecilnya.

Faktor penyebab yang ditemukan biasanya lebih dari satu untuk menemukan mana faktor yang paling dominan, Penulis mengambil beberapa responden, yaitu programmer, responden ini kemudian menentukan mana faktor-faktor penyebab yang merupakan faktor paling dominan.

Menguji dan Menentukan Penyebab Dominan

Setelah memilih beberapa penyebab yang dianggap sebagai penyebab dominan, maka langkah selanjutnya adalah upaya untuk mengetahui sejauh mana penyebab-penyebab tersebut benar-benar mempunyai korelasi yang erat dengan persoalan yang timbul. Untuk itu diperlukan pengujian terhadap masing-masing penyebab yang telah dipilih.

Diperlukan data kualitatif untuk menguji keterkaitan antara faktor penyebab dan persoalan yang ditimbulkan. Untuk memperoleh data kualitatif Penulis melakukan pengamatan terhadap source code. Kemudian data yang diperoleh dari hasil pengamatan diperoleh jumlah function dan sub procedure yang tidak ditest kemudian dibandingkan dengan jumlah komplain user.

Untuk menganalisa keterkaitan antar keduanya Penulis menggunakan alat bantu berupa scatter diagram.

Membuat Rencana dan Melaksanakan Perbaikan
Pada langkah ini, tahapan Plan dalam PDCA telah selesai karena prioritas masalah telah diketahui dan penyebab yang sesungguhnya telah terungkap.

Untuk membuat rencana perbaikan menggunakan alat bantu tabel 5W dan 1H. Selanjutnya untuk melaksanakan perbaikan terlebih dahulu dibuat suatu schedule task yang lebih detail. Kemudian perencanaan itu dilaksanakan dengan tetap memonitoring pelaksanaan perbaikan.

Meneliti Hasil
Pada langkah ini Penulis kembali mengamati data komplain user, pengamatan dilakukan dalam jangka waktu 3 bulan setelah aplikasi yang sudah dilakukan pengetesan dengan Unit Tes diimplementasikan.

Dari hasil data pengamatan kemudian dibandingkan dengan data sebelum diimplementasikan aplikasi yang sudah dites dengan Unit Tes. Untuk melihat apakah perbaikan yang dilakukan berdampak positif atau negatif. Penulis memakai alat bantu scatter diagram untuk melihat lebih jelas perbandingan data sebelum dan sesudah tes dengan menggunakan Unit Test.
Dalam meneliti hasil juga harus disebutkan dampak yang ditimbulkan, baik positif maupun negatif. Dampak positif ini yang akan dibuat standarisasi, sedangkan dampak negatif perlu dieleminasi salah satu caranya dengan sistem sumbang saran.

Membuat Standar Baru
Setelah hasil perbaikan didapatkan maka Penulis membuat suatu standar baru dalam mendevelop sistem. Dengan harapan selanjutnya dalam mendevelop sistem dapat mengikuti standar baku yang baru supaya masalah sebelumnya tidak terjadi lagi.

Mengumpulkan Data Baru dan Menentukan Rencana Berikutnya
Setelah pelaksanaan perbaikan berhasil tidak membuat perbaikan berhenti sampai pada masalah ini, melainkan berkembang terus untuk memperbaiki masalah lain yang terkait dengan peningkatan kualitas sistem dan aplikasi.

HASIL DAN PEMBAHASAN

Stratifikasi Masalah Dan Identifikasi Masalah

Dalam penelitian ini penulis melakukan analisa masalah yang terjadi saat aplikasi mulai digunakan oleh user. Berdasarkan hasil survei yang dilakukan Penulis terhadap aplikasi E-Office dan beberapa aplikasi lainnya di PT. Pindo Deli maka didapatkan beberapa masalah yang kemudian meyebabkan komplain user. Berikut ini adalah masalah-masalah yang sering dialami setelah aplikasi resmi digunakan oleh user:

1. Bug dan Error.
2. Gap antara user dan sistem.
3. User tidak puas dengan aplikasi.
4. Keamanan sistem.
5. Penginstalan aplikasi.

Hasil analisa terhadap beberapa faktor penyebab komplain user dapat dilihat pada tabel 1. Tabel dibawah ini dibuat berdasarkan check sheet penelitian yang merupakan hasil dari observasi di lapangan.

Pada tabel 1 dijelaskan analisa masalah berdasarkan 9 kriteria untuk menentukan masalah mana yang harus diselesaikan terlebih dahulu dengan memberi nilai untuk masing-masing kategori. Untuk lebih jelasnya dapat dilihat di tabel 2.

Adapun kriteria-kriteria tersebut adalah sebagai berikut :

1. Personal Involvement : jumlah orang yang terlibat, dibagi menjadi 2 kriteria yaitu : Group (+) dan Personal (-).
2. Frequency : banyaknya masalah yang terjadi, dibagi menjadi : Often (+) dan Rare (-).
3. Measureable : apakah masalah ini dapat dirumuskan, dibagi menjadi 2 kriteria yaitu : Yes (+) dan No (-).
4. Urgency : tingkat proritas dari setiap masalah, dibagi menjadi 2 kriteria yaitu : Yes (+) dan No (-).
5. Dificulties : Tingkat kesulitan penanganan masalah, dibagi menjadi : Medium (+) dan High (-).
6. Solving Time : berapa lama waktu yang dibutuhkan untuk menyelesaikan masalah, dibagi menjadi 2 kriteria yaitu : Hour (+) dan Day (-).
7. Data Availability : Ketersediaan data pendukung untuk penyelesaian masalah, dibagi menjadi 2 kriteria, yaitu : Yes (+) dan No (-).
8. Problem Definition : Apakah masalah dapat dideskipsikan, dibagi menjadi 2 kriteria yaitu : Know (+) dan Unknow (-).
9. Historical Data : Ketersediaan data historical untuk penyelesaian masalah, dibagi menjadi 2 kriteria yaitu : Yes (+) dan No (-)

Tabel 1. Stratifikasi Masalah

No	Problem	Personal Involvement	Frequency	Measureable	Urgency	Dificulties	Solving Time	Data Availability	Problem Definition	Historical Data
		A	B	C	D	E	F	G	H	I
1	Bug dan Error	Group	Often	Yes	Yes	Medium	Hour	Yes	Know	Yes
2	Gap antara user dan sistem	Group	Rare	No	No	High	Day	Yes	Unknow	No
3	User tidak puas dengan aplikasi	Group	Rare	Yes	No	High	Day	Yes	Unknow	No
4	Kemanan sistem	Group	Rare	Yes	No	Medium	Hour	Yes	Know	Yes
5	Penginstalan aplikasi	Group	Rare	No	No	Medium	Hour	No	Know	No

Tabel 2. Identifikasi Masalah

No	Problem	A	B	C	D	E	F	G	H	I	Σ Positif (+)	Σ Negatif (-)	% Positif
1	Bug dan Error	+	+	+	+	+	+	+	+	+	9	0	38%
2	Gap antara user dan sistem	+	-	-	-	-	-	+	-	-	2	7	8%
3	User tidak puas dengan aplikasi	+	-	+	-	-	-	+	-	-	3	6	13%
4	Kemanan sistem	+	-	+	-	+	+	-	+	+	6	3	25%
5	Penginstalan aplikasi	+	-	-	-	+	+	-	+	-	4	5	17%
											24		100%

Dari hasil analisa tersebut kemudian dibuat identifikasi masalah dengan menyusun berdasarkan kelompoknya (positif atau negatif). Kemudian penggolongan ini diberi bobot atau nilai, yaitu 1 untuk positif dan 0 untuk negatif, kemudian dihitung persentase nilai positifnya dengan rumus

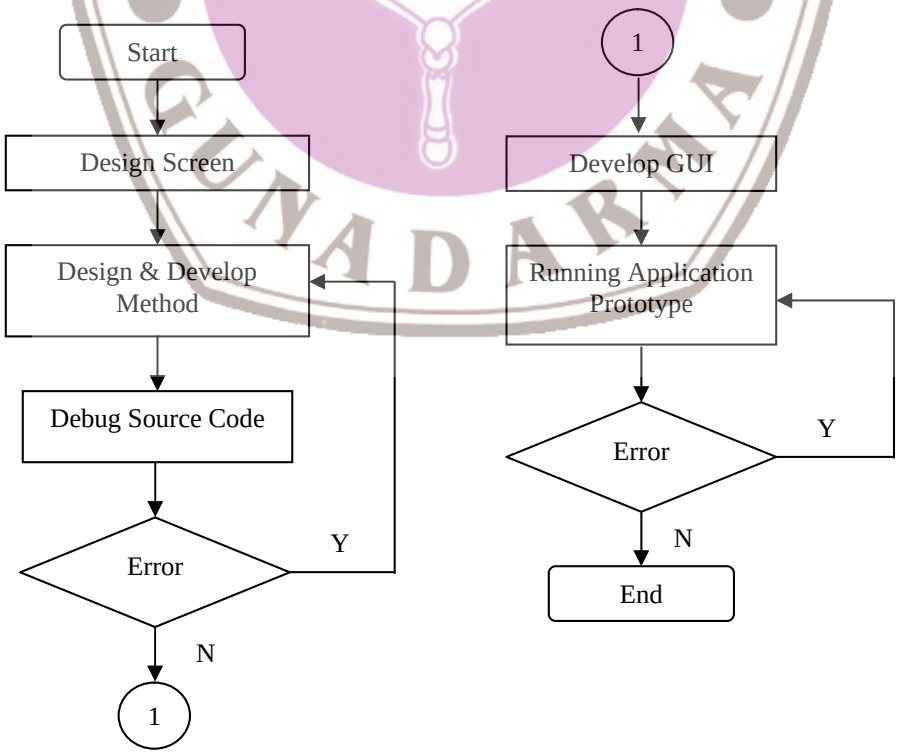
$$(n / \sum n) * 100\%$$

Maka masalah yang diprioritaskan untuk diselesaikan dapat ditentukan.

Dari tabel pengidentifikasian masalah diatas dapat dilihat bahwa persentase masalah bug dan error paling besar yaitu 38%, bug dan error memenuhi seluruh kriteria analisa masalah, sehingga dapat dikatakan bahwa bug dan error merupakan masalah yang paling banyak menyebabkan komplain user. Selain itu bug dan error merupakan masalah utama yang dapat segera ditangani karena, melibatkan banyak orang, sering terjadi, dapat dirumuskan, bersifat urgent, tidak terlalu sulit ditangani, waktu penanganan singkat karena dapat dilakukan dalam hitungan jam, data-data pendukung tersedia, dapat didefinisikan dan data-data historical tersedia.

Kondisi As Is

Pada tahap ini Penulis membuat gambaran bagaimana developer mendevelop aplikasi selama ini. Penggambaran proses perancangan aplikasi berdasarkan kondisi di divisi IT PT. Pindo Deli. Diagram flowchart di bawah ini menggambarkan langkah mendevelop aplikasi sebelumnya atau kondisi As Is dalam mendevelop aplikasi.

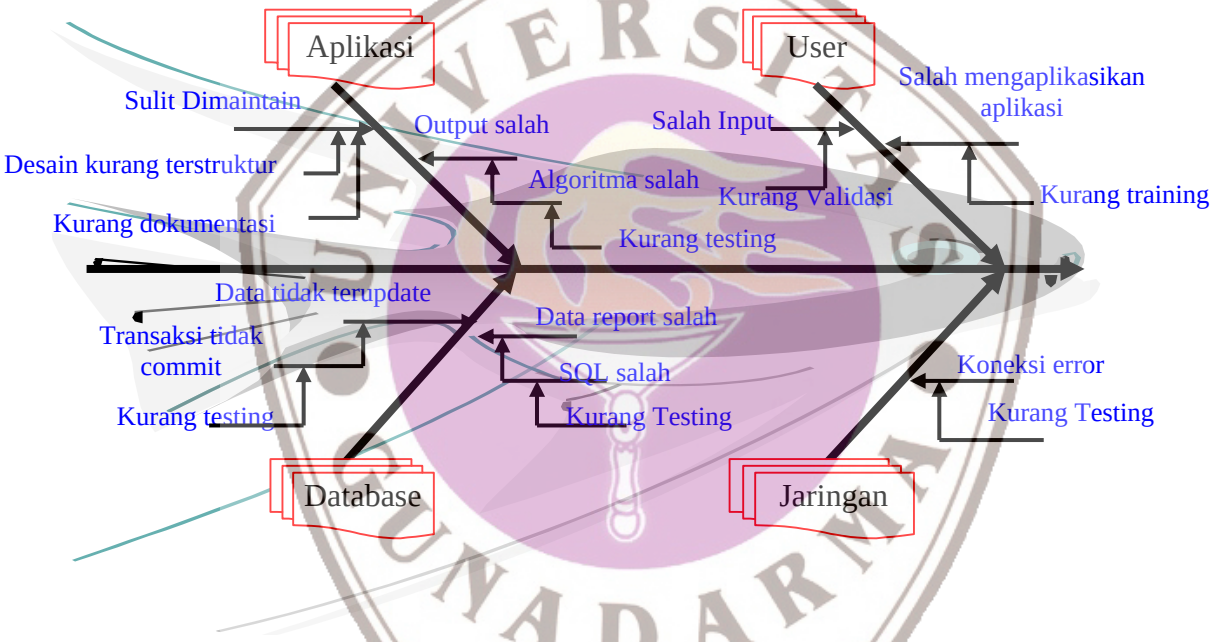


Gambar 2. Diagram Flowchart As Is Testing Aplikasi

Menganalisa Penyebab

Diagram Fishbone menunjukkan bagaimana hubungan sebab akibat sehingga menimbulkan bug dan error. Pada diagram ini digambarkan bahwa terdapat 4 faktor utama penyebab bug dan error yaitu User, Aplikasi, Jaringan dan Database.

Setiap faktor utama memiliki faktor penyebab yang lebih terperinci. Pada perspektif User sering terjadi salah input disebabkan karena kurang validasi, dan salah mengaplikasikan aplikasi disebabkan karena kurang training.
Pada faktor Aplikasi sering terjadi sulit memaintain aplikasi yang disebabkan program kurang terstruktur dan kurang dokumentasi. Selain itu juga sering terjadi salah output yang disebabkan oleh algoritma yang salah, algoritma yang salah disebabkan karena kurangnya testing.



Gambar 3. Diagram Sebab Akibat Identifikasi Masalah Bug dan Error

Pada faktor Jaringan sering terjadi koneksi ke jaringan error yang disebabkan karena kurang testing koneksi.

Pada faktor Database sering terjadi data tidak terupdate disebabkan karena transaksi tidak commit karena kurang ditest. Dan juga sering terjadi output report tidak sesuai disebabkan karena SQL (Structure Query Language) salah, query salah disebabkan karena SQL kurang dites. Semua sebab akibat tersebut yang pada akhirnya menyebabkan terjadinya bug dan error.

Stratifikasi Penyebab Masalah

Dari diagram Fish Bone diatas dapat diambil beberapa akar masalah penyebab bug dan error. Faktor-faktor tersebut antara lain:

1. Kurang Testing
2. Kurang Dokumentasi

3. Design kurang terstuktur
4. Kurang training
5. Kurang validasi

Kemudian dari kelima faktor-faktor tersebut diatas untuk selanjutnya menentukan faktor yang paling dominan adalah dengan cara mengambil 8 responden (dalam hal ini reponden adalah programmer) secara acak untuk melihat manakah dari kelima faktor penyebab tadi yang paling dominan menurut para responsen. Bobot atau nilai prioritas 1 – 5 dengan asumsi 1 yang terrendah dan 5 yang tertinggi. Berikut ini adalah hasil dari responden tersebut.

Table 3. Stratifikasi Penyebab Masalah

No	Cause Factor	A	B	C	D	E	F	G	H	Score
1	Kurang Testing	5	5	5	4	5	5	5	4	38
2	Kurang Dokumentasi	2	2	2	1	2	2	2	1	14
3	Design Kurang terstruktur	4	3	4	3	4	3	3	2	26
4	Kurang training	1	1	1	2	1	1	1	3	11
5	Kurang validasi	3	4	3	4	3	4	4	5	30

Berdasarkan tabel diatas dapat dilihat bahwa hampir semua responden menyatakan bahwa kurang testing merupakan faktor penyebab yang paling dominan. Karena para responden berpendapat bahwa kurang testing merupakan faktor yang paling dominan yang menyebabkan bug dan error pada saat aplikasi diimplementasikan oleh user.

Menguji Faktor Paling Dominan

Pada tahap ini dianalisa bagaimana keterkaitan antara faktor penyebab dominan dengan faktor akibat yaitu Kurang Testing dengan Bug dan Error.

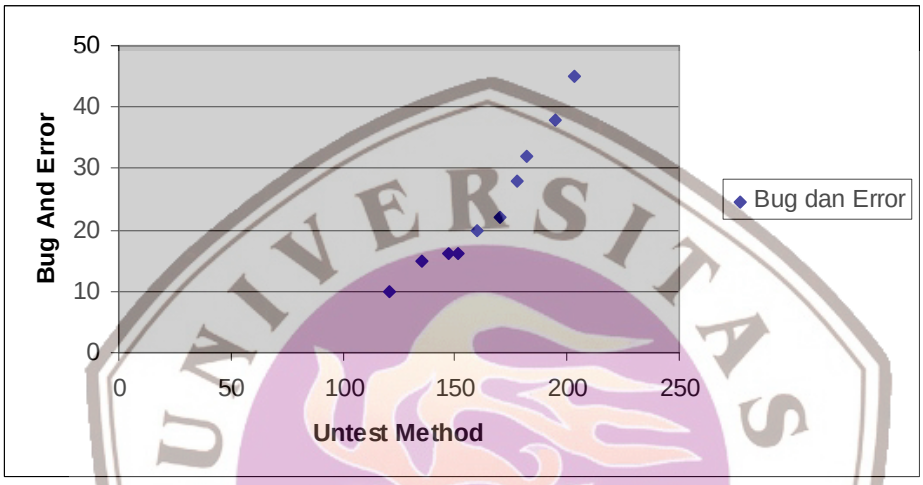
Disini penulis mengambil sampel data secara acak. Pengamatan dilakukan sebanyak sepuluh kali pada method atau function atau sub prosedure pada source code. Penulis mengamati perbandingan jumlah method yang hanya ditest dengan didebug dengan jumlah bug dan error yang ditemukan oleh user. Hasil dari pengamatan dapat dilihat pada tabel dibawah ini.

Table 4. Analisa Penyebab dan Akibat

Case	Kurang Testing	Bug dan Error
1	121	10
2	135	15
3	147	16
4	151	16
5	160	20
6	170	22
7	178	28
8	182	32
9	195	38
10	203	45

Dari data hasil pengamatan ini kemudian dibuatlah grafik scatter dengan tujuan melihat bagaimana hubungan antara faktor penyebab dominan yaitu Kurang Testing dan akibatnya yaitu Bug dan Error.

Pada grafik berikut menunjukkan bahwa semakin banyak method yang kurang dites maka semakin banyak bug dan error yang ditemukan oleh user, dengan demikian kurang testing dapat meningkatkan komplain user.



Gambar 6. Diagram Scatter Perbandingan Penyebab dan Akibat

Membuat Rencana dan Melaksanakan Perbaikan

Pada tahap ini langkah Plan sudah selesai karena pada langkah sebelumnya sudah diketahui permasalahan yang sebenarnya. Pada tahap ini akan dilakukan penyusunan rencana perbaikan dan melakukan uji coba perbaikan.

Untuk membuat rencana perbaikan dalam hal ini adalah menggunakan tabel 5W dan 1H. Rencananya dapat dilihat pada tabel dibawah ini.

Table 5. Tabel Rencana Perbaikan Dengan 5W dan 1H

What	Black Box Testing dan white box testing
Why	Mencegah dan mengurangi komplain user
Where	Semua aplikasi
When	Maret '09
Who	Programmer
How	Membuat unit test dan menjalankan unit test
Target	Divisi IT

Setelah menyusun rencana, berikutnya menyusun target. Dalam menyusun target dilakukan per detail pekerjaan. Adapun pekerjaan detailnya adalah sebagai berikut :

1. Menginstal NUnit.
2. Membuat method dengan Unit Test.
3. Mendebug dan menjalankan program dengan Unit Test.

Tabel berikut akan menjelaskan aksi yang akan dilakukan, material yang dibutuhkan dan targetnya.

Table 6 Tabel Rencana Target Perbaikan

Action	Detail	Material	Date
Membuat dan menjalankan Unit Test	Menginstal NUnit	NUnit 2.4.3	Maret '09
	Membuat Method Unit Test	NUnit.Framework	Maret '09
	Mendebug dan menjalankan program dengan Unit Test		Maret '09

Menginstal NUnit

Detail dari pekerjaan untuk menginstal NUnit dapat dilihat pada tabel berikut ini.

Table 7. Tabel Daftar Pekerjaan Menginstal Unit Test

Action	PIC	Target
Instal Nunit versi 2.4.3	Technical Support	Programmer
Instal Test Driven	Technical Support	Programmer

Membuat Method dengan Unit Test

Setelah selesai memdevelop class kemudian membuat method dengan menggunakan Unit Test. Fungsi dari method-method yang ada di UnitTest adalah untuk mentest apakah method-method yang ada di class berfungsi sebagaimana mestinya, menghasilkan output seperti yang diinginkan.

Selain mentest method-method yang ada di class, unit test juga dapat mentest koneksi ke database.

Berikut ini contoh method-method yang terdapat di Unit Test.

```
<TestFixture()> Public Class PEBDbfUnitTest
    Public Sub New()
        MyBase.New()
    End Sub

    'Performed once before all test begin
    <TestFixtureSetUp()> Public Sub InitOnceBeforeAllTest()
       Server.APPGlobalVariable.glbDBType = Server.DataBaseType.Development
```

```
    End Sub

    'Performed once after all tests are completed
    <TestFixtureTearDown()> Public Sub TearDownOnceAfterAllTest()
    End Sub

    'Performed just before each test method is called
    <SetUp()> Public Sub InitBeforeEachTest()
    End Sub

    'Performed after each test method is run
    <TearDown()> Public Sub TearDownAfterEachTest()
        ' TODO: free unmanaged resources when explicitly called
    End Sub

    ' does not matter what we type the test is not run
    <Test(), Ignore("sample ignore")> Public Sub NotYetFinishedTest()
        Throw New ArgumentException()
    End Sub

    <Test()> Public Sub GetDALHelper_Test()
        Dim _DAL As ExportDoc_PALAPADAL
        Dim _DALHelper As cDALHelper(Of DBaseServer)
        _DAL = New ExportDoc_PALAPADAL
        _DALHelper = _DAL.GetDALHelper()
        Assert.IsNotNull(_DALHelper)
    End Sub
```

Kemudian program didebug untuk melihat methodnya sukses atau gagal dapat dilihat seperti pada pesan di bawah ini. Apabila hasil test sukses akan muncul pesan seperti ini;

```
------ Test started: Assembly: APP.BLL.Eoffice.UnitTest.exe ------

1 passed, 0 failed, 0 skipped, took 4.09 seconds.
```

Apabila test gagal akan muncul pesan seperti ini;

```
------ Test started: Assembly: APP.BLL.Eoffice.UnitTest.exe ------

TestCase 'APP.BLL.Eoffice.UnitTest.ExitPermitTest.FetchData_Test'
failed:
  FetchData do not return valid entity.
  String lengths are both 10. Strings differ at index 5.
  Expected: "0001091730"
  But was:  "0001067362"
  ----------------^
       C:\APPDevelopment\Net2LibraryUnitTest\APP.BLL.Eoffice.UnitTest\ExitPermitTest.vb(
46,0): at APP.BLL.Eoffice.UnitTest.ExitPermitTest.FetchData_Test()

TestFixture failed: Child test failed

0 passed, 1 failed, 0 skipped, took 1.72 seconds.
```

Hasil tes tersebut menunjukkan adanya kesalahan output yang dihasilkan saat fetch data, output seharusnya 0001091730 tetapi data yang dihasilkan 0001067362.

Untuk melihat berapa persen method-method di dalam class yang sudah ditest

dapat dilihat dengan menggunakan Test With Recoverage pada Pop Up Menu.

Mengevaluasi Hasil

Tujuan dari monitoring ini adalah untuk membandingkan antara jumlah komplain user sebelum perbaikan dengan jumlah komplain user setelah perbaikan.

Monitoring mengambil contoh untuk beberapa kasus yang sering masuk ke divisi IT. Kasus-kasus tersebut antara lain:

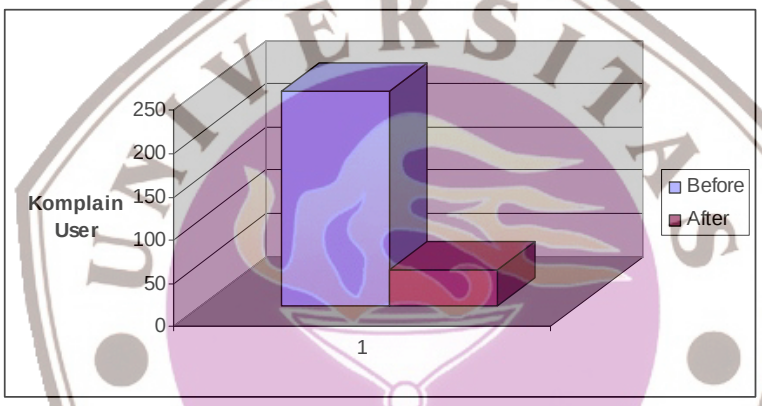
1. Koneksi tidak terbuka (Can't find server).
2. Koneksi error karena data yang ditarik dari database overload.

3. Data tidak terupdate karena data pendukung tidak lengkap, tapi tidak ada pesan kesalahan.
4. Data tidak terupdate karena transaksi tidak commit.
5. Error message / pesan kesalahan sulit dimengerti.

Monitoring terhadap kasus-kasus diatas dilakukan mulai dari sebelum Testing Unit Test dan sesudah Testing Dengan Unit Test yaitu dari bulan Januari sampai Juni.

Table 8. Tabel Total Komplain User Sebelum dan Sesudah Ditest Dengan Unit Test

Month	I	II	III
Before	81	83	85
After	17	18	7



Gambar 7. Diagram Batang Perbandingan Jumlah Komplain User Sebelum dan Sesudah Ditest Dengan Unit Test

Dari kedua hasil pengamatan dapat dilihat bahwa jumlah komplain dari user sesudah menggunakan unit test menurun dibandingkan sebelum menggunakan unit test. Pada bulan April masih ada beberapa komplain itu dikarenakan ada beberapa method hasil prosesnya tidak sesuai harapan, tetapi dengan unit test penyebab bug dan error dapat lebih cepat ditemukan dan segera diperbaiki.

Dari data tersebut maka dapat dihitung tingkat penuruan komplain user setelah ditest dengan menggunakan Unit Test, sebagai berikut :

$$= \frac{\text{Data Actual Sebelum} - \text{Data Actual Sesudah}}{\text{Data Actual Sebelum}} \text{ X } 100\%$$

$$= \frac{(81 + 83 + 85) - (17 + 18 + 7)}{(81 + 83 + 85)} \text{ X } 100\%$$

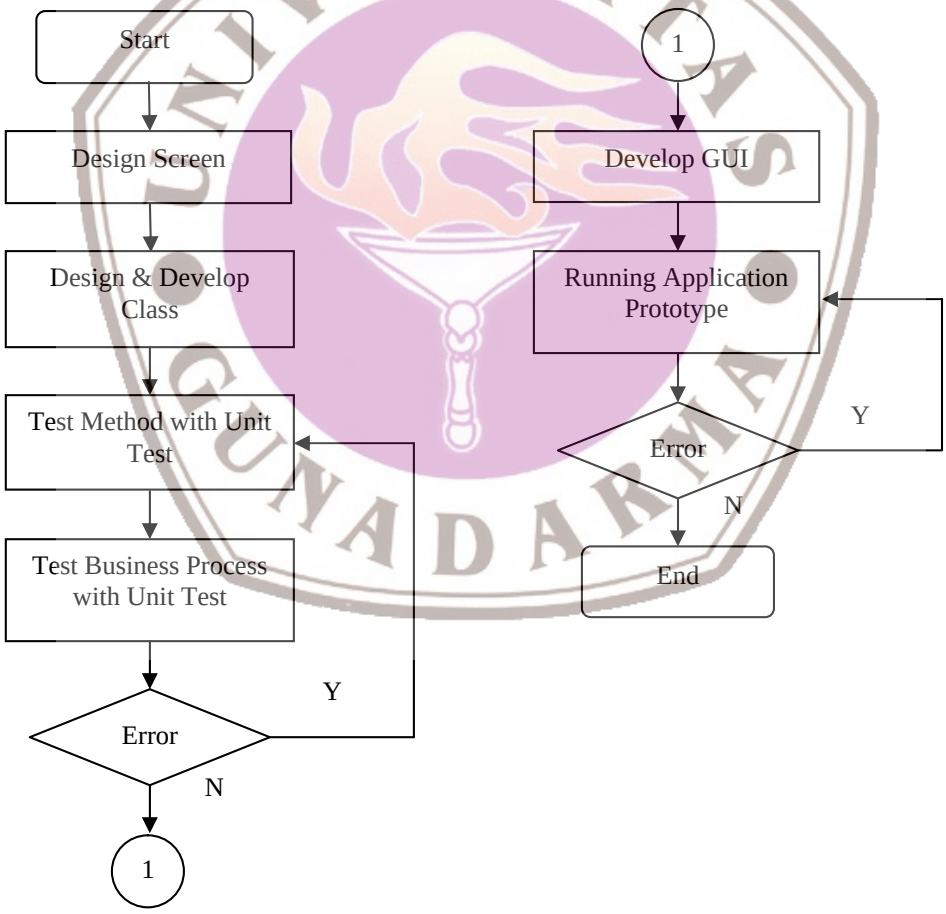
$$= \frac{(249 - 42)}{249} \text{ X } 100\% \quad = 83\ \%$$

Membuat Standar Prosedur Baru

Setelah didapatkan hasil bahwa Unit Test dapat mengurangi user komplain, supaya masalah tidak terulang lagi maka prosedur standar yang lama dalam mendevelop aplikasi E-Office dan aplikasi-aplikasi lainnya perlu diperbaharui. Adapun langkah dalam mendevelop aplikasi adalah sebagai berikut :

1. Desain layar tampilan untuk melihat kebutuhan aplikasi
2. Merancang class (merancang property dan method) untuk membuat properti dan method yang diperlukan aplikasi
3. Mentest method dengan unit test untuk mentest apakah method sudah berfungsi sebagaimana mestinya.
4. Mentest proses sesuai dengan bisnis proses untuk mentest alur aplikasi sesuai dengan bisnis prosesnya.
5. Membuat GUI method untuk mengaplikasikan class di layar tampilan.
6. Mejalankan aplikasi untuk mentest aplikasi secara keseluruhan.

Alur dari langkah-langkah tersebut dapat dilihat pada diagram flowchart dibawah ini.



Gambar 8. Diagram Flowchart To Be Langkah Mendevelop Aplikasi

KESIMPULAN

Dengan dilakukan penelitian ini membuktikan bahwa testing yang selama ini dianggap perbuatan yang sia-sia adalah salah. Unit Test dapat membantu developer untuk

menemukan error yang tidak tertangkap apabila aplikasi hanya didebug. Maka dengan berkurangnya bug dan error otomatis komplain user yang masuk ke divisi Information Technology akan berkurang juga.

Apabila saat implementasi ternyata ditemukan bug dan error developer tidak sulit menemukan penyebabnya, deteksi dapat dilakukan dengan menjalankan ulang Unit Test dengan skenario berdasarkan masalah yang terjadi, untuk selanjutnya ditentukan suatu langkah perbaikan.

Selain itu dengan Unit Test juga dapat dilakukan pengetesan aplikasi program berdasarkan bisnis proses flow. Sehingga menghasilkan aplikasi yang mendukung bisnis proses.

SARAN

Penelitian Penulis dilakukan pada saat aplikasi sudah diimplementasikan ke user, Penulis menyarankan suatu saat apabila ada penelitian sejenis, penelitian dapat dilakukan pada saat aplikasi sedang di-develop. Sering pada saat aplikasi di-develop ada beberapa tambahan dari user yang tentu saja menyebabkan perubahan aplikasi terutama pada sub procedure dan function, maka diperlukan testing apakah penambahan sub procedure dan function baru berdampak terhadap sub procedure dan function yang sudah ada sebelumnya.

Bahkan ada kemungkinan penambahan ini berpengaruh terhadap aplikasi lainnya yang berhubungan dengan aplikasi yang diubah. Sehingga Unit Test tidak hanya dilakukan pada satu aplikasi independen tetapi dapat dibuat di suatu server yang secara khusus melakukan tes aplikasi.

Maka selanjutnya dapat dilakukan penelitian terhadap testing perubahan dan pengembangan aplikasi dengan menggunakan Testing Integration Server. Dimana server ini bertugas mentes beberapa aplikasi yang saling terkait.

Selain itu juga Penulis menyarankan supaya penelitian tidak hanya dilakukan terhadap satu aplikasi saja, tetapi beberapa aplikasi sekaligus terutama aplikasi-aplikasi yang saling berhubungan.

DAFTAR PUSTAKA

Andy Hunt, Dave Thomas. 2003. Pragmatic **Unit Testing In C# with Unit Test**. First Printing, March 2004. United State Of America.

http://www.balancedscorecard.org . **Basic Tools For Improvement - Module 5 – Cause and Effect Diagram.**

http://www.dspace.fsktm.um.edu.my/bitstream/1812/193/8/8-ch6.pdf **. Chapter Six - System Testing and Implementation**

http://www.elucidata.com/refs/sdlc.pdf . **The Software Development Life Cycle For Small to Medium Database Application**

http://www.foundation.phccweb.org/Library/Articles/TQM.pdf **. Total Quality Management - A Continuous Improvement Process**

http://www.the-happy-manager.com/seven-step-problem-solving.html **. The 7 Step Problem Solving Method**

John D. McGregor, David A. Spikes. 2001. **A Practical Guide To Testing Object Oriented Software**. Addison-Wesley, 2001. United State Of Amerika

Komite Penggerak SGA Pusat Asia Pulp And Paper. 1999. **Total Quality Management Manual.** Asia Pulp And Paper, 1999. Serpong.

R. Venkat Rajendran. **White Paper on Unit Testing.** http://www.mobilein.com/whitepaperonunittesting.pdf

Vincent Garspersz. 2001. **Total Quality Management.** PT. Gramedia Pustaka Utama, April 2002. Jakarta.