**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

Nama : **Tegar Riyono Putra**

NRP : **5107 100 009**

Dosen Wali : **F.X. Arunanto, Ir. , M.Sc**

1. **JUDUL TUGAS AKHIR**

***Klasifikasi Severity dari Bug untuk Proyek Perangkat Lunak***

1. **URAIAN SINGKAT**

Manajemen *bug* merupakan salah satu bagian penting dalam proses manajemen proyek perangkat lunak. Untuk memudahkan manajemen *bug*, banyak pengembang menggunakan Bugzilla sebagai kakas bantu pelacakan bug. Dalam aktivitas pembuatan bug baru, seorang pelapor harus memasukkan nilai dari atribut-atribut bug, yang salah satunya adalah atribut *severity*. Penentuan maupun intepretasi nilai *severity* dapat berbeda-beda antar anggota tim pengembang, tergantung pada pengalaman (pemula atau berpengalaman), keahlian (*novice* atau ahli), dan informasi yang tersedia mengenai *bug* itu sendiri. Hal ini dapat menimbulkan kesalahan dalam mengelola *bug*, terutama dalam menentukan waktu yang dibutuhkan untuk penyelesaian suatu bug.

Pada Tugas Akhir ini akan dibangun suatu perangkat lunak yang dapat memberikan rekomendasi *severity* dari *bug* secara otomatis kepada pelapor. Sistem ini akan dibangun dengan menggunakan teknik-teknik penggalian data yang berbasis teks. Model prediksi yang dibangun didasarkan kepada data sejarah bug sebelumnya dari suatu proyek perangkat lunak yang sama. Diharapkan dengan adanya fitur ini pelapor dapat terbantu dalam menetapkan *severity* dari suatu *bug* dengan lebih akurat.

1. **LATAR BELAKANG**

Rekayasa perangkat lunak adalah disiplin teknik yang berhubungan dengan semua aspek dari proses produksi perangkat lunak. Rekayasa perangkat lunak diharuskan mengimplementasikan sebuah pendekatan yang sistematis dan terorganisir dalam kerja mereka dan menggunakan kakasdan teknik yang sesuai berdasarkan masalah yang akan dipecahkan, batasan-batasan dan sumber yang tersedia[1]. Oleh karena itu, rancang bangun perangkat lunak dapat mencakup penelitian, pengembangan baru, modifikasi, *reuse, reengineering*, pemeliharaan atau aktivitas lain yang berdampak pada perangkat lunak. Secara garis besar, cakupan-cakupan di atas masuk kedalam daur hidup perangkat lunak. Ada enam fase dalam daur hidup perangkat lunak yaitu analisis dan spesifikasi kebutuhan, perancangan, pemrograman, pengujian, dan pemeliharaan. Diantara keenam fase tersebut fase pemeliharaan mempunyai pengaruh yang paling besar karena 67% biaya dari keseluruhan daur hidup perangkat lunak ditujukan untuk pemeliharaan[2]. Oleh karena itu diperlukan suatu sistem manajemen pemeliharaanyang baik agar biaya yang dikeluarkan dapat ditekan.

Sistem pelacakan *bug* adalah aplikasi perangkat lunak yang didesain untuk membantu penjaminan kualitas dan *programmer* untuk selalu memantau *bug* (fase *pemeliharaan*)dari perangkat lunak yang telah dilaporkan didalam kerja mereka. *Bug* yang ditemukan oleh pengguna dapat dilaporkan secara langsung melalui sistem ini. Ada beberapa kolom/ field yang harus diisi oleh pelapor, misalnya tingkat kerusakan bug/ *severity*, kelakukan kerusakan dari perangkat lunak, cara mendapatkan *bug*, dan lainnya.

Atribut-atribut bug tersebut haruslah diisi dengan nilai atau isian yang sesuai oleh pelapor. Namun disatu sisi kolom *severity* menjadi suatu masalah jika dihadapkan pada beberapa hal. Yang pertama adalah jika pelapor mempunyai latar belakang yang beragam. Satu *bug* dilaporkan dengan nilai major oleh seorang *tester* dan nilai minor oleh *tester* yang lain. *Tester* yang kurang berpengalaman menganggap bahwa *bug*  tersebut tidak berdampak banyak terhadap fungsionalitas perangkat lunak, sementara *tester* yang berpengalaman melihat ada beberapa fungsi perangkat lunak yang tidak berjalan dengan baik. Masalah yang lain adalah ketika pelapor dihadapakan pada waktu singkat sehingga tidak dapat melakukan pengecekan secara menyeluruh. Oleh karena itu diperlukan suatu rekomendasi kepada pelapor secara otomatis agar dapat membantu dalam pemberian nilai *severity* yang sesuai.

SEVERIS adalah salah satu Sistem pelacakan bug yang digunakan NASA dalam melakukan manajemen *bug*-nya[3]. SEVERIS menggunakan *standart text mining* dan *machine learning* yang diujicobakan pada beberapa set laporan *bug* dalam proses klasifikasi *severity*-nya. Studi kasusnya menggunakan data yang berasal dari *NASA’s Project and Issue Tracing System* (PITS). Pada ujicoba yang dilakukan terhadap 5 set data didapatkan bahwa SEVERIS adalah prediktor yang baik untuk masalah *severity*, mudah digunakan dan efisien.

Dalam proposal ini penulis akan mengusulkan membangun perangkat lunak yang dapat memberikan rekomendasi *severity* kepada pelapor dengan menggunakan beberapa metode *data mining*.

1. **RUMUSAN MASALAH**

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut :

1. Bagaimana mengubah data *bug report*  menjadi file berjenis ARFF
2. Bagaimana memodelkan klasifikasi *severity* dari *bug* berdasarkan atribut-atribut lain dari *bug*.
3. Bagaimana memproses *bug report* agar dapat menghasilkan rekomendasi sesuai dengan model klasifikasi *severity*.
4. **BATASAN MASALAH**

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut :

* + - 1. Aplikasi ini menggunakan data yang berasal dari *repository* Bugzilla untuk proyek pengembangan perangkat lunak Mozilla mulai tanggal 1 januari 2010 sampai dengan 31 Desember 2010.

1. **TUJUAN TUGAS AKHIR**

Tugas Akhir ini memiliki tujuan yang rinciannya dapat dituliskan sebagai berikut :

1. Membuat aplikasi rekomendasi *severity* dari *bug*
2. Memodelkan klasifikasi *severity* dari *bug*.
3. **MANFAAT TUGAS AKHIR**

Manfaat yang diharapkan dari perancangan aplikasi Tugas Akhir ini adalah agar dapat memberikan rekomendasi nilai *severity* dari bug kepada pihak pelapor. Dengan begitu diharapkan *bug report* mempunyai tingkat kebenaran yang berdasar.

1. **TINJAUAN PUSTAKA**

**Bug**

*Bug* merupakan sesuatu yang seharusnya tidak dilakukan oleh perangkat lunak atau perangkat lunak yang tidak melakukan seperti yang seharusnya dilakukan[4]. *Bug* tersebut dibagi menjadi beberapa macam, yaitu

### Arithmetic bugs

Merupakan *bug* yang terjadi karena kesalahan aritmatika, contohnya ketika suatu bilangan dibagi 0.

### Logic bugs

Merupakan *bug* yang terjadi karen kesalahan logika kode program. Sebagai contoh adanya perulangan tek terbatas.

### Syntax bugs

Merupakan *bug* yang terjadi karena kesalahan penulisan kode program.

### Resource bugs

*Bug* yang terjadi karena sumber daya yang ada tidak dapat menginbangi kebutuhan yang dibutuhkan, seperti terjadinya kekurangan memory untuk *buffering*.

### Multi-threading programming bugs

*Bug* yang terjadi pada program yang berjalan secara multi-threading, contohnya terjadi deadlock.

### Interfacing bugs

Merupakan *bug* yang terjadi pada antar muka yang ada seperti pada API, pada penanganan perangkat keras, dan protokol yang digunakan.

### Teamworking bugs

*Bug* yang terjadi saat perangkat lunak dibangun dalam tim. Semakin besar tim yang membangun semakin besar pula kemungkinan terjadi bug.

Semaiki besar perangkat lunak yang dibangun maka semakin besar pula kemungkinan munculnya *bug*. Selain itu juga dalam aktifitas evolusi, diperlukan mencatat perkembangan suatu *bug*. Catatan tersebut diperlukan agar nantinya tidak muncul *bug*-*bug* lain yang disebabkan adanya kesalahan dalam penanganan suatu *bug*. Untuk itulah perlu dikembangkan sebuah sistem untuk mencatat *bug* yang terjadi. Sistem tersebut disebut dengan sistem pelacakan *bug*.

**Sistem Pelacakan Bug**

Merupakan perangkat lunak yang dirancang untuk membantu mancatat dan melacak *bug* perangkat lunak yang dilaporkan, baik oleh pengembang, penjamin kualitas, maupun anggota tim yang lain. Ada juga sistem pelacakan *bug* yang mengizinkan pengguna untuk melaporkan *bug* secara langsung. Salah satu komponen utama dari sebuah sistem pelacakan *bug* adalah database, database tersebut digunakan untuk mencatat *bug*-*bug* tyang ditemukan beserta atribut-atribut yang dimilikinya. Atribut-atribut tersebut antara lain tanggal ditemukannya bug, penemu bug, *severity bug,* dan lain sebagainya. Nantinya atribut-atribut ini digunakan untuk mengetahui sejarah suatu bug. Dalam perkembangannya,. sistem pelacakan *bug* sudah banyak digunakan dalam dunia pengembangan perangkat lunak, bahkan sistem ini diintegrasikan dengan sistem manajemen proyek.

**Bugzilla**

Bugzilla adalah salah satu contoh dari sistem pelacakan bug. Sistem ini mengijinkan pengembang individu atau grup untuk mengawasi *bug* yang muncul di dalam prodouk mereka secara efektif. Bugzilla dapat digunakan untuk melacak *bug* dan perubahan kode, berkomunikasi dengan anggota tim, mengirim dan meninjau *patches* dan *management quality assurance*(QA). Informasi *bug* yang disimpan meliputi beberapa atribut antara lain

* Deskripsi *bug*

Berisi ringkasan akibat dari *bug* atau cara mendapatkan *bug* tersebut.

* Status *bug*

Status *bug* dalam sistem yang meliputi *unconfirmed, new, assigned, ressolved, reopen, verified, closed*.

* Kata kunci *bug*

Berisi daftar kata penting yang berkaitan dengan *bug*.

* Pelapor *bug*

Berisi informasi tentang pelapor *bug*.

* Waktu pelaporan *bug*

Berisi informasi waktu kapan *bug* dilaporkan ke sistem.

* *Severity bug*

Berisi informasi kehebatan yang diakibatkan *bug*.

*Bug* yang ada dalam Bugzilla memiliki siklus/ diagram alir yang menggambarkan status *bug* dari awal dilaporkan sampai *bug* tersebut diselesaikan. Gambar 1 menunjukkan alur hidup dari *bug* yang dimulai dari status *unconfirmed* sampai *closed*.



Gambar 1. Siklus bug dalam Bugzilla [5]

Status *unconfirmed* merupakan status awal dari *bug* *report* ketika pertama kali dilaporkan yang menunjukkan bahwa *bug report* tersebut belum mendapat konfirmasi dari pihak *supervisor/admin*. Setelah diterima sebagai suatu *bug report* yang benar maka statusnya akan menjadi *new*. Status *assigned* menunjukkan bahwa *bug report* sudah ditugaskan kepada pihak *developer* sekaligus menunjukkan bahwa *bug* sedang diperbaiki oleh *developer*. Setelah diperbaiki maka statusnya menjadi *resolved*, kemudian akan berubah menjadi *verified* dan *closed*  jika sudah diverifikasi dan dinyatakan selesai.

*Bug* yang ada di dalam Bugzilla dapat diklasifikasikan *severity*-nya dengan menggunakan beberapa metode penggalian data.

**Penggalian Data**

Penggalian data atau *data mining* didefinisikan sebagai proses untuk menemukan pola pada data yang berjumlah sangat besar dan proses tersebut berjalan secara otmatis atau semi-otomatis [6]. Dari dataset yang ada, menggunakan metode-metode penggalian data, akan ditemukan model yang nantinya dapat digunakan untuk memprediksi sesuatu. Dataset tersebut terdiri atas beberapa atribut yang nantinya akan dicari komposisi dari masing-masing atribut tersebut kepada model yang dibuat.

Dalam tugas akhir ini akan digunakan beberapa metode penggalian data, yaitu 0-R, 1-R, C4.5 Decision Tree, Naive Bayes, Logistic Regression. Masing-masing metode tersebut digunakan untuk menemukan model dari data repositori bugzilla. Selanjutnya model tersebut dapat digunakan untuk memberikan rekomendasi *severity* dari *bug* kepada pelapor. Salah satu sistem sejenis yang juga memberikan rekomendasi severity serupa adalah Severis.

**Severis**

Di dalam suatu sistem misi yang kritis, seperti misi-misi yang dibangun oleh NASA, sangatlah penting bagi *test engineer* untuk dapat mengenali *severity* dari tiap masalah yang mereka identifikasi selama proses *testing*. Pemberian *severity* yang tepat sangat penting untuk pengalokasian sumber yang sesuai dan penjadwalan aktifitas pemeliharaan serta *testing* tambahan. Sementara itu *severity* sangat dipengaruhi oleh tingkat pengalaman *test engineer* dan lama waktu yang dihabiskan dalam menangani tiap masalah.

Oleh karena itu diperlukan suatu *tools* yang mampu memberikan rekomendasi *severity* dari *bug* agar *bug report* mempunyai nilai kebenaran yang berdasar jelas. Salah satu tools yang memenuhi kebutuhan tersebut adalah SEVERIS(*SEVERity Issue assesment*). SEVERIS menggunakan *standart text mining* dan *machine learning* yang diujicobakan pada satu set laporan *bug*. Studi kasusnya menggunakan data yang berasal dari *NASA’s Project and Issue Tracing System* (PITS). Pada ujicoba tersebut disimpulkan bahwa SEVERIS adalah prediktor yang baik untuk masalah *severity*, mudah digunakan dan efisien.

Nasa menggunakan skala 1-5 untuk menilai *severity* dari suatu masalah, dari yang terburuk sampai yang teringan. Untuk masalah yang berkaitan dengan *robotic* dan *human-rated missions* digunakan skala yang berbeda. Tabel 1 menunjukkan klasfikasi *severity* untuk *robotic missions* dan *human rated missions*. *Robotic missions* menyangkut segala aspek yang berhubungan dengan tugas yang dilakukan oleh robot ,sedangkan *human rated missions* berhubungan dengan tugas yang dilakukan oleh manusia.

Tabel 1. NASA's severity scores [3]

|  |
| --- |
| Severities for robotic missions |
| Severity 1: Prevent the accomplishment of an essential capability; or jeopardize safety, security, or other requirement designated critical. |
| Severity 2: Adversely affect the accomplishment of an essential capability and no work-around solution is known; or adversely affect technical, cost or schedule risks to the project or life cycle support of the system, and no workaround solution is known. |
| Severity 3: Adversely affect the accomplishment of an essential capability but a work-around solution is known; or adversely affect technical, cost, or schedule risks to the project or life cycle support of the system, but a work-around solution is known. |
| Severity 4: Results in user/operator inconvenience but does not affect a required operational or mission essential capability; or results in inconvenience for development or pemeliharaan personnel, but does not affect the accomplishment of these responsibilities. |
| Severity 5: Any other issues. |
| Severities for human-rated missions |
| Severity 1: A failure which could result in the loss of the human rated system, the loss of flight or ground personnel, or a permanently disabling personnel injury. |
| Severity 1N: A failure which would otherwise be Severity 1 but where an established mission procedure precludes any operational scenario in which the problem might occur, or the number of detectable failures necessary to result in the problem exceeds requirements. |
| Severity 2: A failure which could result in loss of critical mission support capability |
| Severity 2N: A failure which would otherwise be Severity 2 but where an established mission procedure precludes any operational scenario in which the problem might occur or the number of detectable failures necessary to result in the problem exceeds requirements. |
| Severity 3: A failure which is perceivable by an operator and is neither Severity 1 nor 2. |
| Severity 4: A failure which is not perceivable by an operator and is neither Severity 1 nor 2, nor 3. |
| Severity 5: A problem which is not a failure but needs to be corrected such as standards violations or pemeliharaan issues. |

SEVERIS menggunakan ekstraksi otomatis dan analisis dari deskripsi tekstual dari laporan masalah yang diperoleh dari PITS. SEVERIS menggunakan beberapa teknik *text mining* untuk mengekstrak fitur yang relevan dari tiap laporan, sementara teknik *machine learning* digunakan untuk memberikan fitur-fitur nilai *severity* yang layak, berdasarkan klasifikasi dari laporan yang ada.

Gambar 2 menunjukkan bagaimana *SEVERIS* berinteraksi dengan analis manusia atau atasannya. SEVERIS mengecek valid tidaknya *severity level* yang diberikan pada suatu isu/ masalah dengan tahapan-tahapan berikut :



- Setelah melihat isu dari beberapa artifak, seorang analis membuat beberapa catatan berupa text dan member nilai *severity* X.

- SEVERIS mempelajari pola baru dari isu dengan melihat catatan {*notes, severity X*} melalui suatu modul training yang meliputi :

1. memperbarui SEVERIS *beliefs*

2. memperkirakan berapa nilai *self confidence* yang mungkin dimiliki supervisor terhadap kesimpulan SEVERIS

- SEVERIS melakukan *review* terhadap catatan analis dengan *learned knowledge* dan menghasilkan nilai *severity Y* versinya.

- Supervisor menerima laporan *severity* dari analis dan SEVERIS, jika ternyata nilainya berbeda maka supervisor dapat melakukan review terhadap catatan analis. Selain itu supervisor juga dapat melakukan review terhadap nilai *self confidence* untuk menentukan apakah akan mempercayai rekomendasi SEVERIS atau tidak.

Gambar 2. Diagram alir dari SEVERIS

Satu isu yang menantang dalam SEVERIS ini adalah bahwa *standard machine learners* bekerja dengan baik untuk keadaan dimana hampir semua isi catatan menggambarkan keseluruhan maksud isu tersebut. Namun pada kenyataannya aplikasi *text mining* harus memproses ribuan kata yang unik dimana di dalam paragraph hanya terdapat beberapa kata saja. Oleh karena itu perlu suatu pengurangan dimensi yang dilakukan dengan cara *tokenization, stop word removal, stemming, tf\*idf,* dan *infogain.* Setelah proses pengurangan dimensi dilakukan proses *rule learning* dengan menggunakan Cohen’s RIPPER *rule learner*  versi Java.

Uji coba SEVERIS dilakukan terhadap 5 set data yang berasal dari PITS dimana proses *rule learning*-nya menggunakan metode *10-ways* *cross validation*. Setelah dilakukan evaluasi terhadap 3 istilah dan 100 istilah teratas pada 5 set data dengan menggunakan 3 pengukuran yang umum dalam *Information Retrieval* *(recall, precision* dan *f-measure*) didapati bahwa nilai performa semakin baik sebanding dengan jumlah istilah yang digunakan.

**Evaluasi *f-measure* 3 istilah (X) dan 100 istilah (Y) teratas (f=X%-Y%)**

pitsA, untuk isu dengan severity=2, f = 78-86%;

pitsA, untuk isu dengan severity=3, f = 68-71%;

pitsB, untuk isu dengan severity=3, f = 71-71%;

pitsC, untuk isu dengan severity=3, f = 79-92%;

pitsC, untuk isu dengan severity=4, f = 86-92%;

pitsD, untuk isu dengan severity=3, f = 98-98%;

pitsD, untuk isu dengan severity=4, f = 91-91%;

pitsE, untuk isu dengan severity=3, f = 65-70%;

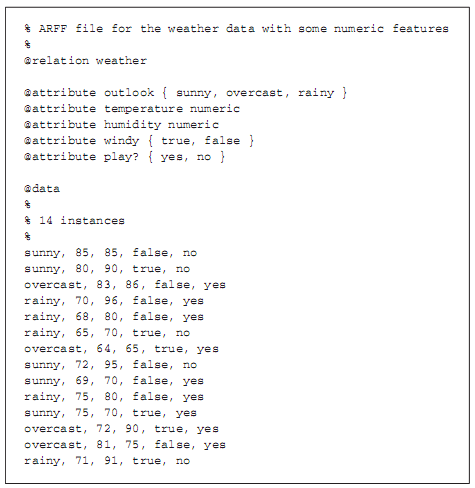
pitsE, untuk isu dengan severity=3, f = 71-75%;

**Pustaka Weka**

Weka adalah perangkat lunak yang mempunyai banyak koleksi algoritma *machine learning* yang digunakan dalam proses *data mining.* Algoritma yang ada dapat diaplikasikan secara langsung pada set data atau dipanggil dari kode Java buatan sendiri. Weka menyediakan *tools* untuk *pre-processing* data, klasifikasi, regresi, *clustering*, *association rules* dan *visualization*. Selain itu, Weka juga dapat digunakan untuk membangun skema *machine learning* baru. Perangkat lunak dapat mengimplementasikan pustaka Weka melalui *Weka API (Aplication Programming Interface)* yang mencakup beberapa proses diantaranya membuat set data di *memory, loading and sabing data*, *filtering, classifying, clustering, selecting attributes, visualization,* dan *serialization*[7]. Dalam melakukan proses *data mining,* Weka menggunakan format file ARFF dalam *input file* maupun *output file*-nya.

**ARFF**

ARFF(Attribute Relation File Format) adalah suatu file teks ASCII yang berisikan sebuah daftar contoh yang berisikan satu set atribut. File ARFF dibagi menjadi 2 bagian, bagian *Header Information* dan *Data Information*. *Header Information* berisikan nama relasi, daftar atribut(kolom dari data) dan tipenya. Sedangkan *Data Information* berisikan data dari file berdasarkan kolom pada bagian *Header Information*.



Gambar 3. Contoh Isi *file* ARFF[7]

1. **RINGKASAN TUGAS AKHIR**

Pada Tugas Akhir ini akan dibangun perangkat lunak yang dapat memberikan rekomendasi *severity* dari *bug* secara otomatis kepada pihak pelapor. Rekomendasi yang dihasilkan ditujukan untuk membantu pihak pelapor dalam memberi nilai *severity* pada *bug report* yang akan dibuat. Diharapkan dengan adanya rekomendasi, nilai *severity* yang dilaporkan sesuai sehingga proses perbaikan bug dapat berjalan dengan baik. Agar perangkat lunak dapat memberikan rekomendasi diperlukan suatu pemodelan klasifikasi *severity* dari *bug* berdasarkan beberapa metode penggalian data.

Pemodelan klasifikasi *severity* dilakukan terhadap *bug report* yang berasal dari repositori Bugzilla. Ada banyak attribut dalam *bug report,* namun hanya beberapa saja yang digunakan untuk pemodelan, antara lain *product, component, status, resolution, assigned\_to, qa\_contact, summary, rep\_platform, op\_sys, version, priority, severity, reporter\_accessible,* dan *cclist\_accessible*.

Sesuai dengan uraian diatas, pembuatan aplikasi ini akan menggunakan data dari repositori bugzilla sebagai data acuan pembentukan model. Kemudian untuk mendapatkan modelnya, pembuatan aplikasi ini menggunakan pustaka Weka, sebuah pustaka yang menyediakan beberapa teknik penggalian data, sebagai alat bantu untuk melakukan penggalian data. Namun untuk dapat melakukan penggalian data, data yang diolah harus diatur dengan format ARFF sesuai dengan permintaan dari Weka. Agar sesuai dengan Weka yang dibangun di atas platform Java, aplikasi ini akan dibangun pula di atas platform Java.

Aplikasi ini akan mengambil data dari repositori bugzilla dengan menggunakan WebService XMLRPC yang merupakan bagian dari Bugzilla API. Data-data tersebut disimpan dalam file yang berformat ARFF agar dapat diolah. Data ini akan digunakan untuk membuat model klasifikasi *severity* dari *bug*. Data tersebut dapat diperbaruhi sesuai keinginan pengguna.

Setelah data didapatkan, aplikasi menjalankan beberapa teknik penggalian data untuk memodelkan klasifikasi *severity*. Teknik-teknik yang digunakan antara lain *tokenization, stop word removal, stemming*, tf\*idf, *info gain* dan *rule learning*. Model yang dihasilkan kemudian disimpan dalam database.

Sebagai gambaran, aplikasi akan dibangun mengguakan alur proses seperti yang terlihat pada gambar 3. Ada dua aktor mengakses aplikasi, yaitu admin dan pelapor. Admin adalah aktor yang bertugas untuk menentukan apakah data dan model perlu diperbarui atau tidak. Pelapor adalah aktor yang melaporkan *bug report* dan menerima rekomendasi *severity* dari *bug report* yang telah dimasukkan.



**Gambar 3. Alur Proses Sistem Rekomendasi Severity dari Bug**

1. **METODOLOGI**
2. **Penyusunan Proposal Tugas Akhir**

Tahap awal untuk memulai Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan pembuatan aplikasi untuk klasifikasi *severity* dari *bug* untuk proyek perangkat lunak.

1. **Studi Literatur**

Pada tahap dilakukan pencarian dan pembelajaran berbagai literatur yang berhubungan dengan rumusan masalah, metode-metode *data mining* yang sesuai, metode *rule learning* yang sesuai, isu/ kasus yang sejenis, *bug tracking sysem API’s*, manual *data mining application*.

1. **Analisis**

Pada tahap ini dilakukan perancangan arsitektur perangkat lunak yang berorientasi objek (*Obejct Oriented Programming*), perancangan *use case model*, *sequence diagram, class diagram*. Skema rancangan arsitektur sistem perangkat lunak rekomendasi link favorit ini adalah sebagai berikut

1. **Implementasi**

Pada tahap ini dilakukan proses pemrograman perangkat lunak dengan menggunakan bahasa pemgograman Java dengan memanfaatkan pustaka Weka. Metode yang digunakan adalah *extreme programming.* *Extreme programming*  bertujuan untuk meningkatkan kualitas perangkat lunak dan produktifitas, mengurangi biaya dari adanya perubahan kebutuhan dalam suatu siklus rancang bangun yang pendek dan berulang. Ada empat aktivitas dasar yang dilakukan dalam proses pembangunan perangkat lunak : *coding, testing, listening,* dan *designing*.

1. **Pengujian dan Evaluasi**

Pengujian dan evaluasi dilakukan terhadap 2 hal yaitu fungsionalitas dan validitas perangkat lunak. Fungsionalitas perangkat lunak duji dengan menggunakan unit testing *JUnit* yang mengimplementasikan *white box testing*. JUnit adalah *unit testing framework* yang digunakan untuk melakukan ujicoba pada perangkat lunak yang dibangun dengan bahasa pemrograman Java. Sedangkan validitas perangkat lunak diuji dengan beberapa metode (*recall, precision, f-measure*, dan lain-lain) dalam *Information Retrieval*(IR) untuk mengukur tingkat kelayakan dari model yang dihasilkan.

1. **Penyusunan Buku Tugas Akhir**

Pada tahap ini merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan aplikasi serta hasil dari implementasi perangkat lunak yang telah dibuat. Secara garis besar, buku tugas akhir nantinya terdiri atas beberapa bagian yaitu :

1. Pendahuluan
   1. Latar Belakang
   2. Permasalahan
   3. Batasan Tugas Akhir
   4. Tujuan
   5. Metodologi
   6. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Uji Coba dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka
7. **JADWAL PEMBUATAN TUGAS AKHIR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **Bulan** | | | | | | | | | | | | | | | | | | | |
| **Maret** | | | | **April** | | | | **Mei** | | | | **Juni** | | | | **Juli** | | | |
| 1 | Studi Literatur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Perencanaan Perangkat Lunak |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Perancangan Perangkat Lunak |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Implementasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Uji Coba dan Evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Penyusunan Buku Tugas Akhir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **DAFTAR PUSTAKA**

[1] Sommerville, Ian, 2007, Software Engineering (8th ed.), Pearson Education , Harlow, England.

[2] Zelkowitz, M. V., Shaw, A. C., dan Gannon, J. D., 1979, *Principles of Software Engineering and Design*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

[3] Marcus, A., Menzies, T., 2008, *Automated Severity Assessment of Software Defect Reports*, Wayne State University, West Virginia University, Detroit.

[4]Telles, Matt and Yuan Hsieh. *The Science of Debugging*. Scottsdale: Coriolis, 2001.

[5] Bugzilla Team, 2010, *The Bugzilla Guide - 3.6 Release*, Mozilla Foundation, Mountain View, CA.

[6]Witten, Ian H & Frank, Eibe.(2005).Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. USA: Elsevier Inc

[7] Frank, E., Hall, M., Kirkby, R., et all, 2010, *WEKA Manual for Version 3-7-3*, University of Waikato, Hamilton, New Zealand.

**LEMBAR PENGESAHAN**

###### **Surabaya, 14 April 2011**

Menyetujui, Menyetujui,

Dosen Pembimbing I Dosen Pembimbing II

# **Daniel Oranova Siahaan,S.Kom,M.Sc,PDEng Umi Laili Yuhana, S.Kom, M.Sc.**

# **NIP.** **19741123 200604 1001 NIP. 19790626 200501 2002**