**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama** : **Median Yuli Hartanto**

**NRP** : **5107 100 161**

Dosen Wali : Dr. H. Agus Zainal Arifin, S.Kom., M.Kom

1. **JUDUL TUGAS AKHIR**

***“Prediksi Umur Bug dalam Proyek Pengembangan Perangkat lunak”***

1. **URAIAN SINGKAT**

Salah satu bagian penting dalam proses manajemen proyek perangkat lunak adalah menajemen *bug*. Salah satu kakas populer untuk melakukan manajemen *bug*, adalah *bug*zilla. Manajemen *bug* ini erat kaitannya dengan penjadwalan kerja dalam tahap evolusi perangkat lunak. Setiap *bug* yang ditemukan akan diperkirakan berapa lama *bug* ini dapat diselesaikan dan ditentukan pula siapa yang akan memperbaikinya. Perkiraan penyelesaian *bug* ini dapat berbeda-beda antar anggota tim pengembang, tergantung sekali pada pengalaman anggota tim yang bersangkutan. Hal tersebut dapat menjadi masalah saat mengelola *bug*, terutama masalah jadwal kerja yang kemungkinan akan mundur, tidak sesuai dengan pengalokasian waktu awal.

Pada tugas akhir ini akan dibangun suatu perangkat lunak yang dapat memberikan rekomendasi umur dari *bug* secara otomatis. Sistem ini akan dibangun dengan menggunkan teknik-teknik penggalian data. Model prediksi yang dibangun didasarkan kepada data sejarah *bug* yang pernah terjadi sebelumnya pada proyek perangkat lunak yang sama. Diharapkan dengan adanya fitur ini, rekomendasi umur *bug* dapat lebih akurat.

1. **LATAR BELAKANG**

Proses pengembangan perangkat lunak secara umum dapat dibagi menjadi beberapa aktifitas, yaitu spesifikasi (*specification),* pengembangan (*development), validasi (validation),evolusi (evolution)*.[1] Biasanya proses pengembangan perangkat lunak dimulai dari tahap spesifikasi kebutuhan sistem, yaitu aktifitas untuk menspesifikasikan layanan yang harus disediakan oleh sistem serta batasannya, baik dalam lingkungan pengembangan maupun operasional. Tahap selanjutnya adalah aktifitas pengembangan*.* Pada aktifitas pengembangan, sistem perangkat lunak mulai dibangun. Dalam aktifitas ini, pertama kali pengembang harus mengembangkan desain perangkat lunak. Setelah itu pengembang mengimplementasikan langsung dari desain tersebut hingga menghasilkan suatu perangkat lunak, baik berupa aplikasi dan dokumentasinya. Setelah perangkat lunak terbentuk, pengembang melakukan aktifitas validasi*.* Validasi merupakan aktifitas yang dilakukan untuk memastikan perangkat lunak yang dibangun sesuai dengan tujuan awal pembangunan perangkat lunak. Dalam aktifitas ini perangkat lunak diperiksa apakah perangkat lunak yang dihasilkan sesuai dengan keinginan pelanggan yang memesannya. Aktifitas terakhir adalah evolusi*.* Aktifitas ini bertujuan untuk membuat perubahan pada perangkat lunak sehingga perangkat lunak tersebut sesuai dengan perubahan-perubahan yang terjadi di luar maupun di dalam sistem perangkat lunak tersebut. Dalam tahapan ini, perangkat lunak akan dirubah beberapa bagian untuk disesuaikan dengan perubahan yang terjadi sehingga perangkat lunak tersebut masih dapat digunakan dan difungsikan dengan baik.

Salah satu aktifitas terpanjang dalam suatu pengembangan perangkat lunak adalah evolusi. Menurut Sommerville, evolusi merupakan aktifitas yang menghabiskan paling banyak dana dan usaha. Dalam gambar 1 disebutkan bahwa pembangunan sistem perangkat lunak hanya memakan 25% dari total biaya yang dikeluarkan, 75%nya dikeluarkan untuk melakuakan evolusi sistemnya. Hal tersebut disebabkan aktifitas evolusi dilakukan setelah perangkat lunak dibuat hingga perangkat lunak sudah tidak lagi dipakai. Disana pihak pengembang dituntut untuk dapat menyelesaikan setiap kesalahan yang ada dalam perangkat lunak. Bagian-bagian yang harus diubah dapat diusulkan oleh tim pengembang maupun pemakai yang nantinya akan disetujui oleh pimpinan proyek.

1.2 Development cost di.eps                                    000231B6Macintosh HD                   B8AA5F2E:

Gambar 1 Perbandingan dana antara pembangunan sistem dengan evolusi sistem [1]

Dalam SLDC (*Software Development Life Circle*), aktifitas evolusi dilakukan dalam tahapan pemeliharaan (*maintenance)*. Dalam aktifitas ini, pengembang dihadapkan pada banyak sekali masalah, terutama masalah untuk mengatur sumber daya. Mulai dari jumlah tim pengembang yang terbatas, dana yang terbatas, dan waktu yang diberikan untuk menyelesaikan suatu masalah terbatas. Maka dari itu pengembang meperlukan suatu tindakan yang efektif dan efisien yang dapat menyelesaikan permasalah tersebut, terutama untuk masalah *bug*. Untuk dapat melakukan tindakan yang efektif dan efisien diperluakan mekanisme manajemen, baik menajemen terhadap sumber daya maupun manajemen terhadap masalah yang ada. Karena masalah yang muncul pada tahap pemeliharaan adalah *bug*, maka diperlukan suatu sistem pelacak *bug*.

Salah satu contoh sistem pelacakan *bug* yang populer adalah bugzilla[8]. Sistem ini akan mencatat atribut *bug* yang ditemukan. Setelah itu juga ditentukan pula berapa lama waktu yang akan diberikan kepada salah seorang anggota tim pengembang untuk menyelesaikannya. Untuk menentukan waktu penyelesaian tersebut tim pengembang masih mengandalkan pengalaman manusia tanpa ada rekomendasi yang diberikan oleh sistem. Padahal, salah satu masalah yang paling krusial dalam sistem pelacakan *bug* adalah masalah penentuan waktu yang diberikan kepada tim pengembang untuk menyelesaikan suatu *bug*. Penentuan waktu tersebut sangat besar kaitannya dengan efisiensi kerja. Penentuan waktu ini akan sangat berpengaruh terhadap penentuan jadwal kerja bagi setiap anggota tim pengembang. Bila pengembang melakuakan kesalahan dalam penentuan waktu ini, maka jadwal kerja yang dibuat untuk masing-masing anggota tim pengembang akan berubah dan akan sangat berdampak terhapa jalannya proses pengembangan perangkat lunak. Oleh karena itu, sangat dibutuhkan penentuan waktu penyelesaian suatu *bug* yang tepat. Biasanya waktu yang dipelukan untuk menyelesaikan suatu *bug* disebut dengan umur *bug*.

Panjer pernah melakukan penelitian mengenai bagaimana memprediksi umur *bug* untuk dataset perangkat lunak eclipse dengan menggunakan pendekatan beberapa metode penggalian data. Dalam penelitian tersebut, hasil terbaik yang ditemukan adalah 34,9% untuk menemukan prediksi benar.[3] Dalam penelitian tersebut, Panjer masih menggunakan aplikasi yant terpisah-pisah untuk mencari prediksi umur *bug*. Panjer menggunakan perkakas Weka untuk membantu menganalisa data, namun Panjer belum membuat aplikasi tersebut terintegrasi.

Untuk itulah diperluakan suatu aplikasi untuk menghitung berapa waktu yang diperluakan untuk menyelesaikan suatu *bug* dalam suatu perangkat lunak.

1. **RUMUSAN MASALAH**

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

1. Bagaimana membaca data repositori dari Bugzilla kemudian merubahnya kedalam format ARFF.
2. Bagaimana menerapkan beberapa teknik penggalian data untuk memodelkan data umur *bug* dengan menggunakan Weka.
3. Bagaimana mengimplementasikan model yang ditemukan untuk mengembangkan aplikasi yang dapat memprediksi umur suatu *bug*.
4. **BATASAN MASALAH**

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

1. Tugas akhir ini akan menggunakan Weka sebagai perkakas untuk teknik penggalian data.
2. Dataset yang digunakan untuk memodelkan umur *bug* digunakan data pengembangan perangkat lunak Eclipse.
3. Teknologi yang digunakan adalah platform Java.
4. **TUJUAN TUGAS AKHIR**

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

1. Membaca data repositori dari Bugzilla kemudian merubahnya kedalam format ARFF.
2. Menerapkan beberapa teknik penggalian data untuk memodelkan data umur *bug* dengan menggunakan Weka.
3. Mengimplementasikan model yang ditemukan untuk mengembangkan aplikasi yang dapat memprediksi umur suatu *bug*.
4. **MANFAAT TUGAS AKHIR**

Manfaat yang diharapkan dari tugas akhir ini adalah membuat suatu perangkat lunak yang mampu memberikan rekomendasi berapa lama waktu yang dibutuhkan untuk menyelesaikan suatu *bug*.

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 programer, penjamin kualitas, maupun anggota tim yang lain. Ada juga sistem pelacak *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* yang 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 jamak digunakan dalam dunia pengembangan perangkat lunak, bahkan sistem ini diintegrasikan dengan sistem manajemen proyek.

Di pasaran juga telah tersedia banyak perangkat lunak sistem pelacakan *bug*, mulai dari yang *open source* hingga yang berbayar. Contohnya GNAT (1992), [Debbugs](http://en.wikipedia.org/wiki/Debian_bug_tracking_system) (1994), [Bugzilla](http://en.wikipedia.org/wiki/Bugzilla) (1998), [MantisBT](http://en.wikipedia.org/wiki/Mantis_Bug_Tracker) (2000), [Roundup](http://en.wikipedia.org/wiki/Roundup_(issue_tracker)) (2001), [BugTracker.NET](http://en.wikipedia.org/wiki/BugTracker.NET) (2002), [Flyspray](http://en.wikipedia.org/wiki/Flyspray_(bug_tracking_software)) (2003) , [Launchpad](http://en.wikipedia.org/wiki/Launchpad_(website)) (2004), [Trac](http://en.wikipedia.org/wiki/Trac) (2006), [Redmine](http://en.wikipedia.org/wiki/Redmine) (2006) untuk yang open source. Selain itu juga ada sistem yang berbayar yaitu [FogBugz](http://en.wikipedia.org/wiki/FogBugz) (2000), [JIRA](http://en.wikipedia.org/wiki/JIRA) (2004), [YouTrack](http://en.wikipedia.org/wiki/YouTrack) (2009).

**Bugzilla**

Bugzilla merupakan salah satu sistem pelacakan *bug* yang populer, mulai dikembangkan pada tahun1998 dan menjadi salah satu sistem gratis yang populer untuk digunakan mengembangkan perangkat lunak. Sampai dengan saat ini, Bugzilla tercapat digunakan oleh 1162 proyek.

Bugzilla memiliki fungsi utama untuk melacak *bug* dan perubahan kode suatu perangkat lunak. Selain itu bugzilla juga dapat digunakan sebagai sarana komunikasi antar anggota tim pengembang. Bugzilla juga menyediakan fitur untuk mengelola jaminan kualitas (QA).

Gambar 2 menjelaskan bagaimana suatu *bug* hidup dalam Bugzilla. *Bug* memiliki status-status yang menunjukkan bagaiman kondisi *bug* tersebut dalam proyek perangkat lunak. Setiap user memiliki hak masing-masing dalam menentukan status tiap bug. Seperti untuk menentukan apakah suatu *bug* sudah terverifikasi merupakan hal dari QA (Quality Assurence).



Gambar 2 alur hidup *bug* pada Bugzilla [6]

Dalam Bugzilla, *bug* memiliki atribut-atribut yang berfungsi untuk mencatat spesifikasi dari *bug* tersebut. Atribut-atribut tersebut dapat digunakan untuk memihat catatan-catatan mengenai bug. Di dalam Bugzilla, but memiliki 20 atribut yang semestinya diisi oleh pelapor *bug*. Atribut-atribut tersebut antara lain, Product and Component, Status and Resolution, Assigned To, QA Contact, URL, Summary, Status Whiteboard, Keywords, Platform and OS, Version, Priority, Severity, Target, Reporter, CC list, Time Tracking, Attachments, Dependencies, Votes, Additional Comments.

**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 [2]. 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.

**Weka**

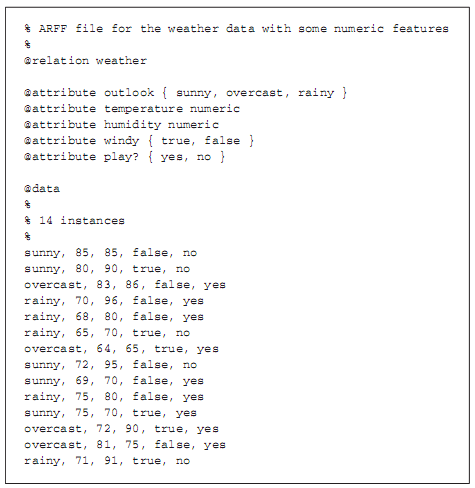
Salah satu unsur terpenting dalam pembuatan tugas akhir ini adalah penggalian data. Penggalian data tersebut digunakan penulis untuk menemukan model dari umur *bug* dalam dataset yang digunakan. Untuk membuat implementasi dari penggalian data penulis membutuhkan banyak sekali sumber daya, mulai dari waktu, teknik, tenaga. Untuk itu dalam pembuatan tugas akhir ini akan menggunakan pustaka untuk melakukan penggalian data. Pustaka yang akan digunakan adalah Weka[7]. Weka merupakan kepanjangan dari *Waikato Environment for Knowledge Analysis.*

Weka merupakan kumpulan dari banyak algoritma *learning machine* dan alat preprosesing data yang digunakan untuk melakukan penggalian data yang dikembangkan oleh universitas Waikato. Weka dikembangkan menggunakan bahasa Java dan menggunakan  [*GNU General Public License*](http://www.gnu.org/copyleft/gpl.html)untuk lisensinya*.* Weka diklam dapat melakukan *data pre-processing, classification, regression, clustering, association rules,* dan *visualization*. Suatu 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*.[5]

Weka hanya dapat menjalankan suatu algoritma dengan data masukkan yang berformat tabel relasional tunggal yang berformat ARFF.

**ARFF**

  ARFF (Attribute-Relation File Format) adalah berkas teks berformat ASCII yang berisikan daftar satu set data atribut. Berkas ARFF dikembangkan oleh Machine Learning Project di Departemen Ilmu Komputer Universitas Waikato untuk digunakan dengan [sebagai](http://translate.googleusercontent.com/translate_c?hl=id&ie=UTF-8&sl=en&tl=id&u=http://www.cs.waikato.ac.nz/~ml/&prev=_t&rurl=translate.google.com&twu=1&usg=ALkJrhg43DDiExq3gw98yF6QlY-9N9Hzwg) dataset Weka. Gambar 3 menunjukkan contoh dari isi berkas ARFF.



Gambar 3 Contoh Isi Berkas ARFF [2]

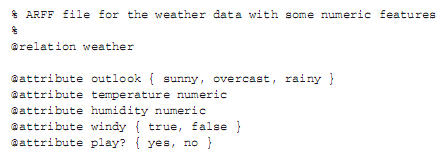
Sebuah berkas ARFF terdiri atas 2 bagian yaitu Header dan Data. Header terdiri atas 3 informasi kunci yaitu nama dari relasi, daftar atribut dan tipe datanya. Gambar 4 menunjukkan salah satu contoh header pada berkas ARFF. Simbol “%” digunakan untuk menuliskan komentar. Kemudian, @relation <relation-name> digunakan untuk mendeklarasikan nama dari relasi dalam berkas tersebut, dimana <relation-name> berupa string. Format untuk @attribute adalah @attribute <attribute-name> <datatype>, dimana <attribute-name> adalah suatu string nama untuk atribut tersebut dan <datatype> merupakan salah satu dari tipe data yang didukung oleh Weka, yaitu numeric, nominal-specification, string, date . @attribute digunakan untuk mendeklarasikan nama atribut dan tipe datanya.

Untuk tipe data yang didukung oleh weka terdiri atas 4 macam yaitu

1. Numeric digunakan untuk angka real dan integer.
2. Nominal-Specification digunakan untuk mebuat suatu urutan nilai yang disusun sendiri oleh pengguna. Untuk menuliskannya dalam header, Nominal-specification ini ditulis dengan format {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}. Contohnya :

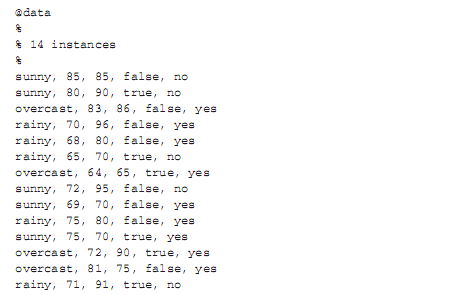
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

1. String digunakan untuk data yang berupa tulisan string.
2. Date digunakan untuk data tanggal. Untuk menuliskannya, date ini menggunakan format @attribute <name> date [<date-format>], dimana [<date-format>] diisi dengan kombinasi dari "yyyy-MM-dd'T'HH:mm:ss”.



Gambar 4 bagian *header* pada berkas ARFF [2]

Pada bagian @data, bagian ini berisikan data-data dari berkas ARFF tersebut. Penulisan data pada ARFF dimulai dengan menuliskan @data kemudian diikuti data-data atribut yang dicatat. Penulisan data menggunakan urutan deklarasi atribut pada bagian header dengan dipisahkan tanda “, “. Setiap baris merupakan representasi satu data. Gambar 5 merupakan contoh dari bagian data pada berkas ARFF.

****

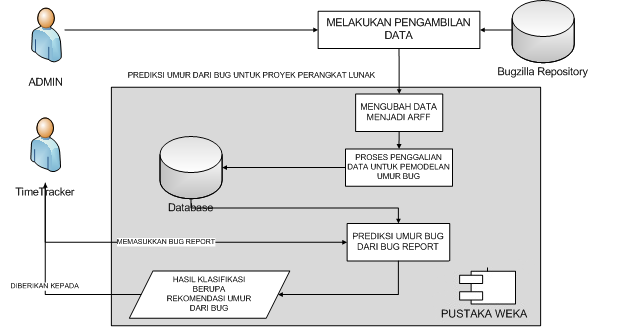
Gambar 5 bagian data pada berkas ARFF [2]

1. **RINGKASAN TUGAS AKHIR**

Pada tugas akhir ini, penulis akan membuat sebuah aplikasi yang dapat digunakan untuk memperkirakan umur dari suatu *bug*, yaitu rentan waktu antara *bug* ditemukan hingga *bug* tersebut selesai diperbaiki sehingga perangkat lunak menjadi lebih baik. Kebanyakan penentuan umur *bug* ditentukan langsung oleh anggota developer yang masuk ke dalam *timetrackinggroup*. Selain itu, dia juga bertanggung jawab untuk menentukan siapa yang akan mengerjakannya. Biasanya penentuan waktu berapa lama *bug* harus dikerjakan hanya mengandalkan pengalaman. Bila yang umur *bug* tersebut harus ditentukan oleh orang yang belum memiliki pengalaman, maka akan terjadi deviasi hasil keputusan dengan orang yang sudah berpengalaman. Hal tersebut akan menjadi masalah dalam hal penjadwalan proyek, maka dari itu diperlukan suatu aplikasi untuk menentukan hal tersebut. Aplikasi tersebut nantinya harus dapat membantu orang-orang yang belum memiliki pengalaman sehingga keputusannya tidak terlalu berbeda dengan orang yang sudah memiliki pengalaman.

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 ingin diolah harus diatur dengan format ARFF sesuai dengan permintaan dari Weka. Menyesuaikan dengan Weka yang dibangun di atas platform Java, aplikasi ini akan dibangun pula di atas platform Java.

Setelah data didapatkan, aplikasi menjalankan beberapa teknik penggalian data untuk menyoba memodelkan prediksi umur *bug*. Teknik-teknik yang digunakan antara lain, 0-R, 1-R, C4.5 Decision Tree, Naive Bayes, Logistic Regression. Dengan menggunakan teknik-teknik tersebut, aplikasi akan menghasilkan model mengenai prediksi umur *bug*. Model tersebut akan disimpan. Dari model yang disimpan tersebut akan didapatkan keluaran prediksi umur *bug* berupa jumlah hari. Agar didapatkan keluaran seperti itu, model prediksi umur *bug* tersebut harus diberikan masukan berupa atribut-atribut *bug* yang dibutuhukan.



Gambar 6 Skema Proses Sistem Prediksi Umur *Bug*

Sebagai gambaran, aplikasi akan dibangun mengguakan skema seperti yang terlihat pada gambar 6. Sistem akan memiliki dua aktor memiliki peran masing-masing. Admin, aktor yang bertugas untuk menentukan apakah data dan model perlu diperbarui atau tidak. TimeTracker, aktor yang menerima rekomendasi umur *bug* dari *bug* yang telah dia masukkan.

Aplikasi ini berjalan dengan urutan seperti bawah ini;

1. Melakukan pengambilan data. Data tersebut diambil dari repositori bugzilla dengan menggunakan BugzillaAPI. BugzillaAPI melakukan pertukaran data menggunakan konsep webservice XMLRPC. Data yang diambil terdiri atas beberapa atribut yang dibutuhkan, seperti tanggal terakhir diubah, tanggal mulai dicatat, komponen, os, severity, dan lain sebagainya. Data tersebut diambil dari bugs.eclipse.org/bugs. Skema pengambilan data dari repositori Bugzilla digambarkan dalam Gambar 7.

Gambar 7 Skema Pengambilan data bug dari Repositori Bugzilla

1. Mengubah Data Menjadi ARFF. Data yang diambil dari repositori bugzilla diubah sesuai dengan kebutuhan file ARFF dan disimpan dengan format ARFF. Hal ini dilakukan karena Weka membutuhkan file berformat ARFF untuk dapat melakukan proses penggalian data. Dari file yang berformat ARFF tersebut, aplikasi akan melakukan permodelan. Skema penyimpanan data ke dalam file ARFF digambarkan dalam Gambar 8.

Gambar 8 Skema penyimpanan data dalam file ARFF

1. Proses Penggalian data untuk Permodelan Umur Bug. Data ARFF tersebut diolah menggunakan beberapa teknik penggalian data untuk mendapatkan model umur bug dari data histori bug yang ada, untuk kasus tugas akhir ini menggunakan data bug dari perangkat lunak eclipse. Teknik-teknik yang digunakan adalah 0-R, 1-R, C4.5 Decision Tree, Naive Bayes, Logistic Regression. Hasil dari proses penggalian data tersebut akan disimpan ke dalam database. Skema penggalian data untuk memodelkan umur bug digambarkan dalam Gambar 9.

Gambar 9 Skema Penggalian data untuk mendapatkan model umur bug

1. Prediksi Umur Bug dari Bug Report. Setelah didapatkan model umur bug, pengguna yang memiliki hak sebagai timetracker dapat memasukkan laporan bug. Dari data bug yang dimasukkan, sistem akan membuat prediksi berapa lama bug tersebut dapat diperbaiki. Inilah keluaran terakhir dari aplikasi ini. Skema pemberian prediksi umur bug dari sebuah bug report digambarkan dalam Gambar 10.

Gambar 10 Pemberian prediksi umur bug dari sebuah bug report

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

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan Proposal Tugas Akhir. Pada proposal ini, penulis mengajukan gagasan pembuatan aplikasi untuk memprediksi umur suatu *bug* dalam proyek pengembangan perangkat lunak.

1. Studi Literatur

Pada tahapan ini akan dilakukan studi literatur mengenai metode yang digunakan, diantaranya :

* + - * 1. Beberapa teknik penggalian data.
        2. Penggunaan perkakas Weka untuk menggali dataset.
        3. Bugzilla.

1. Analisa dan Perancangan

Dalam tahap ini akan dilakukan analisa bagaiman seharusnya perangkat lunak ini dibuat. Nantinya akan dibuat juga rancangan arsitektur perangkat lunak ini. Analisa dan perancangannya akan menggunakan metode berorientasi obyek (*Object Oriented).* Dimulai dengan merancang *usecase model,* kemudian *sequence diagram,* dan terakhir *class diagram.*

1. Implementasi

Implementasi merupakan tahap membangun aplikasi, yaitu mengimplementasikan rancangan yang dibuat ke dalam baris kode program. Dalam pembangunan perangkat lunak ini, penulis akan menggunakan bahasa pemrograman Java. Pustaka weka akan digunakan untuk menangani masalah metode penggalian data. Untuk metode pengembangan perangkat lunak, akan digunakan metode *Extreme Programming,*  yaitu melakukan siklus-siklus kecil dalam mengembangkan perangkat lunak. Siklus pengembangan yang kecil tersebut akan memiliki waktu yang pendek untuk tiap siklusnya namun akan dibagi ke dalam siklus yang cukup banyak dan terus berulang. Sebagai gambaran dalam metode ini ada 4 aktifitas dasar yang dilakuakan, yaitu *coding* (menulis baris program), *testing* (mengetes baris kode program), *listening* (mendengarkan komentar dari *stakeholder* perangkat lunak) dan *designing* (dari hasil listening akan coba dibuat desain yang baru).

1. Pengujian dan Evaluasi

Dalam tahap ini akan dibagi menjadi 2 kegiatan utama. Yang pertama adalah uji fungsionalitas perangkat lunak. Unji fungsionalitas ini bertujuan untuk memastikan seluruh fungsi-fungsi dari perangkat lunak berjalan dengan baik. Untuk uji fungsionalitas ini akan digunakan *JUnit. JUnit* adalah *unit testing framework* untuk bahasa Java. *Unit testing* adalah perangkat lunak yang digunakan untuk mengecek apakah suatu perangkat lunak berjalan sesuai dengan harapan pembuat atau tidak.

Tahap kedua adalah uji Uji validitas perangkat lunak. Dalam uji ini, perangkat lunak dicoba apakah hasil yang dikeluarkan valid atau tidak. Untuk tugas akhir ini, nantinya akan menggunakan metode *Kappa Statistic* untuk mengukur tingkat kelayakan dari model yang dihasilkan.

1. Penyusunan Buku Tugas Akhir

Tahap terakhir merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perancangan dan prototyping yang telah dibuat. Secara garis besar, buku laporan tugas akhir ini 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 KEGIATAN TUGAS AKHIR**

Tugas akhir ini diharapkan bisa dikerjakan menurut jadwal sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **Bulan** | | | | | | | | | | | | | | | |
| **Maret 2011** | | | | **April 2011** | | | | **Mei 2011** | | | | **Juni 2011** | | | |
| 1. | Penyusunan Proposal Tugas Akhir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. | Studi Literatur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. | Analisa dan perancangan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. | Implementasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. | Pengujian dan Evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6. | Penyusunan Buku Tugas Akhir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **DAFTAR PUSTAKA**

[1] Sommerville, I.(2004). Software, 8th ed. Harlow, England:Pearson Education Ltd.

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

[3] Panjer, L.D. (2007). Predicting Eclipse *Bug* Lifetime. IEEE.

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

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

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

[7] *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*, (<http://www.cs.waikato.ac.nz/ml/weka/index.html>, diakses tanggal 06 April 2011)

[8] *Bugzilla*, (<http://www.bugzilla.org/>, diakses tanggal 06 April 2011)

**LEMBAR PENGESAHAN**

###### **Surabaya, 12 Mei 2011**

Menyetujui,

Dosen Pembimbing II

Sarwosri, S.Kom, M.Kom

NIP : 19760809 200112 2001

Menyetujui,

Dosen Pembimbing I

# Daniel Oranova Siahaan, S.Kom, M.Sc, PDEng

# NIP. 19741123 200604 1001