Identifikasi Atribut Tingkat Lebih Tinggi untuk Prediksi Umur *Bug*

Identify Higher Level Attributes for Bug Age Prediction

Suluh Sri Wahyuningsih[1]*, Nursalim[2]
[1,2]Fakultas Teknik, Universitas Muhammadiyah Palu, Indonesia

Abstrak: Sebuah perangkat lunak yang berkualitas dapat diartikan sebagai suatu produk yang memiliki jumlah kesalahan atau *bug* yang sedikit. Berbagai cara dilakukan untuk menggurangi jumlah *bug*, seperti sistem pelacak bugzilla informasi yang disimpan dapat digunakan untuk menyelidiki fenomena yang berbeda. Manajemen proyek perlu memperkirakan waktu yang dibutuhkan untuk menangani suatu *bug* agar dapat membuat perencanaan proyek yang baik. Penelitian sebelumnya menggunakan atribut primitif (*Low Level Attribute*) untuk prediksi umur *bug*, merekomendasikan penggunaan atribut *bug* tingkat yang lebih tinggi. Oleh karena itu, atribut tingkat lebih tinggi diprediksi keberhasilan dengan akurasi umur *bug*. Dalam penelitian ini, identifikasi atribut tingkat lebih tinggi digunakan untuk meningkatkan akurasi prediksi umur sebuah *bug*. Untuk mengidentifikasi atribut mana yang signifikan pengaruhnya terhadap prediksi umur *bug* digunakan pencarian nilai informasi (*infogain)*. Langkah kedua, yaitu mengukur akurasi klasifikasi berdasarkan atribut-atribut yang ditemukan, oleh karena itu menggunakan sejumlah metode, yaitu *Zerro_R, One_R* , *Decision Tree*, dan *Naive Bayes*. Metode-metode ini baik digunakan untuk dataset yang memiliki korelasi, melibatkan 24 buah atribut, 7 kelas *bug_lifetime* dan data set sebesar 1000 *bug*. Hasil penelitian mengidentifikasi 6 atribut tingkat tinggi, dimana 2 diantaranya (*summary*, dan *last change time*) dianggap memiliki pengaruh yang signifikan dalam memprediksi umur *bug*. Kombinasi atribut tingkat tinggi (2 atribut), tingkat rendah (3 atribut) dan seleksi (1 atribut) menghasilkan indeks kappa tingkat *substantial* (0,81). Hal tersebut menunjukkan dengan penambahan atribut tingkat lebih tinggi untuk prediksi umur *bug* dapat bekerja lebih baik dari penelitian sebelumnya yang menghasilkan indek kappa *moderate* (0,60).

***Abstract**: A high quality software is represented by a product which has as minimum as possible number of error or bug. In project management, it is necessary to precisely estimate the time needed to solve a bug in order to ensure the success of the project. Previous research related to bug lifetime prediction recommends the use of higher level attributes of bug to increase the prediction accuracy. It suggest that bug's lifetime is highly dependent on higher level bug attributes. This research identifies those higher level bug attributes. To measure its significant factor in determining bug's lifetime, we used infogain. Then four classification mehods is use to give insight on the accuracy gained by introducing those high level bug attributes in determining bug's lifetime. We used Zerro_R, One_R, Decision Tree, and Naive Bayes. This method is good for classifying huge dataset. This research involved 24 (low and high level) bug attributes as well as dataset as much 1000 bugs. Our research identified 6 new high level bug attribute, which 2 of them (summary, and last change time) have significant factor in determining bug's lifetime. Our testing shows that combination of (2) high level, (3) low level and attribute (1) bug attributes for predicting bug's lifetime produce kappa value of 0,81. This improves the result from the previous research which produce kappa value of 0,60.*

Kata Kunci:
Prediksi Umur *Bug;*
Atribut Tingkat Lebih Tinggi;
Atribut Tingkat Lebih Rendah

Keywords:
Bug Age Prediction;
Higher Level Attributes;
Lower Level Attributes

INDEXED IN

Crossref
Google Scholar
Garba Rujukan Digital: Garuda

CORRESPONDING AUTHOR

Suluh Sri Wahyuningsih

Fakultas Teknik, Universitas Muhammadiyah Palu, Indonesia

EMAIL

suluhsw@gmail.com

OPEN ACCESS
e ISSN 2623-2022

Jurnal Kolaboratif Sains (JKS)
Volume 6 Issue 3 Maret 2023
Pages: 164-180

LATAR BELAKANG

Proses perangkat lunak adalah suatu rangkaian dari aspek kegiatan untuk menghasilkan sebuah perangkat lunak (Tan, Steinbach, and Kumar 2006). Sebuah perangkat lunak perlu dijaga kualitasnya. Pertama, produk perangkat lunak diharapkan mempunyai fungsi dan waktu yang sama ketika pemakai memerlukannya. Kedua, produk harus bisa dijalankan. Jika suatu produk terdapat kesalahan, maka produk itu tidak memiliki kelayakan (Humphrey 2005). Oleh karena itu, perangkat lunak yang berkualitas dapat diartikan suatu produk yang memiliki jumlah kesalahan atau *bug* yang sedikit.

Kesalahan atau *bug* dapat menyulitkan pemakai, karena *bug* dapat menghambat suatu pekerjaan. Sebuah *bug* perlu untuk dihilangkan, tetapi *bug* tidak dapat dihindari dalam perangkat lunak. Berbagai cara dilakukan untuk mengurangi jumlahnya seperti, manajemen proyek perlu memperkirakan waktu yang dibutuhkan untuk menangani suatu *bug* agar dapat membuat perencanaan proyek yang baik, sehingga kualitas perangkat lunak diterima penguna menjadi lebih baik. Salah satu contoh sistem pelacakan *bug* yang dapat digunakan untuk menangani suatu *bug* adalah *Eclipse bugzilla*, Sistem ini akan mencatat atribut *bug* yang ditemukan.

Atribut *bug* mempunyai pengaruh terhadap prediksi umur hidup *bug.* Umur hidup *bug* untuk kelas linux dapat memanjang sampai bertahun-tahun (Chou et al. 2001). Proyek perangkat lunak *open-source* memiliki umur *bug* rata-rata 200 hari dan bisa memanjang sampai bertahun-tahun (Kim and Whitehead Jr 2006). Sistem pelacak Bugzilla untuk proyek Eclipse ini memiliki 24 atribut *bug*, yaitu *Bug_id, Assigned_to, Qa_contact, Severity, Priority, Component, Product, OS, Platform, Version, Target_Milestone, CC_count, Bug_dependensi, Comment, Resolution, Creator, Creation_time, Last_change_time, status, number_of_word_in_comment,number_of_sentence_in_commet,average_word_in_commentdan bug_lifetime*. Dua puluh empat Atribut *bug* untuk selanjutnya disebut sebagai atribut primitif. Panjer menyatakan bahwa Atribut primitif berpengaruh terhadap proses penentuan umur hidup *bug*, yaitu atribut *Assigned_to, Qa_contact, Severity, Priority, Component, Product, OS, Platform, Version, Target_Milestone,CC_count,Bug_dependensi, dan bug_lifetime* (Hussain 2007).

Atribut primitif akan digunakan untuk menentukan atribut tingkat yang lebih tinggi. Sebab, atribut tingkat yang lebih tinggi diprediksi memiliki kekuatan besar dalam penentuan umur hidup *bug* (Hussain 2007). Salah satu masalah dalam sistem pelacakan *bug* adalah penentuan waktu yang diberikan kepada tim pengembang dalam menyelesaikan suatu *bug.* Waktu yang diperlukan untuk menyelesaikan suatu *bug* disebut umur *bug.* Oleh karena itu, atribut tingkat lebih tinggi dari hasil identifikasi akan digunakan untuk prediksi umur *bug*.

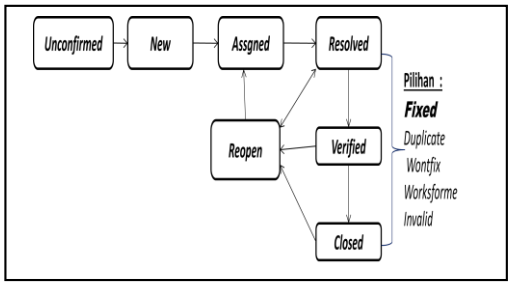
Dilatar belakangi oleh rekomendasi (Hussain 2007) tersebut, penggunaan atribut *bug* tingkat yang lebih tinggi seperti *Severity, Priority* dan *comment* diduga dapat meningkatkan akurasi prediksi umur *bug*. Penelitian ini akan mencoba mengidentifikasi atribut tingkat lebih tinggi yang signifikan pengaruhnya terhadap prediksi umur hidup *bug.* Metode yang digunakan untuk prediksi umur *bug* menggunakan kakas bantu WEKA. Kemudian dilakukan proses infogain, dan empat klasifikasi untuk melihat data set yang terbentuk. Kakas bantu WEKA menyediakan teknik klasifikasi dan alat preprosesing yang dapat digunakan untuk penggalian data. Penggalian data ini menggunakan *Zerro-R, One-R, Decision Trees*, dan *Naive Bayes*. Penggunaan infogain disebabkan atribut *comment* bertipe *string*, sehingga sebelum proses klasifikasi perlu dilakukan pembobotan *term*. Proses Pembobotan *term* dilakukan dengan perhitungan tf*idf untuk mengetahui integrasi antar *term frequency (tf)* dan *inverse*

document frequency (idf), kemudian dihitung menggunakan infogain untuk mendapatkan term yang baik untuk proses klasifikasi.

Penggunaan teknik klasifikasi *Zerro-R*, *One-R*, *Decision Trees*, dan *Naive Bayes*, karena metode ini tersedia pada mesin pembelajaran yang terdapat pada Weka, selain itu karena empat metode tersebut baik digunakan untuk dataset yang berkorelasi, serta untuk membandingkan dengan yang dilakukan oleh (Hussain 2007).

TINJAUAN LITERATUR

Siklus Sistem. Sistem pelacak *Eclipse Bugzilla* mengikuti serangkaian transisi status yang diperiksa setiap titik-titiknya, yaitu: *Unconfirmed. New, Assigned, Resolved, Verified, Closed* dan *Reopen*. Status *Resolved, Verified*, dan *Closed* memiliki lima pilihan resolusi, yaitu: *Fixed, Duplicate, Wonfix, Workforce* dan *Invalid. S*iklus ini digunakan untuk mengelola dan memfasilitasi proses perbaikkan *bug*. Siklus sistem pelacak *Eclipse Bugzilla* ditunjukkan seperti gambar 1.



Gambar 1. Siklus sistem bugzilla

Sistem pelacak *Eclipse Bugzilla* merekam berbagai karakteristik tentang keadaan *bug* yang dilaporkan. Perubahan status digunakan pelapor untuk melihat tahap penyelesaian *bug* (Hussain 2007). Informasi diperiksa setiap titik-titik tertentu sepanjang siklus sistem *Eclipse Bugzilla*. Informasi yang disimpan dalam sistem pelacak *bug* ini digunakan oleh para peneliti untuk menyelidiki fenomena yang berbeda, seperti penelitian ini akan mengidentifikasi atribut tingkat yang lebih tinggi untuk prediksi umur hidup *bug*.

Koleksi data *bug* dikelompokkan dalam atribut dan obyek. Atribut akan diklasifikasikan sesuai dengan factor tingkat. Sedangkan obyek yang diprediksi dari klas model klasifikasi berdasarkan nilai atribut atau fitur.

Atribut. Sebuah *bug* adalah suatu karakteristik yang dapat menggurangi kegunaan dari suatu *item*. *Item* adalah sebuah obyek dari kelompok. *Item* digunakan untuk menerangkan informasi yang direkam. *Item* ini disebut atribut, misalnya atribut *Severity, priority, Comment* dan sebagainya. Atribut diartikan suatu karakteristik dari obyek dunia nyata yang tidak tergantung dengan obyek lain.

Atribut primitif *bug* adalah atribut dasar yang secara eksplisit tersimpan dalam database sistem pelacak *bug*. Atribut primitif ini akan digunakan untuk mencari atribut tingkat lebih tinggi. Atribut tingkat lebih tinggi diprediksikan oleh (Hussain 2007) memiliki kekuatan besar dalam penentuan umur hidup *bug.* Atribut tingkat yang lebih tinggi adalah atribut yang nilainya di kalkulasi dari nilai atribut primitif atau atribut yang nilainya diperoleh dari penggabungan satu atau lebih atribut primitif.

Tabel 1. Atribut primitif *bug*

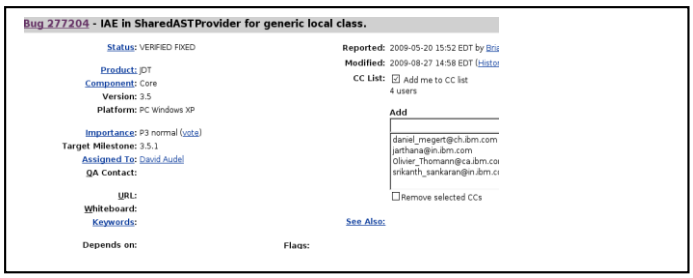
No	Atribut primitif	Definisi
1.	*Bug_id*	Id unik yang digunakan untuk menyimpan data nomor id laporan *bug*.
2.	*Assigned_to*	Pengembang yang bertugas untuk memperbaiki *bug* dengan meninggalkan *log* berupa *email*.
3.	*Qa_contact*	*Quality ansurance* yang bertugas untuk merubah status *bug* dengan meninggalkan log berupa email.

4.	*Priority*	Tingkat kepentingan *bug* yang harus diperbaiki.
5	*severity*	Tingkat keparahan dari sebuah *bug*
6.	*Component*	Lokasi terjadinya permasalahan atau bagian dari produk yang memiliki *bug*
7.	*Product*	Wilayah tertentu yang memiliki *bug*
8.	*OS*	Sistem operasi yang dilaporkan permasalahannya
9.	*Platform*	lingkup sistem operasi yang terindikasi *bug*
10.	*Version*	Nomor versi paket
11.	*Target_Milestone*	Nomor rilis versi paket
12.	*CC_Count*	Daftar pengguna yang turut dikirimi email jika terjadi perubahan terhadap suatu bug
13.	*Bug_dependent*	Jumlah bug yang dependent dengan bug pelapor lain
14.	*Keyword*	Jumlah kata kunci yang dimiliki oleh bug
15.	*Resolution*	Tahapan penyelesaian bug berdasarkan aturan yang ada pada siklus hidup *bug*
16.	*Creator*	Pelapor bug yang berupa log email
17.	*Creation_time*	Menyimpan tanggal lapor
18.	*Last_change_time*	Menyimpan tanggal terakhir
19	*comment*	Komentar penyelesaian Bug
20.	*Status*	Kondisi penyelesaian *bug* berdasarkan siklus hidup pada sistem pelacak *Bugzilla*
21.	*Number_of-word_in_comment*	Jumlah kata dari comment
22.	*Number_of_sentence_in_comment*	Panjang kalimat dari comment
23.	*Average_word_in_comment*	Rata-rata kata per kalimat
24.	*Bug_lifetime*	Menyimpan umur bug

Filtering, data available at http://www.briandunning.com/cf/936

Tabel 1 ini merupakan detail dari atribut primitif. Misalkan untuk atribut *Bug_lifetime* nilai diperoleh dari proses atribut primitif *creation_time* yang resolusi menujukkan fik *(veried fixed*) dikurangi oleh atribut *last_change_time*. Perubahan atribut *bug* pada rekayasa perangkat lunak dapat mengandalkan perubahan *log* yang ditinggalkan oleh pengembang (Murphy and Cubranic 2004). Jika perubahan *log* dapat memberikan petunjuk, maka revisi dianggap sebagai hasil pengolahan.

Atribut primitif akan digunakan untuk mengidentifikasi atribut tingkat lebih tinggi. identifikasi atribut tingkat lebih tinggi ini digunakan untuk prediksi umur *bug*. Umur hidup *bug* adalah waktu yang diperlukan untuk menyelesaikan suatu *bug*, sehingga kualitas dari proyek perangkat lunak dapat meningkatkan produktifitas. Jika melalui cara pemberitahuan awal tentang *bug*, mulai dari kapan dibuat sampai pergantian status, maka akan meningkatkan kesadaran dan responsibilitas anggota tim http://www.briandunning.com/cf/936.



Gambar 1. Laporan Bug (www.eclipsebugzilla.org)

Waktu digunakan untuk mengetahui seberapa lama suatu proses *bug* mulai dilaporkan sampai perbaikan, seperti terlihat pada gambar 1 adalah contoh laporan *bug* yang terdapat pada sistem pelacak *eclipse bugzilla*.

Pengalian Data

Seleksi term. Seleksi *term* (*term selection*) digunakan untuk menemukan *term* representative dari dokumen. Ada tiga metode seleksi *term* yang banyak digunakan yaitu *tf-idf*, *tf.df*, dan tf^2. *Term frequency-inverse document frequency* (*tf.idf*) adalah skema pembobotan yang menghitung bobot *term* berdasarkan frekuensinya pada suatu dokumen dan seluruh koleksi dokumen. Perhitungan *tf.idf* didefinisikan dalam persamaan (Chen, Tseng, and Liang 2010):

$w(t,d) = tf(t,d) * log2(N/nt)$ Adapun penjelasan persamaan, adalah: *w(t,d)* adalah bobot dari *term* t dalam dokumen d. *tf(t,d)* adalah frekuensi *term* dalam dokumen*(tf)*. *N* adalah jumlah seluruh dokumen yang digunakan untuk penghitungan *idf*. *nt* adalah jumlah dari dokumen yang mengandung nilai *t*.

Langkah awal perhitungan tersebut adalah menghitung tf, kemudian menghitung df dan idf. Langkah terakhir menghitung nilai tf.idf, kemudian akan dibentuk matrik antara term dan dokumen. Dokumen yang dimaksud berupa atribut tingkat lebih tinggi.

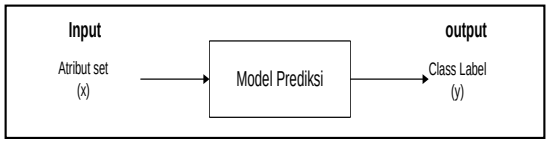
Infogain. Infogain adalah suatu nilai informasi yang diperoleh *term* dari perhitungan atribut-atribut untuk mengetahui tingkat kompleksitas. Perhitungan *infogain* ini bertujuan untuk mengukur tingkat kompleksitas suatu atribut dengan mencari nilai *item* terbaik untuk mempermudah konsep target (McCabe 1976). Perhitungan *infogain* digunakan rumus sebagai berikut:

$$Gain(S,A) = Entropy(S) - \Sigma_{\upsilon \epsilon \text{Values}} \frac{|S_v|}{S} \text{Entropy}(S_v)$$

Penjelasan rumus gain, *S* adalah Himpunan sampel. Dimana *A* adalah atribut. v adalah nilai yang mungkin untuk atribut *A*. Values(*A*) adalah himpunan yang mungkin untuk atribut *A*, |Sv| adalah jumlah sampel untuk nilai v. |S| adalah jumlah seluruh sampel dalam data. *Entopy(Si)* adalah *Entropy* untuk sampel yang memiliki nilai v. Sedangkan perhitungan nilai *Entropy* sebagai berikut:
$Entropy(S) = \Sigma_i^c \ -pilog_2pi$.

Penjelasan rumus entropy sebagai berikut: *c* adalah jumlah nilai yang terdapat pada atribut target atau jumlah kelas klasifikasi, p adalah peluang kemunculan dan *pi* adalah jumlah proporsi sampel atau peluang untuk kelas *i*. Entropy ini merupakan suatu parameter yang digunakan untuk mengukur tingkat keragaman dari kumpulan data. Jika suatu data semakin beragam, maka nilai Entropy akan semakin tinggi.

Model. Model penggalian sebuah data secara garis besar dapat dilihat pada gambar 3. *Input* pada proses prediksi adalah berupa kumpulan atribut. Setiap atribut memiliki karakteristik (x, y), di mana x adalah himpunan atribut dan y adalah atribut khusus berupa jenis kelas. Model prediksi adalah proses untuk memetakan himpunan atribut *input* x ke jenis kelas y.



Gambar 3. Model Prediksi

Model prediksi ini menggunakan Weka, sebab Weka memiliki kemampuan untuk analisa data dari data set. Tujuan pengalian data ini adalah untuk mencari term-term yang dapat mewakili isi document, sehingga dilakukan analisis hubungan antar dokumen. Cara yang ditempuh untuk mengidentifikasi atribut tingkat lebih tinggi yang signifikan pengaruhnya terhadap prediksi umur hidup *bug* digunakan praproses, perhitungan TFIDF, infogain, kemudian digunakan 4 klasifikasi untuk melihat data set yang terbentuk. Model prediksi menggunakan Zerro_R, One_R, Decision tree, dan naive bayes. Karena baik digunakan untuk dataset yang sangat besar.

Kappa. Ukuran akurasi atribut tingkat lebih tinggi untuk prediksi umur *bug* dari hasil klasifikasi akan menggunakan indek kappa. Indek kappa ini digunakan untuk memberikan pengukuran secara kuantitatif dari besarnya kesepakatan antar pengamat (Yang et al. 1999). Kappa ini, pertama muncul diperkenalkan oleh Kappa Cohen (Hussain 2007) tentang indek kappa sebagai ukuran kesepakatan antara atribut tingkat lebih tinggi yang selanjutnya disebut kappa. Adapun cara perhitungan Kappa dapat dilihat pada Persamaan 1 dan 2

$P(A) = \frac{a+d}{100}$, ...(1)

Dimana *a* adalah jumlah pengamat yang setuju dan hasilnya setuju, *d* adalah jumlah pengamat yang tidak setuju dan hasilnya tidak setuju.

$P(E) = [(\frac{n_1}{n}) * (\frac{m_1}{n})] + [(\frac{n_0}{n}) * (\frac{m_0}{n})]$..(2) Dimana n_1 adalah jumlah hasil yang setuju, n_0 adalah jumlah hasil yang tidak setuju, n adalah jumlah seluruh data, m_1 adalah jumlah pengamat yang setuju, m_0 adalah jumlah pengamat yang tidak setuju. Dari perhitungan pada persamaan 1dan 2 akan dihitung nilai Kappa dengan penghitungan pada Persamaan 3.

$k = \frac{P(A)-P(E)}{1-P(E)}$ .....(3)

Simbol *k* adalah lambang indek kappa. P(A) adalah proposisi berapa kali atribut tingkat lebih tinggi diamati setuju. P(E) adalah proporsi berapa kali dari atribut tingkat lebih tinggi diharapkan setuju. Ahli yang mengkategorikan nilai indek kappa, (Hussain 2007) ditunjukkan pada tabel 2.

Tingkat efektifitas suatu sistem pada klasifikasi diketahui dengan mengukur nilai akurasi. Nilai akurasi ini berupa nilai indek kappa dengan menambahkan atribut tingkat lebih tinggi diharapkan dapat mencapai nilai indek kappa pada proporsi kesepakatan tingkat cukup. Kappa ini disajikan sebagai ukuran dari perjanjian antara prediksi dan klasifikasi adalah benar. Nilai indek kappa akan digunakan sebagai ukuran akurasi di semua kelas hasil (*output*).

Tabel 2. Tingkatan Kreteria [7]

Nilai Indeks Kappa	Proporsi Kesepakatan
< 0	Rendah (*Poor*)
0.01 – 0.20	Sedikit (*slight*)

0.21 – 0.40	Cukup (*fair*)
0.41 – 0.60	Sedang (*moderate*)
0.61 – 0.80	Banyak (*substantial*)
0.81 – 1	Hampir Sempurna (*almost perfect*)

Berdasarkan interpretasi nilai indek kappa menunjukkan, bahwa hasil kappa yang diperoleh (Rish 2001) dari klasifikasi target data set nilai indek kappa rata-rata baru mencapai proporsi kesepakatan pada tingkat *fair*, yaitu nilai indek kappa dengan kisaran 0,21. Dalam penelitian ini dengan menambahkan atribut tingkat lebih tinggi diharapkan dapat mencapai proporsi *moderate* yaitu nilai indek kappa dengan kisaran 0,60. Tabel 3 adalah hasil klasifikasi (Panjer 2007) menggunakan lima algoritma, *Zerro_R, One_R,* Pohon Keputusan C4.5, dan *Naive Bayes*, sebagai awal prediksi umur *bug* dengan menggunakan atribut primitif. Tabel 3 Prediksi Kappa (Panjer 2007).

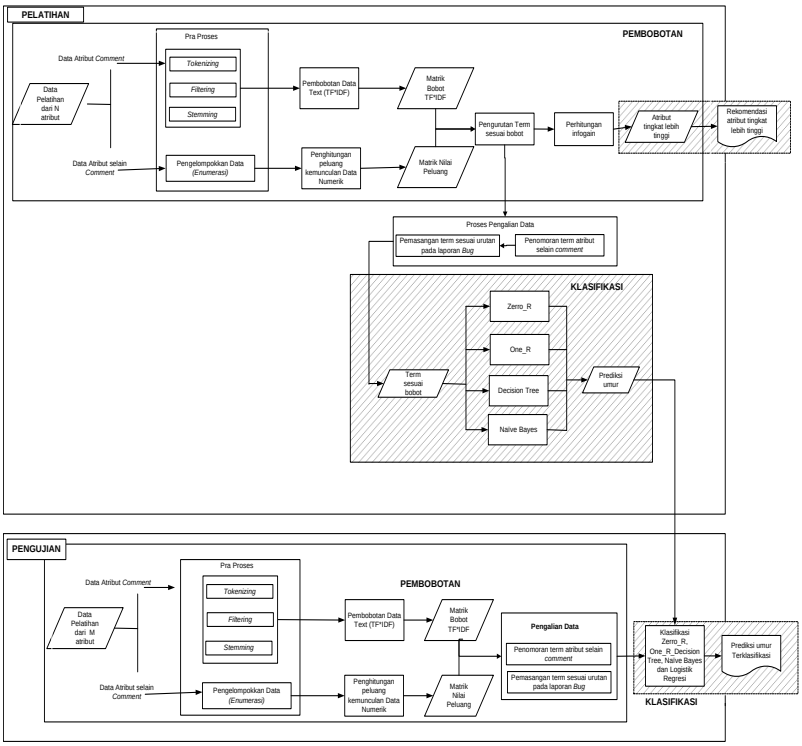
Tabel 3. Prediksi umur *bug*[12]

Algoritma	Prediksi (%)	Kappa
Zerro_R	29,1	0,0000
One_R	31,0	0,0747
Decion Tree	31,9	0,0938
Naïve Bayes	32,5	0,1195

METODE

Perancangan Sistem. Berdasarkan analisis atribut primitif dan identifikasi atribut tingkat lebih tinggi bahwa atribut *bug* pada awal prediksi ini terdapat 6 jenis atribut tingkat lebih tinggi yaitu atribut *Number_of_sentence_in_comment, Number_of_word in_comment, average_word_in_comment, Summary, Creation time* dan *Last change time*. Atribut ini merupakan atribut turunan dari atribut bug yang terdapat pada *eclipse bugzilla*. Atribut tingkat lebih tinggi ini dibutuhkan agar data yang ada menjadi lebih kecil sehingga proses komputasinya menjadi lebih ringan.

Selain itu jika atribut tingkat lebih tinggi ini disusun baik, maka hasil dari model diprediksikan bisa lebih baik. Proses pemilihan atribut tingkat lebih tinggi ini akan dikerjakan menggunakan WEKA, dimana WEKA merupakan alat preprosesing data yang digunakan untuk melakukan klasifikasi. Secara garis besar proses pemilihan atribut tingkat lebih tinggi ini, sebagai berikut:



Gambar 4. Perancangan sistem

Pemilihan atribut tingkat lebih tinggi ini mengacu pada serangkaian langkah yang pernah dilakukan oleh (Van Leekwijck and Kerre 1999). Langkah ini dipilih karena sudah terbukti dalam pemilihan atribut penting yang digunakan untuk menentukan tingkat *severity bug*, selain itu data yang digunakan juga sama yaitu data *bug*, sehingga memerlukan perlakuan yang hampir serupa tetapi terdapat perbedaan pada model dan tujuan dilakukan klasifikasi.

Perbedaannya yang pertama sebelum praproses. Data *comment* yang berupa *summary* dilakukan *tokenizing*, *filtering* dan *stemming*, tetapi untuk proses pemilihan atribut tingkat lebih tinggi ini dilakukan untuk persyaratan klasifikasi pada WEKA sehingga rangkaian praproses adalah terlebih dahulu memilih atribut bug untuk dijadikan dataset dengan cara *filter Remove, StringToNominal dan Discretize.*

Perbedaan yang kedua penentuan pola, penelitian ini menggunakan atribut tingkat lebih tinggi untuk klasifikasi (*zerro_R, one_R, decision tree, dan naive bayes*), sehingga didapatkan model data. Setelah model terbentuk berharap mendapat prediksi umur bug yang berupa rentang jumlah hari.

Parameter. Mengetahui tingkat keberhasilan dari penelitian ini diukur dengan parameter pembanding, yaitu Nilai Kappa jika diakhir proses identifikasi atribut tingkat lebih tinggi untuk prediksi umur bug menghasilkan indek kappa lebih besar dari nilai indek kappa (Rish 2001) 0,21 atau nilai interprestasi kappa mencapai proporsi kesepakatan *fair* seperti yang terterah tabel 2.6 tingkatan kreteria kappa, maka metode yang diusulkan reliabel untuk mempresentasikan ahli (manusia) dalam memprediksi umur *bug*.

HASIL DAN DISKUSI

Persiapan Dataset. Analisis dan uji coba model prediksi yang diusulkan, langkah yang perlu dilakukan adalah menentukan dataset yang sesuai yaitu berupa dokumen yang bisa dibaca oleh kakas bantu WEKA. Penelitian ini menggunakan standard dataset yang ada pada format WEKA untuk teknik *machine learning*. Informasi detail tentang dataset tersebut dapat dijelaskan sebagai berikut:

Tabel 4 merupakan data yang akan dijadikan sebagai acuan atribut tingkat lebih tinggi dalam menentukkan prediksi umur *bug* yang disimpan pada atribut *bug_lifetime*.

Tabel 4. Aktual umur hidup *Bug*

Label	Aktual Umur hidup *Bug*
1.	kurang dari 2 hari
2.	kurang dari 4 hari
3.	kurang dari 8 hari
4.	kurang dari 20 hari
5.	kurang dari 53 hari
6.	kurang dari 156 hari
7.	infinite (inf)

Atribut *bug_lifetime* ini dibuat berdasarkan dari atribut *last_change_time* yang dikurangi oleh atribut *creation_time*. Langkah berikutnya memetakan dengan data aktual prediksi umur *bug*. Tabel 4 adalah sebagai standar acuan untuk memetakan atribut *bug_lifetime*. *Atribut bug_lifetime* ini dibedakan menjadi 7 kelas prediksi umur *bug* dan kelas-kelas diberi nama label 1 sampai 7. Langkah kedua sebelum dilakukan pelatihan perlu untuk diketahui bahwa kakas bantu WEKA hanya bisa menerima sebuah file yang dalam bentuk format data ARFF. Tujuan dari proses ini adalah untuk mempermudah proses evaluasi. Sebuah file ARFF dalam WEKA memiliki atribut-atribut yang digunakan untuk mendeklarasikan jenis data yang dapat disimpan dalam setiap barisnya, maka judul informasi untuk file ARFF, sebagai berikut:

@RELATION *bug* 2010
@ATTRIBUTE *bug_id NUMERIC*
@ATTRIBUTE *assigned_to String*
@ATTRIBUTE *qa_contact String*
@ATTRIBUTE *priority* {?,P1,P2,P3,P4,P5}
@ATTRIBUTE *severity {blocker, critical, major, normal, minor, trivial, enhancement}*
@ATTRIBUTE *product String*
@ATTRIBUTE *component String*
@ATTRIBUTE *op_sys String*
@ATTRIBUTE *platform String*
@ATTRIBUTE version {'0.1','0.1.2','0.1.3','0.1.4','0.1.5','0.1.6','0.2','0.2.0','0.2.1',
'0.2.2','0.2.3','0.3','0.3.1','0.4','0.4.0','0.4.1','0.4.2','0.4.3','0.4.4','0.4.5','0.5','0.5.1','0.5.2','0.6','0.6.1'
,'0.6.2','0.7','0.7.0','0.8','0.9','0.x','1.0','1.0,Branch','1.1','1.2','1.3','1.3.1','1.4,Branch','1.5','1.5.0.x,
Branch','1.5R1','1.5R2','1.5R3','1.5R4','1.5R4.1','1.5R5','1.6','1.6R1','1.6R2','1.6R3','1.6R4','1.6R
5','1.6R6','1.7','1.7 Branch','1.7R1','1.8','1.8.0
Branch','1.8','Branch','1.9','1.9.0','Branch','1.9.1','Branch','1.9.2,Branch','1.x','2.0','2.0,Branch','
2.1','2.2','2.3','2.3.1','2.4','2.5','2.6','2.7','2.8','2.10','2.11','2.12','2.13','2.14','2.14.1','2.14.2','2.14.3','
2.14.4','2.14.5','2.15','2.16','2.16.1','2.16.2','2.16.3','2.16.4','2.16.5','2.16.6','2.16.7','2.16.8','2.16.9'

,'2.16.10','2.16.11','2.17','2.17.1','2.17.2','2.17.3','2.17.4','2.17.5','2.17.6','2.17.7','2.18','2.18.1','2.18.2','2.18.3','2.18.4','2.18.5','2.18.6','2.19','2.19.1','2.19.2','2.19.3','2.20','2.20.1','2.20.2','2.20.3','2.20.4','2.20.5','2.20.6','2.20.7','2.21','2.22','2.22.1','2.22.2','2.22.3','2.22.4','2.22.5','2.22.6','2.22.7','2.23','2.23.1','2.23.2','2.23.3','2.23.4','2.x','3.0','3.0.1','3.0.2','3.0.3','3.0.4','3.0.5','3.0.6','3.0.7','3.0.8','3.0.9','3.0.10','3.0.11','3.0','Branch','3.1','3.1.1','3.1.2','','3.1.3','3.1.4','3.1.11','3.1.12','3.2','3.2.1','3.2.2','3.2.3','3.2.4','3.2.5','3.2.6','3.2.7','3.2.8','3.2.9','3.2.10','3.3','3.3.1','3.3.2','3.3.3','3.3.4','3.4','3.4.1','3.4.2','3.4.3','3.4.4','3.4.5','3.4.6','3.4.7','3.4.8','3.4.9','3.4.10','3.4.11','3.5','3.5.1','3.5.2','3.5.3','3.5,Branch','3.6','3.6.1','3.6.2','3.6.3','3.6.4','3.6.5','3.6','Branch','3.7','3.7.1','3.7.2','3.7.3','3.7.4','3.7.5','3.7.6','3.7.7','3.7.8','3.8','3.8.1','3.8.2','3.8.3','3.9','3.9.1','3.9.2','3.9.3','3.9.4','3.9.5','3.9.6','3.9.7','3.10','3.10.1','3.10.2','3.11','3.11.1','3.11.2','3.11.3','3.11.4','3.11.5','3.11.6','3.11.7','3.11.8','3.11.9','3.11.10','3.11.11','3.11.12','3.11.13','3.11.14','3.12','3.12.1','3.12.2','3.12.3','3.12.3.1','3.12.3.2','3.12.4','3.12.5','3.12.6','3.12.7','3.12.8','3.12.9','3.12.10','3.12.11','3.12.12','3.13','3.x','4.0','4.0.1','4.0.2','4.0.3','4.0','Branch','4.1','4.1.1','4.1.2','4.1.3','4.2','4.2.1','4.2.2','4.2.3','4.2.4','4.2.5','4.2.6','4.2.7','4.2.8','4.2.9','4.2.10','4.3','4.3.1','4.3.2','4.3.3','4.3.4','4.3.5','4.4','4.4.1','4.4.2','4.4.3','4.4.4','4.4.5','4.5','4.5.1','4.5.2','4.5.3','4.5.4','4.5.5','4.6','4.6.1','4.6.2','4.6.3','4.6.4','4.6.5','4.6.6','4.6.7','4.6.8','4.6.9','4.6.10','4.7','4.7.1','4.7.2','4.7.3','4.7.4','4.7.5','4.7.6','4.7.7','4.8','4.8.1','4.8.2','4.8.3','4.8.4','4.8.5','4.8.6','4.8.7','4.8.8','4.8.9','4.8.10','4.9','4.x','5','5.0','5.x','5Branch','6.x','7.x','8.x','9.x','10.0','10.1','10.2','10.3','10.x','11.x','12.x','13.x','1998-03-31','1998-04-08','1998-04-29','1998-06-03','1998-07-28','1998-09-04','ACR-0.*','ACR1.*','BW1.1.x','BW1.2',
'BW1.2.1','BW2.0','Current','Deki','Development','Fennec1.1','Firefox4','Firefox 4.0','Firefox 5','Firefox 6','Firefox 7','Firefox 8','head','Lightning 0.3','Lightning 0.3.1','Lightning 0.5','Lightning 0.7','Lightning 0.8','Lightning 0.9','Lightning 1.0b1','Lightning 1.0b2','MDN','Mozilla1.8,Branch','other', 'Other,Branch','PocketPC','SeaMonkey 1.0 Branch','SeaMonkey 1.1 Branch','SeaMonkey 2.0 Branch','SeaMonkey 2.1 Branch','SeaMonkey 2.2 Branch','SeaMonkey 2.3 Branch','Sunbird 0.2','Sunbird 0.3','Sunbird 0.3.1','Sunbird 0.3a1','Sunbird 0.3a2','Sunbird 0.5','Sunbird 0.7','Sunbird 0.8','Sunbird 0.9','Sunbird 1.0b1','Trunk','trunk','unspecified','WinCE'}

@ATTRIBUTE target_milestone String
@ATTRIBUTE cc_count Numeric
@ATTRIBUTE dependent_bug Numeric
@ATTRIBUTE keywords Numeric
@ATTRIBUTE resolution String
@ATTRIBUTE creator String
@ATTRIBUTE creation_time DATE yyyy-MM-dd
@ATTRIBUTE last_change_time DATE yyyy-MM-dd
@ATTRIBUTE summary String
@ATTRIBUTEstatus{UNCONFIRMED,NEW,ASSIGNED,REOPENED,RESOLVED,VERIFIED,CLOSED}
@ATTRIBUTE number_of_word_in_comment NUMERIC
@ATTRIBUTE number_of_sentence_in_comment NUMERIC
@ATTRIBUTE average_word_in_comment NUMERIC
@ATTRIBUTE bug_lifetime {1,2,3,4,5,6,7}

File ARFF ini digunakan untuk menyimpan langsung data *bug* yang diambil dari *repositori eclipse bugzilla*. Gambar 4.1 adalah Informasi data yang disimpan dalam file ARFF. Gambar 5 ini menunjukkan satu baris data ARFF digunakan untuk menyimpan informasi satu *bug*.

```
@DATA
'537427','jeremy.orem+bugs@gmail.com','mrz@mozilla.com','?','major','mozilla.org','Server
Operations','Other','All','other','?','1','0','0','FIXED','justdave@mozilla.com','2010-1-1
0:57:0','2010-1-1 1:22:5','pm-app-amo24 constantly segfaulting','RESOLVED',71,3,23,1
'537441','pcvrcek@mozilla.cz','cs@communities.bugs','?','normal','Mozilla Communities','cs /
Czech','All','All','unspecified','?','0','0','0','FIXED','pcvrcek@mozilla.cz','2010-1-1
10:42:0','2010-1-1 23:7:11','Vytvořit stránku představující "Firefox Mobile"
(Fennec)','RESOLVED',13,1,13,1
'537444','nobody@mozilla.org','layout.fonts-and-text@core.bugs','?','normal','Core','Layout:
Text','Mac OS X','x86','unspecified','?','5','0','0','FIXED','jdawiseman@gmail.com','2010-1-1
12:42:0','2010-4-2 13:35:49','Missing &uArr; glyph in default <h2>
typeface','RESOLVED',59,18,3,6
'537449','dtownsend@mozilla.com','startup@toolkit.bugs','?','normal','Toolkit','Startup and Profile
System','Windows 7','x86','1.9.2
Branch','mozilla1.9.3a1','12','1','4','FIXED','alice0775@yahoo.co.jp','2010-1-1 16:3:0','2010-3-22
22:15:24','opening new (chrome) windows is broken after canceling window.onbeforeunload event (e.g.
Google Docs','RESOLVED',156,40,3,6
'537451','kazuki.ohta@gmail.com','jemalloc@core.bugs','?','normal','Core','jemalloc','Linux','x86',
ecified','mozilla2.0b10','6','0','1','FIXED','kazuki.ohta@gmail.com','2010-1-1 16:52:0','2011-3-7
18:38:0','jemalloc can be deadlock with fork(2)','RESOLVED',356,21,16,4
```

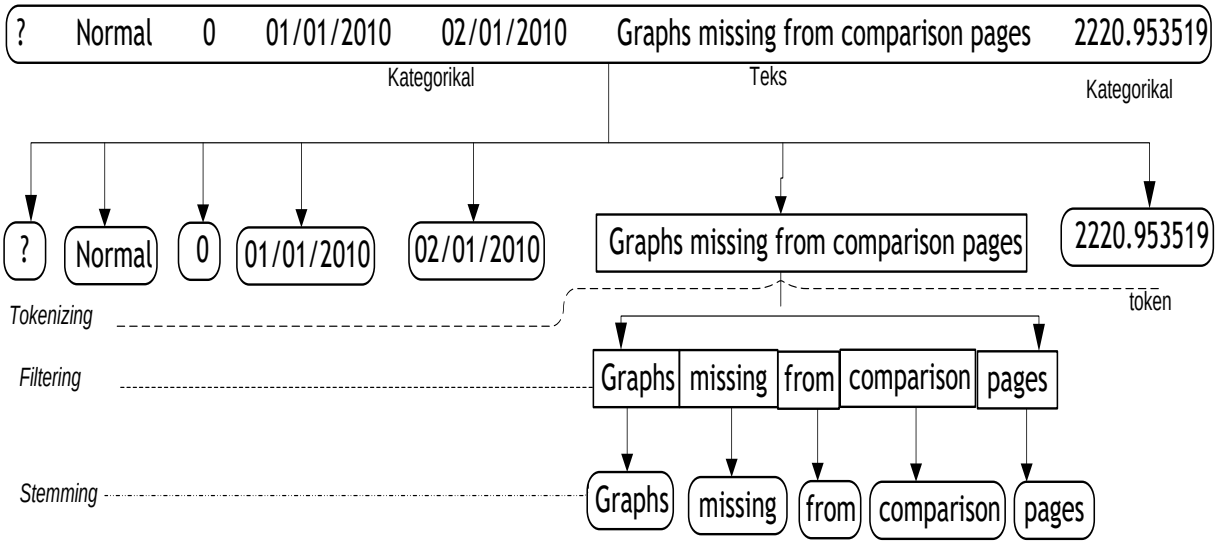
Gambar 5. informasi data *bug* dalam file

Praproses. Data *bug* didapatkan dari laporan Bugzilla merupakan dokumen yang berupa *string*. Data tersebut dapat dikategorikan menjadi dua, yaitu pertama dokumen yang berbentuk kalimat diperoleh dari atribut *comment* yang telah diolah menjadi bentuk atribut *summary.* dan kedua dokumen yang kategorikal diperoleh dari atribut selain *comment (*seperti *priority, severity, creation time, last change time, bug_lifetime)* lebih jelasnya lihat tabel 5 data *bug*.

Tabel 5. Potongan data *bug*

Atribut Priority 1:[nominal]	Atribut Severity 2:[nominal]	Atribut Keyword 3:[numeric]	Atribut Creation_time 4:[date]	Atribut Last_change_time 5:[date]	Atribut Summary 6:[string]	Atribut number_of_word_in_comment 7:[nominal]	Atribut Number_of_sentence_in_comment 8:[nominal]	Atribut Bug_lifetime 9:[nominal]
?	major	0	01/01/2010	01/01/2010	pm-app-amo24 constantly segfaulting	71	3	1
?	normal	0	01/01/2010	01/01/2010	VytvoÅ™it strÃ¡nku pÅ™edstavujÃ-cÃ- "Firefox Mobile" (Fennec)	13	1	1
?	normal	0	01/01/2010	02/04/2010	Missing ⇑ glyph in default (h2) typeface	59	18	6
?	normal	0	01/01/2010	26/06/2010	Port Bug 227305 (Support drag-drop single message to desktop / file-system window)	30	3	7
?	normal	4	01/01/2010	22/03/2010	(e.g. Google Docs	156	40	6
?	normal	1	01/01/2010	07/03/2011	jemalloc can be deadlock with fork(2)	356	21	7
?	normal	0	01/01/2010	14/01/2010	switch CC* release automation to chatzilla-from-hg	54	6	4
?	major	0	01/01/2010	13/01/2010	Missing munin hosts on nm-munin01	12	50	4
?	normal	0	01/01/2010	05/01/2010	Please add TMZ's (Tom Ellins's) blog to planet.	13	4	7
?	normal	0	01/01/2010	19/12/2010	Russian Holidays 2010-2020	8	1	7
?	normal	3	01/01/2010	22/03/2010	Box shadow with spread radius is displayed without all given values	108	25	7
?	normal	0	01/01/2010	02/01/2010	Graphs missing from comparison pages	22	2	1
?	minor	0	01/01/2010	04/01/2010	Cannot open Browse all Add-ons with browser.link.open_newwindow set to 1	131	23	7
?	major	0	01/01/2010	05/04/2010	All "topcrash by signature reports" are returning no data beginning on 1 Jan 2010	112	5	6
?	normal	1	02/01/2010	10/01/2010	Don't ship dictionaries with language packs	47	1	4
?	normal	0	02/01/2010	03/01/2010	JSON.parse doesn't correctly add properties with numeric identifiers	70	12	1
?	normal	0	02/01/2010	10/01/2010	nsSigHandler.cpp should use _M_IX86 instead of _M_IA32	52	6	4
?	normal	1	02/01/2010	17/12/2010	Bookmarks and History Sidebars Empty on Initial Opening	167	33	7
P1	trivial	0	02/01/2010	05/01/2010	Not translated "Localization Dashboard"	7	5	2
P2	normal	0	02/01/2010	02/01/2010	Polish localizer team	13	1	1
?	normal	0	02/01/2010	02/01/2010	ZpÅ™ehlednit strÃ¡nku "Ke staÅ¾enÃ-"	51	9	1
?	normal	0	02/01/2010	02/01/2010	VytvoÅ™it detailnÃ- strÃ¡nky "Ke staÅ¾enÃ-"	27	3	1

Penggalan data laporan Bugzilla dapat dilihat gambar 5 atribut *comment* merupakan dokumen yang berbentuk kalimat, sedangkan atribut *priority, severity, creation time, last change time,* dan *bug_lifetime* merupakan dokumen yang dapat dikategorikan.



Gambar 5. Pengalan Data

Analisa Data. Penggalian data atau *data mining* ini digunakan untuk menemukan pola pada data yang berjumlah sangat besar. Proses penggalian data memerlukan dataset sebagai kumpulan data yang akan dicari polanya. Dataset yang digunakan terdiri 24 atribut primitive dan 1000 *bug*, tetapi setelah melalui proses analisis,bahwa tidak semua atribut primitif bisa dijadikan atribut tingkat tinggi sebab:

Atribut mengandung suatu data yang variatif jumlahnya bisa mempengarui nilai infogain menjadi urutan tertinggi, misal *Bug_id* tidak bisa dijadikan atribut tingkat lebih tinggi karena jangkauan nilai yang terdapat pada *bug_id* ini tergantung dari banyaknya *bug* yang dilaporkan ke sistem pelacak *Bugzilla*, sehingga setiap pelapor bisa lebih dari satu *bug* yang dilaporkan. Peluang tersebut menyebabkan jangkauan nilai *bug_id* bervariasi. Atribut yang tergolong mengandung suatu data yang variatif ini, yaitu *Bug_id* dan *dependent_bug*.

Atribut *bug* yang mengandung banyak jenis nama seperti atribut *assigned_to* tidak bisa dijadikan atribut tingkat lebih tinggi sebab atribut tersebut isinya tentang ragam nama email dari pelapor. Atribut yang tergolong dalam banyak jenis nama ini, yaitu, dan *assigned_to, qa_contact, product, component*, Op_sis, *platform* dan *creator.*

Atribut *bug* yang mengandung perubahan titik pekerjaan dari *bug* yang terdapat pada siklus *Bugzilla*, seperti *Bug_status* tidak bisa dicari atribut tingkat lebih tinggi sebab semua status pada *bug* baru tidak memiliki status dan resolusi. Atribut *bug* yang tergolong dalam jenis perubahan titik pekerjaan yang terdapat pada siklus sistem pelacak *Bugzilla*, yaitu *Bug_status* dan *Resolution.*

Berdasarkan analisisdan Identifikasi hasil proses penggalian data atribut primitif diberi nama atribut tingkat lebih tinggi. Atribut tingkat tinggi ini meliputi atribut, *number_of_sentence_in_comment, number_of_sentence_in_comment, average_word_in_comment, summary, creation_time*, dan

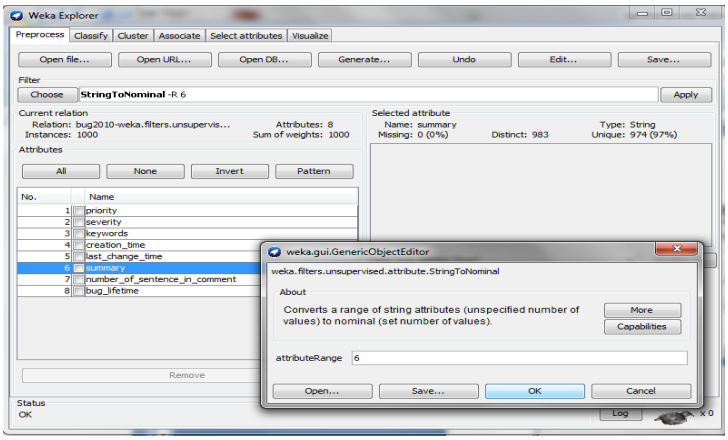
last_change_time. Sedangkan atribut *priority* dan *severity* ini tidak termasuk dalam atribut tingkat tinggi karena dua atribut ini merupakan terkategori dalam atribut tingkat rendah.Cara penggalian data atribut tingkat lebih tinggi diambil dari atribut primitif dengan pencarian nilai infogain yang diproses melalui kakas bantu WEKA. Atribut dari hasil pemilihan infogain yang terbaik akan dimodelkan untuk prediksi. Model prediksi umur *bug* menggunakan beberapa metode penggalian data, yaitu *Zerro-R, One-R, Decision Tree (j48)*, dan *Naive Bayes*. Masing-masing metode digunakan untuk menemukan model umur *bug* dari data mentah *bugzilla*. Berikut ini penjelasan proses penggalian data menggunakan kakas bantu WEKA.

Data diambil dari webservice Bugzilla dengan cara bertahap dengan tujuan untuk meminimalkan kehilangan data. Karena, jika pengambilan data dari webservice Bugzilla terjadi kerusakkan dari salah satu atribut, maka keseluruhan proses satu kali permintaan akan dihentikkan.

Data yang didapatkan dari *webservice Bugzilla* dipotong-potong untuk dimasukkan ke dalam variabel-variabel yang sesuai, kemudian disimpan dalam file ARFF.

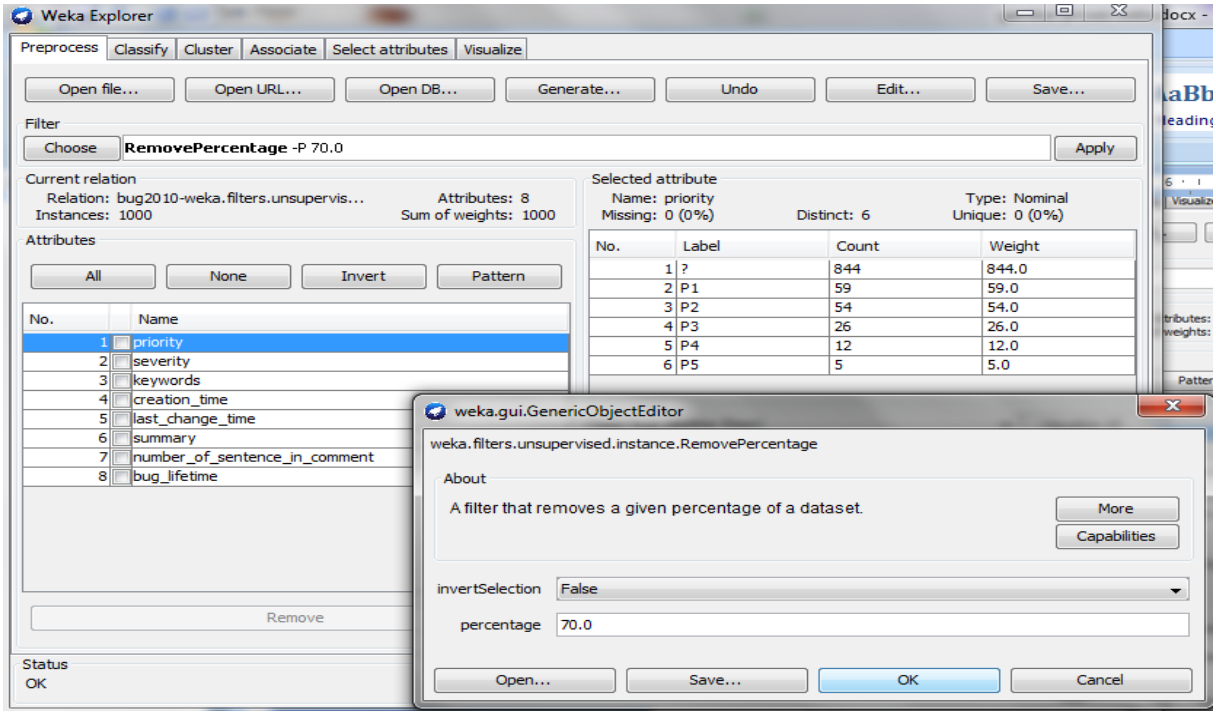
File ARFF diinisialisasi dalam kakas bantu. Proses inisialisasi akan menghasilkan object instance. Instance ini yang akan diolah melalui praproses seperti tahapan pada sub.bab 3.2.5 yaitu proses filterisasi untuk mempersiapkan data agar dapat diterima oleh masing-masing metode klasifikasi.

Proses filterisasi WEKA terdiri 3 proses yaitu *StringtoNominal*, *Discretize string* dan *Remove*. *StringtoNominal* ini digunakan untuk merubah atribut tipe menjadi nominal, seperti pada gambar 6. *Discretize* digunakan untuk mengubah atribut yang bertipe numerik menjadi tipe nominal. Sedangkan *Remove* digunakan untuk menghapus beberapa atribut dari suatu dataset yang tidak diperlukan.



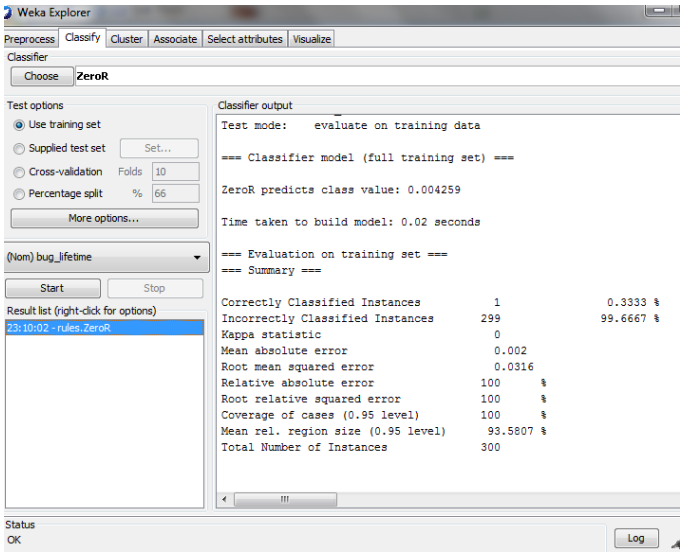
Gambar 6. Proses Perubahan tipe data dalam WEKA

Jika proses filterisasi selesai, maka perlu dilakukan pembagian data pelatihan (*traning*) dan data pengujian (*testing*) yaitu menggunakan *RemovePrecentage.* Cara yang ditempuh untuk pembagian data yaitu *wekafilter.unsupervised.instance.RemovePrecentage*. Hasil pembagian data masing-masing disimpan berdasarkan prosentasi. 70% digunakan untuk penentuan data *traning*, sedangkan 30% digunakan untuk penentuan data *testing*. Seperti gambar 7.



Gambar 7. Proses penentuan data *training*

Setelah dilakukan proses filterisasi dataset siap digunakan untuk penggalian data dengan metode klasifikasi agar mendapatkan model prediksi, seperti gambar 8. Untuk melakukan model prediksi diperlukan tiga tahapan penggalian yaitu, *use training set*, *supplied test set* dan *cross_validation*.



Gambar 8. Model Prediksi

Model prediksi dievaluasi menggunakan Kappa statitik untuk mengetahui reliabel metode yang diusulkan dalam mempresentasikan ahli (manusia) pada prediksi umur bug. Serangkain pengujian data *training* dengan metode yang diusulkan diperlihatkan hasil pada table data *training*. Tabel hasil kappa ini menunjukkan hasil, bahwa *naive bayes* menghasilkan tingkatan kreteria kappa *substantial* pada atribut *high_level* 0,74; atribut *high_low level* 0,75 dan 24 atribut *bug* menghasilkan nilai kappa 0,76, hal ini menyatakan *naive bayes* memberi hasil yang baik dan tingkat akurasi yang lebih tinggi, jika diujicobakan pada data yang attribut-attributnya terdapat korelasi. Sedangkan pada kappa statistik atribut *low_level* sejumlah 0,54 menunjukkan metode *naive bayes* kurang baik jika terdapat atribut yang tidak berkorelasi.

Tabel 5. Hasil Kappa Data Training

Clasify				
Data Training	*Zerro_R*	*One_R*	*j48*	*Naive Bayes*
Kappa statistic Attribute Bug	0	0,9865	0,4213	0,7627
Kappa statistic Low Level Attribute Bug	0	0,4303	0,5396	0,5396
Kappa statistic High Level Attribute Bug	0	0,9865	0,4415	0,7412
Kappa statistic High_Low Level Attribute Bug	0	0,9865	0,4415	0,7502

Tabel 6. Hasil Kappa Data Testing

Clasify				
Data Testing	*Zerro_R*	*One_R*	*j48*	*Naive Bayes*
Kappa statistic Attribute Bug	0	0,9803	0,4769	0,8127
Kappa statistic Low Level Attribute Bug	0	0,5486	0,6758	0,6055
Kappa statistic High Level Attribute Bug	0	0,9803	0,4728	0,7282
Kappa statistic High_Low Level Attribute Bug	0	0,9803	0,4728	0,6881

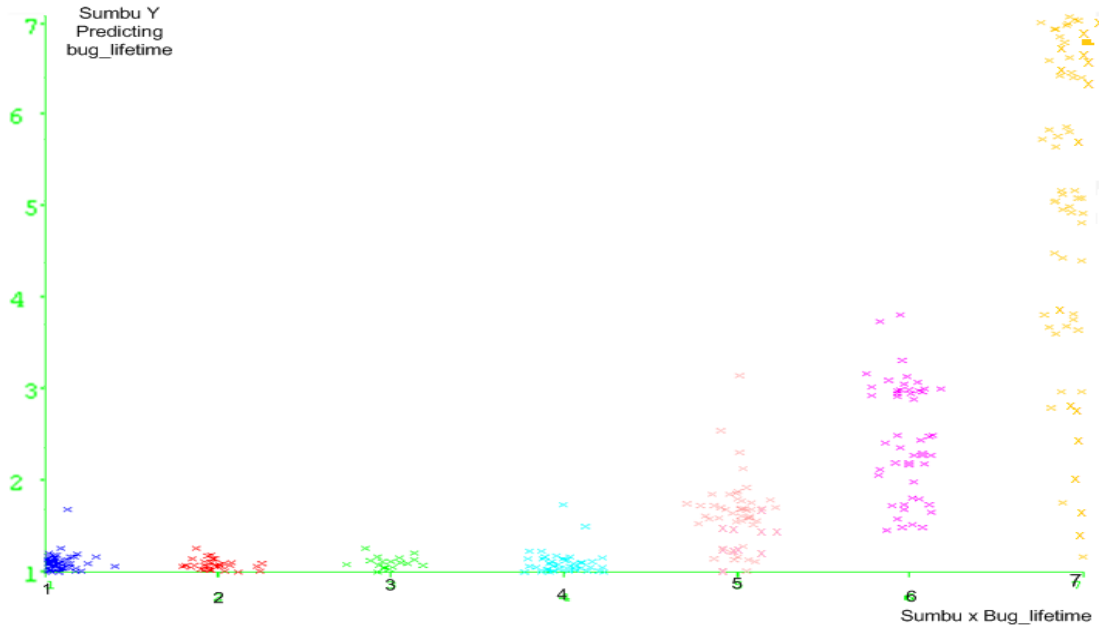
Berdasarkan tujuan *infogain* digunakan untuk mengukur tingkat kompleksitas atribut dengan mencari nilai *item* terbaik untuk mempermudah konsep target, maka proses pencarian nilai infogain menghasilkan nilai tertinggi atribut *summary* 2,67 dan terendah *number_of_sentence_in comment* 0,02. Penelitian ini nilai tertinggi adalah 2,7 dan kappa statitik nilainya 0,81 maka atribut yang memiliki nilai 2,7 atau mendekati 2,7 dianggap memiliki atribut tingkat lebih tinggi. Nilai infogain dari masing-masing atribut adalah sebagai berikut:

(1.) *summary* : 2,69
(2.) *creator* : 1,88
(3.) *assigned_to* : 1,59
(4.) *qa_contact* : 1,55
(5.) *component* : 1,47
(6.) *last_change_time* : 1,2

KESIMPULAN

Studi ini menyimpulkan bahwa perolehan informasi (*infogain*) atribut-atribut hasil penelitian mengidentifikasi 6 atribut tingkat tinggi, dimana 2 diantaranya (*summary*, dan *last change time*) dianggap memiliki pengaruh yang signifikan dalam memprediksi umur *bug*. Kombinasi atribut high_level (2 atribut), atribut low_level (3 atribut) dan atribut bug (1) menghasilkan nilai kappa 0,81. Oleh karena itu metode yang diusulkan dapat bekerja lebih baik dari penelitian sebelumnya yang menghasilkan nilai kappa 0,60. Hasil infogain menunjukkan jumlah *term* tidak berpengaruh terhadap

tingkat akurasi prediksi, tetapi jenis *term* yang unik yang dapat meningkatkan akurasi prediksi umur *bug.* Prediksi umur *bug* menggunakan atribut tingkat tinggi ditunjukkan gambar 9 yang terdiri 7 kelas *bug_lifetime*.



Gambar 9. Prediksi umur *bug*

Tujuh kelas *bug_lifetime* ini terdiri kelas 1 menunjukkan prediksi umur *bug* kurang dari 2 hari, kelas 2 menunjukkan predisi umur sebuah bug kurang dari 4 hari, kelas 3 menunjukkan prediksi umur sebuah bug kurang dari 8 hari, kelas 4 menunjukkan prediksi umur sebuah bug kurang dari 20 hari, kelas 5 menunjukkan umur sebuah bug kurang dari 53 hari, kelas 6 menunjukkan prediksi umur sebuah bug kurang dari 156 hari dan umur bug yang lebih dari 156 hari terkategori kelas 7 atau *infine.* Atribut tingkat tinggi berhasil memprediksi umur bug sesuai ketentuan dari pemetaan kelas bug_lifetime bahwa umur sebuah bug bisa memanjang sampai tahun yang dibuktikan pada gambar 9 umur bug yang sering muncul pada kelas 7 (infinite). Berdasarkan nilai kappa dan hasil prediksi menunjukkan pengujian identifikasi atribut tingkat lebih tinggi berdasarkan nilai perolehan informasi sudah sesuai target yang diinginkan yaitu tingkat kreteria *moderate* atau *substantial.*

DAFTAR PUSTAKA

Chen, Chun-Ling, Frank S C Tseng, and Tyne Liang. 2010. "Mining Fuzzy Frequent Itemsets for Hierarchical Document Clustering." *Information processing & management* 46(2): 193–211.

Chou, Andy et al. 2001. "An Empirical Study of Operating Systems Errors." In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, , 73–88.

Humphrey, Watts S. 2005. "A Personal Commitment to Software Quality." In *Software Engineering—*

ESEC'95: 5th European Software Engineering Conference Sitges, Spain, September 25–28, 1995 Proceedings, Springer, 5–7.

Hussain, H M. 2007. "Using Text Classification to Automate Ambiguity Detection in SRS Documents."

Kim, Sunghun, and E James Whitehead Jr. 2006. "How Long Did It Take to Fix Bugs?" In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, , 173–74.

Van Leekwijck, Werner, and Etienne E Kerre. 1999. "Defuzzification: Criteria and Classification." *Fuzzy sets and systems* 108(2): 159–78.

McCabe, Thomas J. 1976. "A Complexity Measure." *IEEE Transactions on software Engineering* (4): 308–20.

Murphy, G, and Davor Cubranic. 2004. "Automatic Bug Triage Using Text Categorization." In *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, Citeseer, 1–6.

Panjer, Lucas D. 2007. "Predicting Eclipse Bug Lifetimes." In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, IEEE, 29.

Rish, Irina. 2001. "An Empirical Study of the Naive Bayes Classifier." In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, , 41–46.

Tan, P N, M Steinbach, and V Kumar. 2006. "Introduction to Data Mining, Addison Wesley Publishers."

Yang, Yiming et al. 1999. "Learning Approaches for Detecting and Tracking News Events." *IEEE Intelligent Systems and their Applications* 14(4): 32–43.