ISSN: 2477-5126
e-ISSN: 2548-9356

# Analisa Studi Empirik Pengukuran Kualitas Perangkat Lunak Bebas Cacat

**Agus Pamuji**[*)]
Jurusan Informatika, Fakultas Teknik Dan MIPA, Universitas Indraprasta PGRI, Jakarta
Jln. Raya Tengah , Gedong, Kota Jakarta Timur, 13000, Indonesia
email: jurnal.agus.pamuji@gmail.com

***Abstract* – Testing activitiy is a strategic step to determine software quality was generated, so that is accepted by the end user. In the testing an errors were found that may be cause to risk a defect on the software. This study was conducted by establishing a measurement framework to analyze software metrics test toward risk prediction of defects consisting of defect density, defect removal, and Line of code. In the analysis, the data set contains 53 module samples through a statistical approach with correlation analysis techniques. Based on the hypothesis were proposed, that there are only 2 of 3 items is received and shows a high significance of defect density and removal of defects towards software quality measurement.**

***Abstrak* – Aktifitas pengujian merupakan suatu langkah strategi untuk menentukan kualitas perangkat lunak yang dihasilkan agar diterima oleh pengguna. Didalam pengujian sering ditemukan adanya kesalahan yang dapat menimbulkan resiko cacat. Penelitian ini dilakukan dengan membuat suatu kerangka kerja pengukuran untuk menganalisa uji metrik perangkat lunak terhadap prediksi resiko cacat yang terdiri dari kepadatan cacat, penghapusan cacat dan baris kode program. Didalam analisa, set data berisi 53 sampel modul di analisa melalui pendekatan statistik dengan teknik analisa korelasi. Berdasarkan hipotesis yang diusulkan, bahwa hanya ada 2 dari 3 item yang diterima serta menunjukan signifikasi yang tinggi yaitu kepadatan cacat dan penghapusan cacat terhadap pengukuran kualitas perngkat lunak.**

***Kata Kunci* – Analisa, kepadatan, penghapusan, cacat, pengukuran kualitas, baris ko program, korelasi.**

## I. PENDAHULUAN

Pengujian merupakan suatu proses pelaksanaan pemeriksanaan suatu program dengan tujuan mencari kesalahan[1],[2]. Walaupun demikian, pengujian merupakan proses pengembangan perangkat lunak yang tidak murah serta banyak menghabiskan waktu[3]. Hal tersebut disebabkan dari 50% dipakai untuk jadwal pengujian [4],[5]. Perngkat lunak yang mengalami cacat dapat berakibat biaya pengembangan, perawatan [6], dan estimasi semakin meningkat, serta menurunkan kualitas perangkat lunak [7] .

Saat ini pengembangan perangkat lunak sedang berkembang dengan cepat [8]. Hal ini juga ditandai dengan pertumbuhan jumlah cacat yang cepat. Kualitas perangkat lunak menjadi suatu hal yang sangat penting untuk pengembangan suatu perusahaan [9]. Kualitas perangkat lunak bergantung pada kode sumber program yang ditulis dalam teks editor. Kode sumber program ini ditulis dengan memakai bahasa pemrograman serta disesuaikan dengan sintak serta prosedur yang ada didalamnya. Apabila kode sumber ditulis sesuai dengan sintak dan kaidah maka hal ini dapat mengurangi jumlah kesalahan sehingga dapat meningkatkan kualitas.

Kualias perangkat lunak dapat diukur berdasarkan dari jumlah cacat yang ditemukan selama proses pengujian [10]. Sedangkan cacat itu berasal dari kode sumber yang menjadi kontributor utama kegagalan perangkat lunak sehingga menyebabkan pengerjaan proyek perlu diulang, proyek mengalami keterlambatan, dan biaya pengembangan meningkat [11]. Kemampuan dalam merubah merupakan kemudahan dengan kode sumber yang dapat dirubah. Hal ini dapat dievaluasi melalui perhitungan matriks dari riwayat perubahan yang telah dibuat. Walaupun demikian, usaha menemukan cacat perangkat lunak dapat dilakukan dengan debugging, yaitu pencarian dan menemukan dengan melibatkan semua kode sumber program.

Terkait dengan perubahan kode sumber, tim pengembang banyak menghabiskan waktu untuk bisa memahami dan merubah kode sumber secara bergantian selama proses pengujian, hal ini menimbulkan kerawanan bug yang belum dipertimbangkan secara kuantitatif [12]. Kerawanan ini dapat berakibat kegagalan fungsi sistem untuk berkerja dengan baik dan menimbulkan cacat. Kegagalan fungsi sistem ini terjadi ketika sebuah kesalahan kode program diekseskusi dan masih memiliki hubungan dengan baris kode program lain.

Oleh sebab itu, penelitian ini menggunakan pendekatan *software metric* sebagai kerangka kerja untuk mngukur kualitas perangkat lunak [13]. *Software metric* ini memiliki peran untuk menampung beraneka macam atribut serta dapat digunakan untuk mengukur berbagai macam atribut dalam siklus hidup pengembangan (Ahmed, 2010).

Pada *software metric* terdapat atribut sebagai skala pengukuran perangkat lunak yaitu: (1) Matrik kualitas perangkat lunak berbasis pada cacat; (2) Matrik Kegunaan (*Usability Metric*); (3) Pengujian Matrik (*Testing Metrics*); dan (4) baris kode program (*Line of Code*). Matrik kualitas perangkat lunak berbasis cacat meliputi (1) kepadatan cacat ( *defect density*); dan (2) Penghapusan Cacat ( *Defect removal*) [13],[14]. Sedangkan matrik kegunaan meliputi persentase efektifitas tugas. Efisiensi temporal dan periode produktivitas, efisiensi pengguna relatif. Pengujian matrik meliputi *Test Focus* (TF), cakupan kesalahan matrik (*fault coverage metric*). LOC ( *Line of Code*) berupa baris kode program berisi baris kode yang dieksekusi, serta baris komentar untuk

*) penulis korespondensi (Agus Pamuji)
Email: jurnal.agus.pamuji@gmail.com

memberikan pemahaman dan mampu dibaca oleh setiap anggota pengembang [15]. Keempat parameter ini akan diklasifikasikan dengan algoritma *K-Nearest Neighboor* sehingga didapat mana yang berpengaruh dalam pengujian dan pengukuran perangkat lunak agar bebas cacat.

Kepadatan cacat mencakup rasio jumlah cacat dengan ukuran perangkat lunak. Sejumlah cacat mengukur banyaknya cacat yang terdeteksi selama proses pengujian [8]. Tingkat kecacatan dapat diukur sebagai cacat yang dialami selama periode tertentu, sebagai contoh per bulan, per minggu. Penghapusan cacat ( *defect removal* ) dapat didefinisikan sebagai cacat yang dihapus pada fase siklus hidup berbanding dengan cacat laten [10].

## II. PENELITIAN YANG TERKAIT

Prediksi cacat secara umum digunakan sebagai panduan uji coba perangkat lunak [12]. Dalam memprediksi cacat perangkat lunak, software metrics digunakan pada variabel bebas (*independent variable*) sedangkan cacat digunakan pada variabel terikat (*dependent variable*).

Analisa cacat dilakukan dengan penghapusan cacat diuji coba oleh S. Kumaresh dan R. Baskaran dimana melakukan penelitian secara eksperimen untuk menganalisa cacat pada saat ingin memperbaiki proses pengembangan perangkat lunak . didalam penelitiannya menggunakan model penghapusan cacat untuk memprediksi jumlah cacat yang masih tersisa pada setiap fase keluar. Walaupun demikian, model tersebut digunakan dengan tujuan mengurangi jumlah cacat yang ditunjang dengan strategi untuk proses perbaikan. Penelitian yang dilakukan lebih fokus pada penghapusan cacat. Penghapusan cacat ini diterapkan bersamaan dengan strategi perbaikan proses yang memiliki 4 fase yaitu (1) *requirement phas;* (2) *design phase;* (3) *coding phase; dan* (4) *testing phase*. Terkait penghapusan cacat, terdapat 2 indikasi yaitu adanya penyuntikan cacat dan penghapusan cacat, kemudian ditampilkan dalam bentuk matrik data cacat[16]. Akhir hasil penelitiannya bahwa, cacat yang tersebesar selama proses pengujian adalah fase pengkodean dan hasilnya sangat signifikan dalam peningkatan peluang cacat perangkat lunak.

Prediksi cacat perangkat lunak juga dilakukan oleh Gabriel Kofi Armah dkk (2013) dengan menggunakan multi level data pra proses diterapkan pada awal fase pengembangan. Hasil penelitiannya terbukti secara efisien mampu mengklasifikasi modul yang cacat dan yang tidak cacat , dengan menyeleksi dan membandingkan dengan set data non pra proses [17]. Data pra-proses dan data non pra-proses diseleksi dan dibandingkan dengan menggunakan metode KNN Nearest Neighbor. Hasil observasi selama pengujian bahwa didalamnya ada atribut yang ganda sehingga perlu dilakukan penghapusan. Penghapusan dilakukan untuk atribut yang dianggap tidak relevan. Penghapusan dilakukan dengan teknik resampling, dan dimesi reduksi serta menganalisa ketidakseimbangan kelas.

Tahun 2014, prediksi cacat dilakukan oleh Ruchika Malhotra dengan mengeksplorasi kemampuan prediksi komputasi evelusioner hibridasasi. Didalam penelitiannya dengan 15 komputasi evolusioner hidbridasi ke 5 kumpulan set data ya ng diperoleh dari Apache Software Foundation menggunakan koleksi cacat dan sistem pelaporan. Hasil penelitiannya menunjukan dapat berjalan dengan baik terbukti melalui seleksi perangkingan menggunakan metode Friedman.

Penelitian yang dilakukan oleh Qimeng Cao, dkk (2015), menggunakan teknik TCANN (*Transfer Component Analysis Neural Network*) dalam memprediksi cacat perangkat lunak. Hal ini dilatar belakangi oleh ketidakseimbangan kelas dan perbedaan distribusi fitur diantara sumber daya dan tujuan proyek. Teknik yang dipakai oleh Qimeng Cao secara memadai untuk menanggulangi kelas yang tidak seimbang dan data yang kacau [10]. Ada 3 langkah dalam mengatasi msaalah tersebut. Pertama, menggunakan metode berbasis *Inter Quartile Range* (IRQ) dengan tujuan untuk menghapus kegaduhan data. Sedangkan yang kedua, *Transfer Component Analysis Neural Network* (TCANN) digunakan untuk mengurangi perbedaan distribusi fitur antara sumber daya dengan target proyek (data). Terakhir, menggunakan *Dynamic Sampling Neural Network* untuk menanggulangi ketidakseimbangan kelas set data latih. Hasil menunjukan bahwa metode yang diusulkan mampu memperbaiki kinerja yang ada didalam proyek maupun antar proyek dalam perbandinnggannya dengan metode lain.

Kepadatan cacat sudah pernah diteliti oleh Neeraj Mandhan dkk (2015) yaitu menggunakan metrik statis. Matrik statis yang digunakan ada 7 yaitu (1) *coupling*; (2)*depth*; (3)*cohesion*; (4)*response*; (5)*weighted method*; (6)*comments*; dan (7) *line of code*. Set data yang digunakan sebagai bahan eksperimennya berasal dari NASA PROMISE yang terdapat 20 metrik berbeda [8]. Dalam menganalisa prediksi kepadatan cacat, maka teknik atau metode regresi linier. Hakikatnya metrik statis digunakan untuk ekstrak informasi abstraks dari baris kode. Didalam penelitiannya, melakukan eksperimen untuk melihat adanya hubungan antara metrik statis dengan kepadatan cacat baik secara individual maupun gabungan. Relasi ini digunakan untuk analisa jumlah cacat. Terkait dengan teknik yang dipakai dalam penelitiannya, tim peneliti melakukan uji normalitas dari setiap bagian matrik statis, uji regresi tunggal dan ganda untuk melihat prediksi secara individu maupun gabungan. Tujuannya adalah matrik statis mana yang lebih berguna dan kurang berguna serta melihat korelasi positif dan negatif terhadap cacat. Hasil akhir menunjukan bahwa terdapat pengaruh yang signifikan antara kepadatan cacat dengan matrik statis.

Pada penelitian ini akan diterapkan kerangka kerja dalam upaya pengukuran kualitas supaya bebas dari cacat. Pengukurannya dilakukan dengan prediksi cacat melalui analisa kepadatan cacat (*defect density*), peghapusan cacat ( defect removal, baris kode program (*line of code*).

## III. METODE PENELITIAN

Didalam penelitian ini dilakukan dengn mengusulkan model atau kerangka kerja sebagai upaya perbaikan serta kontribusi untuk menjaga kualitas perangkat lunak. Metrik proses dasar meliputi jumlah desain kasus uji, jumlah uji kasus dieksekusi, jumlah uji kasus lolos, dan jumlah kegagalan uji kasus. Metrik dasar dan pengukuran perangkat lunak selanjutya akan dilakukan uji hipotesis terhadap kualitas didukung dengan analisa data melalui pendekatan statistik menggunakan analis korelasi.

Pengolahan data dilakukan dengan perangkat lunak bantu SPSS melalui pendekatan statistik. Terkait dengan pengukuran, maka ada tiga variabel menjadi tolak ukur yaitu

ISSN: 2477-5126
e-ISSN: 2548-9356

kepadatan cacat, dan penghapusan cacat serta baris kode program. Ketiga variabel ini akan diuji normalitas yang selanjutnya melalui korelasi antara kedua variabel tersebut. Strategi pengumpulan data dilakukan melalui teknik observasi atau inkuisitif berada pada tingkatan pertama yaitu bentuk kusioner dan di padukan dengan eksperimen.

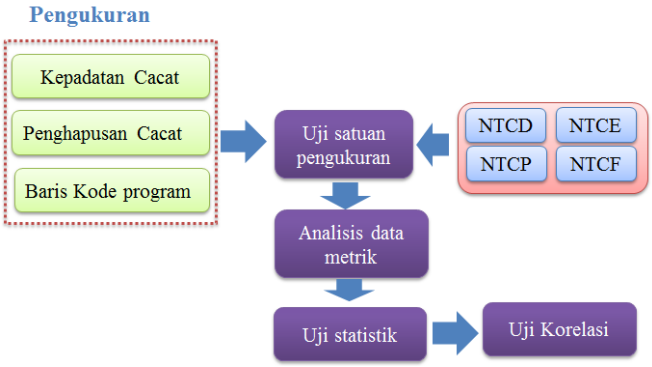
$$Defect\ Density = \frac{Number\ of\ Defect}{KLOC\ in\ Line\ Code} \quad (1)$$

Cacat adalah sebuah kondisi secara kebetulan menyebabkan sebuah unit gagal untuk menjalankan fungsinya [8]. Bentuk kepadatan cacat dapat ditentukan persaman diatas dimana jumlah cacat dibagi baris kode program. Walaupun cacat ditentukan jumlahnya namun disisi lain dapaat dihapuskan melalui persamaan 2 dan persamaan 3 [14].

$$DRE = \frac{defect\ removed\ in\ a\ given\ life\ cycle\ phase}{latent\ defect} \quad (2)$$

$$ideal\ DRE = \frac{DB}{DB + DA} \quad (3)$$

Keterangan :
DB = *depicts the defect encountered before*
DA = *depicts the defect encountered after delivery*




Gbr. 1 kerangka kerja pengukuran kualitas.

Pengujian pengukuran mengambil proses dan tingkatan yang terjadi selama pengujian. Hakikatnya ada beberapa yang menjadi parameter yaitu jumlah kasus uji yang dirancang (*number of test case designed* - NTCD), jumlah kasus uji dieksekusi (*number of test case execute* - NTCE ), jumlah kasus uji yang berhasil (*number of test case passed* - NTCP), jumlah kasus uji gagal (*number of test case failed* - NTCF ). Data yang digunakan sebagai bahan eksperimen adalah kode sumber program aplikasi penjualan yang dijalankan dan ditulis dengan bahasa pemrograman java.

Dalam literatur uji statistik terdapat 4 kategori pengujian diantaranya adalah *independent sample*, *dependent sample*, *association between* individu, dan *causal relationship*. *Independent sample* dibagi menjadi 2 bagian yaitu 2 sampel dan lebih 2 sampel yang kemudian ada 2 jenis pengujian yaitu parametrik dan non parametrik, begitu pula dengan dependent sample [18], [19]. Kategori berikutnya adalah association between individu pengujian dilakukan dengan uji *chi-square* serta *causal relationship* pengujian menggunakan *univariate regression analysis*. Terkait dengan penelitian yang dilakukan maka akan dilakukan analisis korelasi (*correlation analysis*).

Hakikat analisa korelasi merupakan teknik analisis data dalam statistik yang digunakan untuk mencari hubungan antara dua variabel atau lebih yang bersifat kuantitatif. Analisis korelasi dapat dibedakan menjadi 3 bagian, yaitu (1) *bivariate correlation* yang merupakan korelasi/hubungan antara satu variabel *independend* dengan variabel *dependent*; (2) *partial correlation* merupakan korelasi/hubungan secara parsial atau sendiri-sendiri antara variabel *independent* dengan variabel dependent; (3)*multiple correlation* merupakan korelasi atau hubungan secara bersama-sama lebih dari satu variabel [9]. Analisa korelasi menentukan nilai koefisien korelasi dinotasikan dengan r atau *rho*. Besar nilai koefisien korelasi antara -1 sampai dengan 1. Koefisien korelasi digunakan untuk melihat keeratan antar variabel dan arah hubungan antar variabel. Keakuratan hubungan antar variabel dilihat dari nilai koefisien korelasi (r) semakin mendekati -1 atau 1 maka kekuatan hubungannya semakin besar, seperti pada Tabel I.

TABEL I
NILAI KORELASI R

| Interval nilai korelasi r | Kategori |
|---|---|
| 0,000 ≤ r ≤ 0,200 | Sangat rendah |
| 0,200 ≤ r ≤ 0,400 | Rendah |
| 0,400 ≤ r ≤ 0,600 | Cukup |
| 0,600 ≤ r ≤ 0,800 | Tinggi |
| 0,800 ≤ r ≤ 1,000 | Sangat tinggi |

TABEL II
HIPOTESIS PENELITIAN DAN HASIL

| ID Hipotesis | Isi Hipotesis | Hasil |
|---|---|---|
| **H1** | **H01**: tidak ada hubungan secara signifikan antara kepadatan cacat dengan pengujian pengukuran.<br>**HA1**: ada hubungan secara signifikan antara kepadatan cacat dengan pengujian pengukuran. | Ditolak<br>Diterima |
| **H2** | **H02-1:** tidak ada hubungan secara signifikan antara penghapusan cacat dengan pengujian pengukuran.<br>**H02-2:** ada hubungan secara signifikan antara penghapusan cacat dengan pengujian pengukuran. | Ditolak<br>Diterima |
| **H3** | **H02-1:** tidak ada hubungan secara signifikan antara baris program dengan pengujian pengukuran.<br>**H02-2:** ada hubungan secara signifikan antara baris program dengan pengujian pengukuran. | Ditolak<br>Diterima |

Langkah selanjutnya menentukan hipotesis penelitian dan hasil yang dicapai. Terdapat 3 hipotesis yang diusulkan terkait dengan pengukuran kualitas perangkat lunak agar bebas cacat yang ditampilkan pada Tabel II.

## IV. HASIL DAN PEMBAHASAN

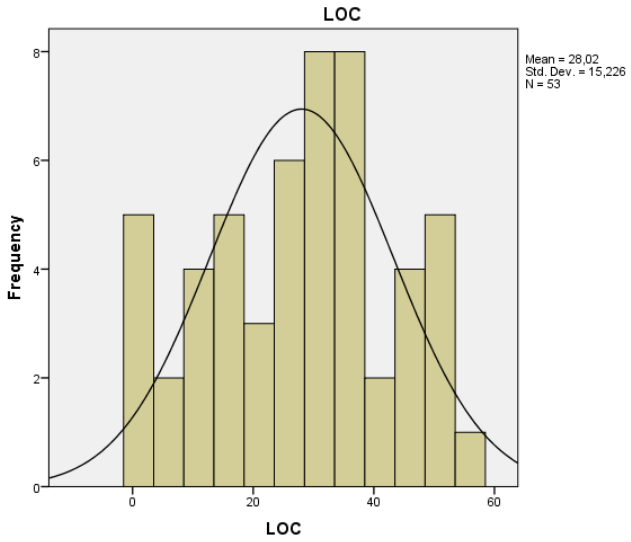
Hasil eksperimen dilakukan dengan menggunakan sebuah laptop Zyrex dan sistem operasi yang digunakan adalah windows 8, sedangkan perangkat lunak untuk pengembangan adalah menggunakan bahasa pemrograman java yang diterapkan pada IDE *(Integrated Development Environment*).

Dalam upaya menganalisis kinerja kerangka model yang diusulkan, aplikasi SPSS versi 20 digunakan dalam membantu olah data.

Kepadatan dihitung berdasarkan persamaan kepadatan cacar (*defect density)*, sedangkan penghapusan cacat didasarkan persamaan DRE ( *Defect Removal Effectivenes*). Nilai pengukuran satuan merupakan hasil penjumlahan semua atribut NTCD, NTCP, NTCE, dan NTCF. Terkait dengan pengujian yang menggunakan pendekatan uji statistik, maka pada penelitian ini, pengolahan data dilakukan dengan analisa data matrik. Analisa data matrik meliputi uji statistik *descriptive*, distribusi data, analisa outlier, dan analisa korelasi. Uji statistik deskriptif meliputi mean,median, mode ,dan sebagainya. Adapun data analisa uji statistik deskriptif ditunjukan pada Tabel III.

TABEL III
PENYEBARAN DATA

| Item | Mean | Median | Mode | Std. Deviation |
|---|---|---|---|---|
| Baris kode | 121,45 | 126,00 | 126 | 68,043 |
| Kepadatan Cacat | 1,54 | 0,535 | 0.40 | 5,071 |
| Penghapusan Cacat | 0,206 | 0,080 | 0,01 | 0,046 |
| Pengukuran Satuan | 107,22 | 100 | 99,09 | 35,17 |




Gbr. 2 Kurva Distribusi Data

Ukuran penyebaran dapat dianalisa dari tabel diatas, rata-rata dan ukuran penyebaran dapat menggambarkan distribusi data tetapi tidak cukup untuk menggambarkan sifat distribusi [19]. Kemiringan berarti ketidaksimetrisan. Sebuah distribusi disebut simetris apabila nilai sebarannya merata disekitar nilai rata-ratanya. Berdasarkan data yang diproses hampir semua variabel memiliki kemiringan ke kanan (positif). Kemudian analisa keruncingan menunjukan memiliki karakteristik kurva normal dari 3 jenis ukuran kurva keruncingan, seperti pada Gbr.2.

*Outlier* atau anomali merupakan himpunan data yang dianggap memiliki sifat yang berbeda dibandingkan dengan data lainnya. Analisis outlier dikenal juga dengan analisis anomali atau deteksi anomali atau deteksi deviasi atau *exception minin* [20]. Ada beberapa pendekatan yang digunakan untuk analisis *outlier* yaitu menggunakan: (1)*blox plots*; (2)*z-scores*, dan (3) *scatter plots* [21]. Blox plots merupakan pendekatan yang menggunakan metode grafis untuk menganalisa apakah data mengalmai *outlier* atau tidak. Di dalam *blox plots* terdapat nilai minimum, quartil terendah (Q1), kuartil pertengahan (Q2), kuartil tertinggi (Q3), nilai maksimum dan nilai interquartile range (IQR). Nama lain dari interquartile range (IQR) adalah simpangan kuartil yang dapat diselesaikan melalui persamaan Q3 – Q1. Adapun untuk menntukan posisi nilai kuartil pada kasus ini adalah sebagai berikut :

$$Q1 = 1 ( n + 1 ) / 4$$
$$Q2 = 2 ( n + 1 ) / 4 \quad (4)$$
$$Q3 = 3 ( n + 1 ) / 4$$

Suatu nilai pada kasus ini dikatakan *oulie*r apabila

$$Q3 + (1.5 \times IQR) < \text{outlier} \leq Q3 + (3 \times IQR) \text{ atau}$$
$$Q1 - (1.5 \times IQR) > \text{outlier} \geq Q1 - (3 \times IQR)$$

Langkah selanjutnya, pada suatu nilai dikatakan ekstrim apabila lebih besar dari Q3 + (3 x IQR) atau lebih kecil dari Q1 – (3 x IQR).

TABEL IV
NILAI QUARTIL RINCI

| | Q1 | Q2 | Q3 | IQR | Outlier |
|---|---|---|---|---|---|
| Loc | 27 | 27 | 34 | 7 | 42 < o < 52 |
| Density | 49 | 17 | 58 | 12 | 76 < o < 94 |
| Removal | 20 | 22 | 38 | 18 | 65 < o < 92 |
| Meas | 18 | 22 | 20 | 2 | 23 < o < 26 |
| Dnumber | 114 | 88 | 272 | 63 | 272 < o < 366 |

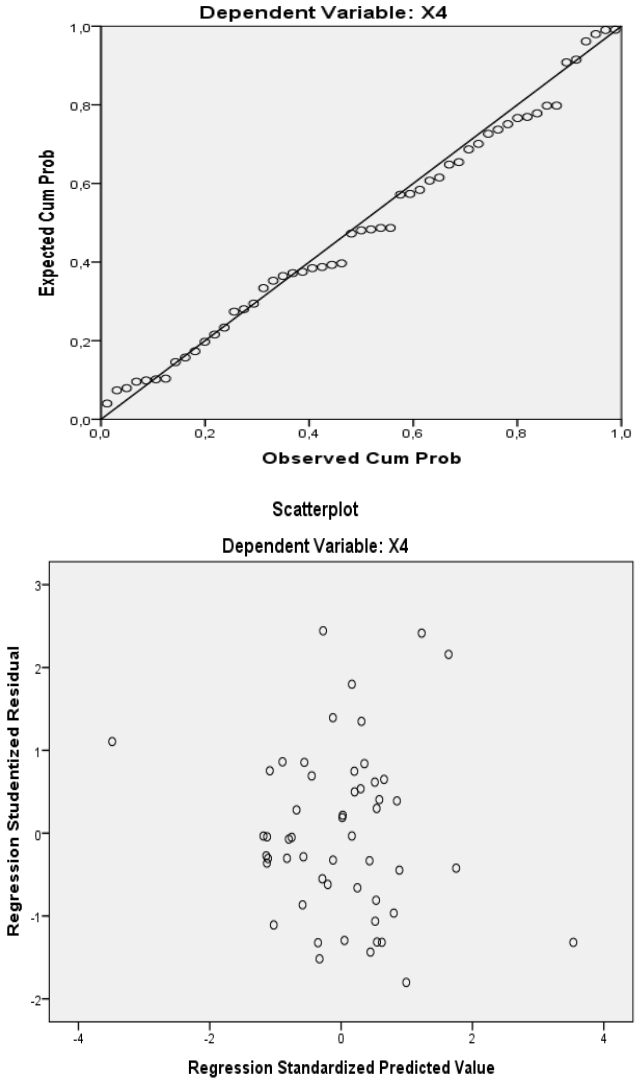
Berdasarkan data yang disajikan pada tabel IV bahwa data tidak mengalami anomali (*outlier*). Pada analisa pendekatan z-score akan dilakukan dengan menggunakan persamaan

$$z = \frac{xi - x}{s} \quad (5)$$

Sebuah nilai disimpulkan *outlier*, apabila nilai Z yang didapat lebih besar dari angka +2,5 atau lebih kecil dari angka -2,5 (-2,5 ≤ Z ≤ +2,5). Adapun data *z-score* ada pada tabel V dengan menunjukan bahwa tidak mengalami *outlier*.

TABEL V
NILAI Z-SCORE OUTLIER

| # | Loc | Density | Removal | Meas | Dnumber |
|---|---|---|---|---|---|
| 1 | -1,773 | 1,084 | -0,584 | -0,883 | -1,101 |
| 2 | -0,613 | 0,216 | -0,990 | -0,731 | -1,140 |
| 3 | -0,162 | -0,825 | -0,245 | 1,165 | -0,157 |
| 4 | 0,546 | 0,968 | -1,125 | -1,187 | -0,235 |
| 5 | -0,098 | 1,142 | 1,244 | -1,414 | 0,748 |
| 6 | 1,448 | 0,853 | -0,245 | -0,125 | 1,299 |
| ... | .... | ... | ... | ... | ... |




Gbr. 3 Hasil Uji Data Outlier

Sedangkan analisa *scatter plots* tidak menunjukan adanya data mengalami anomali. Hal ini dibuktikan dengan semua titik penyebaran terpusat pada garis lurus. Dari ketiga pendekatan ini dapat disimpulkan bahwa data berdistribusi normal. Gbr.3 merupakan salah satu pendekatan dengan *scater plots* yang menunjukan tidak ada data yang mengalami outlier.

Berdasarkan hasil uji statistik pada distribusi frekuensi serta uji normalitas bahwa data dinyatakan berdistribusi normal. Berdasarkan hasil uji korelasi pada pendekatan statistik didapat ada 2 hipotesis yang diterima. Pertama, ada hubungan secara signifikan antara kepadatan cacat dengan pengukuran satuan kualitas dengan signifikasi 0,000 < 0,05. Kedua, ada hubungan secara signifikan antara penghapusan cacat dengan pengukuran satuan kualitas dengan signifikasi 0,003 < 0,05. Hipotesis ke 3 tidak dapat diterima yang menjelaskan tidak ada hubungan signifikan antara baris kode program dengan satuan pengukuran kualitas. Disamping itu penghapusan cacat memiliki nilai korelasi r atau kekuatan yang tinggi terhadap pengkuran kualitas perangkat lunak yang ditunjukan dengan nilai 0,962.

## V. KESIMPULAN

Penelitian ini yang mengkaji pengukuran kualitas perangkat lunak. penelitian telah dilakukan dengan eksperimen untuk menganalisa kekuatan hubungan antara kepadatan cacat, penghapusan cacat dan baris kode program terhadap pengkuran kualitas. Ada 2 aspek yang mempengaruhi dalam pengukuran kualitas perangkat lunak diantaranya kepadatan cacat dan penghapusan cacat yang memiliki nilai yang signifikan. Selain itu, hasil eksperimen yang sudah dilakukan menunjukan bahwa memberikan teknik baru adanya metrik uji untuk dapat memperbaiki metode *white-box* dimana metode tersebut menguji baris kode program secara keseluruhan yang membutuhkan waktu yang lama akibat dari program dengan kompleksitas yang tinggi. Hasil eksperimen ini masih harus dikembangkan kembali yaitu mengantisipasi faktor resiko kesalahan serta kegagalan program melalui logika *fuzzy*. Dengan logika *fuzzy* dapat menyeleksi serta memprioritaskan program dengan ukuran ( *size* )serta kesalahan yang rendah yang berpotensi perangkat lunak dapat mengalami cacat saat dihasilkan.

## DAFTAR PUSTAKA

[1] V. Pham and M. Böhme, 2016, September. Model-Based Whitebox Fuzzing for Program Binaries. *ASE 2016 Proceedings of the 31st ACM International Conference on Automated Software Engineering, 2016. on (* pp. 552–562). ACM.

[2] K. M. R, T. Nadu, and S. G. Jacob, 2016. Improved Random Forest Algorithm for Software Defect Prediction through Data Mining Techniques, *International Journal of Computer Applications*. 117(23), pp. 18–22.

[3] M. Kakkar and M. Kakkar, 2016, January. Feature selection in software defect prediction : A comparative study Feature Selection in Software Defect Prediction : A Comparative Study. *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)* on (pp. 658-663).IEEE.

[4] R. S. Pressman,2010. *Software Engineering;A Practitiner's Approach*. McGrawHill.

[5] C. Nagar and A. Dixit, 2011. Software Efforts and Cost Estimation with a Systematic Approach. *CIS Journal*.. 2(7), pp. 312–316.

[6] Y. Singh, 2012. *Software Testing*. Cambridge Press .

[7] E. A. Felix and S. P. Lee, 2017. "Integrated Approach to Software Defect Prediction," *IEEE Access*, 3536(c) pp. 1–2.

[8] N. Mandhan, D. K. Verma, and S. Kumar, 2015. Analysis of approach for predicting software defect density using static metrics," *Int. Conf. Comput. Commun. Autom.*, pp. 880–886.

[9] V. Chauhan and D. L. Gupta, "Chauhan and Gupta A Comparative Analysis of DIT over MVG to Improve Quality of Software," no. 1, pp. 3–16.

[10] Q. Cao, Q. Sun, Q. Cao, and H. Tan,2015. Software defect prediction via transfer learning based neural network. *2015 First Int. Conf. Reliab. Syst. Eng. (ICRSE),IEEE*, pp. 1–10.

[11] E. Irawan *et al.*,2015. Penggunaan Random Under Sampling untuk Penanganan Ketidakseimbangan Kelas pada Prediksi Cacat Software Berbasis Neural Network, 1(2), pp. 92–100.

[12] R. Malhotra, N. Pritam, and Y. Singh, 2014, .On the applicability of evolutionary computation for software defect prediction, *2014 International Conference on Advances in Computing,Communications and Informatics (ICACCI)*, 2014 pp. 2249–2257, IEEE.

[13] R. Malhotra,2016. *Empirical Research in Software Engineering*. CRC Press.

[14] P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz, 2004, May. An empirical study of software reuse vs. defect-density and stability, *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. pp. 282–291.

[15] S. Kaur, S. Assistant, J. Kaur, S. Faculty, N. Chandigarh, and S. Singh, 2013. Effect of Data Preprocessing on Software Effort Estimation, *International Journal Computation Application*, 69( 25), pp. 975–8887,

[16] S. Kumaresh and R. Baskaran, 2012, April. Experimental design on defect analysis in software process improvement, *International Conference on Recent Advances in Computing and Software Systems*. pp. 293–298.IEEE

[17] G. K. Armah, G. Luo, and K. Qin, 2013, November. Multi_level data pre_processing for software defect prediction, *6th International Conference on Information Management, Innovation Management and Industrial Engineering* . pp. 170–174.IEEE

[18] F. Varanini, G. M. Hill, and W. Curlee, 2015. *Simple Statistical Methods For Software Engineering Projects And Complexity*.McGrawHill.

[19] K. Sobhana, "Software Reliability Growth Model on Burr Type III-An Order Statistics Approach," pp. 57–68.

[20] W. E. Saris and I. N. Gallhofer, 2007.*Design, Evaluation, and Analysis for Questionnaire for Survey Research*.Wiley..

[21] Davis. C,2013. *SPSS for Applied Sciences_ Basic Statistical Testing*.CSIRO Publishing.