KUANTIFIKASI PENGARUH KLONING DAN KOMPLEKSITAS KODE TERHADAP CACAT PADA EVOLUSI PERANGKAT LUNAK

Bayu Priyambadha[1)], Siti Rochimah[2)]
[1)]Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya
[2)]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, ITS
Email : bayu_priyambadha@ub.ac.id[1)], siti@its-sby.edu[2)]

ABSTRAK

Kloning adalah hal yang biasa dilakukan oleh seorang pengembang dalam mengembangkan sebuah perangkat lunak. Kloning dapat menyebabkan menurunnya tingkat perawatan (maintainability) sebuah perangkat lunak. Kloning membutuhkan perhatian yang besar, karena kurangnya perhatian terhadap kloning kode akan menimbulkan sebuah kondisi yang tidak konsisten. Kondisi tidak konsisten dapat menimbulkan cacat perangkat lunak. Selain itu, cacat perangkat lunak dapat ditimbulkan oleh atribut-atribut kode, antara lain adalah kompleksitas kode. Tujuan penelitian ini adalah mencari tahu nilai keterkaitan antara kloning kode, kompleksitas kode, dan LOC (Line of Code) terhadap kemungkinan terjadinya cacat (defect) perangkat lunak. Pencarian hubungan antara kloning, kompleksitas kode, dan LOC dengan cacat dilakukan dengan pendekatan statistika. Regresi dan korelasi adalah metode yang digunakan untuk mencari keterkaitan antara beberapa hal. Penelitian ini menyimpulkan bahwa ketiga atribut kode (kloning, kompleksitas dan LOC) mempengaruhi terjadinya cacat pada perangkat lunak dengan nilai yang besar, yaitu 95%. Masing-masing atribut kode (kloning, kompleksitas dan LOC) memiliki pengaruh yang berbeda-beda. Kloning tidak selalu menjadi pencetus terjadinya cacat yang paling besar.

Kata Kunci: *Kloning kode, kompleksitas kode, LOC, cacat (defect), evolusi perangkat lunak.*

1. PENDAHULUAN

Proses pengembangan perangkat lunak fase implementasi adalah fase realisasi dari fase desain. *Programmer* adalah orang yang bertanggung jawab atas pembuatan kode program. Dalam proses pembuatan kode, programmer sering menggunakan kembali kode program yang sudah pernah dibuat dengan cara melakukan *copy* dan *paste*. Proses penduplikasian kode dengan atau tidak disertai dengan modifikasi disebut sebagai proses kloning [1][2]. Proses kloning kode merupakan hal yang biasa dalam proses pembuatan kode program [2]. Pembahasan yang berkaitan dengan kode kloning telah dilakukan oleh beberapa peneliti. Berbagai macam metode dalam proses pendeteksian kloning pada sebuah kode sumber (*source code*) telah dibahas [1]. Begitu juga perbandingan *tools* atau alat pendeteksian kode kloning juga sudah dilakukan [2].

Kloning dipercaya dapat dapat menurunkan tingkat perawatan (*maintainability*) sebuah software [3]. Hal ini disebabkan oleh adanya perilaku yang tidak konsistensi melakukan perubahan pada kloning kode. Sebagi conoh, kloning dilakukan pada beberapa modul dalam sebuah perangkat lunak, apabila satu dilakukan perubahan maka yang lain juga harus mendapatkan perlakuan yang sama. Perlakuan yang tidak sama dapat menyebabkan kondisi yang tidak konsisten. Kodisi ini dapat menumbuhkan cacat (*defect*) pada perangkat lunak yang dikembangkan [3].

Pada beberapa penelitian diungkapkan bahwa cacat perangkat lunak juga dipengaruhi oleh kompleksitas kode program [4][5]. Kompleksitas kode program diukur dengan beberapa matrik yaitu *LOC (Line of Code)* [4], *McCabe's Cyclomatic Complexity*, dan *Halstead's Volume* [5]. Kompleksitas kode program dijadikan sebagai salah satu acuan dalam melihat kualitas program dan dengan menggunakan metode klasifikasi dapat dijadikan menjadi sebuah model dalam pendeteksian cacat pada perangkat lunak sedini mungkin [5].

Dari kenyataan tersebut, beberapa peneliti berusaha untuk mencari keterkaitan antara kloning, kompleksitas dan cacat [3] atau kerusakan (*decay*) [6] pada perangkat lunak. Dari beberapa penelitian tersebut disimpulkan dua hal yang saling berlawanan. Lozano [6] berpendapat bahwa tidak ada hubungan secara kuat antara kloning (*clone*) dan kompleksitas. Kerusakan perangkat lunak hanya disebabkan oleh kloning. Sebaliknya, Selim [3], memasukkan beberapa matrik kompleksitas sebagai salah satu acuan prediksi cacat pada perangkat lunak. Matrik kompleksitas itu adalah *LOC* dan *Cyclomatic Complexity*. Disini dapat disimpulkan bahwa dalam penelitian yang dilakukan oleh Selim, selain kloning, kompleksitas kode juga diposisikan sebagai suatu hal yang penting yang mempengaruhi cacat. Hal tersebut memunculkan sebuah motivasi untuk melakukan

penelitian lebih dalam pada kaitannya dengan kloning, kompleksitas dan cacat perangkat lunak.

Penelitian ini menerapkan sebuah pendekatan statistika untuk menganalisa kedekatan hubungan antara kloning, kompleksitas dan cacat. Pendekatan statistika diambil sebagai sebuah cara untuk melakukan proses kuantifikasi atas hubungan ketiga hal tersebut, sehingga dapat diukur lebih jelas seberapa besar keterikatan antara ketiga hal tersebut. Penelitian ini menggunakan dataset dari kode sumber perangkat lunak ArgoUML, Ant, jEdit dan jHotDraw. Metode statistika yang digunakan adalah metode regresi dan korelasi dengan prediktor ganda. Metode tersebut digunakan untuk mencari hubungan kausal atau fungsional antara dua variabel [7][8]. Matrik pengukuran kompleksitas kode yang digunakan adalah *LOC* dan *McCabe's Cyclomatic Complexity*.

Penelitian ini dibagi menjadi tujuh (7) bagian, Latar belakang dikemukakan pada bagian pertama. Bagian kedua memaparkan kloning, kompleksitas dan cacat perangkat lunak. Bagian ketiga menjelaskan tentag pendekatan statistika yang digunakan dalam menyelesaikan masalah. Bagian keempat menjelaskan tentang metode yang digunakan dalam penelitian ini. Bagian kelima dan enam menjelaskan implementasi metode dan analisa hasil. Bagian paling akhir menjelaskan kesimpulan dari penelitian ini

2. KLONING, KOMPLEKSITAS DAN CACAT

2.1 Kloning

Kloning kode dapat diartikan sebagai hasil dari proses redundansi [9] atau duplikasi kode dalam pengembangan perangkat lunak tanpa atau dengan disertai perubahan [1]. Adanya kloning pada kode program membutuhkan perhatian yang besar karena kloning di satu sisi dapat menguntungkan tapi di sisi lain dapat membahayakan [2]. Sebagai contoh, apabila sebuah kesalahan ditemukan dalam salah satu fragmen kode dimana kode tersebut telah dikloning pada beberapa bagian file, maka perbaikan juga harus dilakukan pada seluruh fragmen kode yang sama. Beberapa fragmen membutuhkan penyesuaian dengan alur logika dimana fragmen tersebut ditempatkan. Fragmen kode yang tidak mengalami penyesuaian akan menimbulkan kesalahan proses. Fakta yang lain, masih banyak pembahasan tentang tingkat bahaya kode kloning [1]. Di sisi lain, pertanyaan tentang kloning dapat menimbulkan cacat pada perangkat lunak juga masih dalam pertentangan banyak peneliti [1].

Menurut Ira Baxter dalam Mens [9], kloning kode yang sama dapat dinyatakan dengan beberapa pengertian tentang similaritas. Dari pernyataan tersebut kloning kode dapat dibedakan menjadi beberapa tipe yaitu kloning berdasarkan teks, leksikal atau struktur, atau secara kesamaan semantik [9]. Kesamaan kode secara semantik dapat dilihat dari kesamaan perilaku kode program. Mens dan Demeyer [9] membagi kesamaan berdasarkan teks menjadi tiga tipe kloning sebagai berikut ini:

- Tipe 1, kloning tipe ini adalah duplikasi kode tanpa modifikasi (sama persis).
- Tipe 2, kloning dengan persamaan struktur atau secara sintak, nama variabel, tipe data, identifier fungsi yang berbeda.
- Tipe 3, kloning dengan modifikasi lebih dari tipe 2, terdapat perubahan (penambahan atau pengurangan) pada isi fungsi.

2.2 Kompleksitas Kode (*Code Complexity*) dan Cacat Perangkat Lunak (*Software Defect*) Catatan Kejadian

Terdapat banyak acuan dalam menganalisa kualitas perangkat lunak. Kualitas perangkat lunak dapat dilihat berdasarkan karakteristik kodenya. Salah satu acuan adalah dilihat dari kompleksitas kode dari kode sumber (*source code*) perangkat lunak. Terdapat beberapa matrik pengukuran kompleksitas perangkat lunak, yaitu LOC, *McCabe's Cyclomatic Complexity*, dan *Halstead's Volume* [5].

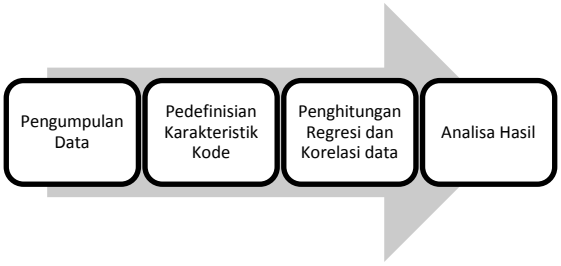
Zhang et al [5] dan Zhang [4] telah membuktikan adanya hubungan antara matrik kompleksitas seperti *LOC, McCabe's Cyclomatic Complexity*, dan *Halstead's Volume* terhadap terjadinya cacat pada perangkat lunak. Cacat perangkat lunak adalah kesalahan langkah, proses dan pendefinisian data pada program komputer atau perangkat lunak [10].

3. REGRESI DAN KORELASI

Hubungan atau korelasi antara dua variabel dapat dihitung dengan metode regresi dan korelasi. Kedua variable tersebut diklasifikasikan menjadi variabel bebas dan variabel terikat. Variabel bebas adalah variabel yang menjadi penyebab akan timbulnya, berubahnya, atau dipengaruhinya sebuah variabel terikat [7]. Dengan kata lain, variabel terikat adalah variabel respon, hasil, atau konsekuensi yang muncul dari adanya variabel bebas [7]. Pada beberapa sumber ditemukan istilah lain untuk variabel bebas dan terikat yaitu variabel independen dan dependen [7]. Penentuan kuat atau lemahnya hubungan antara dua variabel dapat ditentukan dengan melihat nilai dari koefisien korelasi, sedangkan pengaruh variabel bebas (independen) terhadap variabel terikat (dependen) ditentukan dengan melihat nilai koefisien regresi [8].

Persamaan regresi [7] dengan *n* variabel bebas adalah sebagai berikut :

$$Y = a + b_1X_1 + b_2X_2 + \cdots + b_nX_n \qquad [1]$$



Gambar 1. Aliran Proses Pengerjaan Penelitian

dimana *Y* adalah variabel terikat (dependen) yang diprediksi, *X* adalah bebas (independen), *a* dan *b* adalah koefisien regresi. *a* dan *b* dapat dihitung dengan menggunakan beberapa persamaan yang disesuaikan dengan jumlah variabel bebas. Apabila *a* dan *b* dapat dicari, maka untuk menentukan koefisien korelasi antara variabel bebas dan variabel terikat dapat dilakukan dengan menggunakan rumus sebagai berikut ini.

$$R = \sqrt{\frac{b_1 \sum X_1 Y + 2 \sum X_2 Y + \ldots + b_n \sum X_n Y}{\sum Y^2}} \quad [2]$$

Koefisien determinasi adalah hasil penguadratan dari koefisien korelasi, atau dapat direpresentasikan sebagai berikut ini.

$$R^2 = \frac{b_1 \sum X_1 Y + 2 \sum X_2 Y + \ldots + b_n \sum X_n Y}{\sum Y^2} \quad [3]$$

Koefisien determinasi adalah nilai keterkaitan antara variabel bebas terhadap variabel terikat [2][3]. Contoh kasus, hasil perhitungan R^2 didapatkan sebanyak 0.90, maka dapat diartikan bahwa variabel bebas mempengaruhi variabel terikat sebanyak 90% dan 10% dipengaruhi oleh faktor lain.

4. METODOLOGI

Penelitian ini dilakukan dengan beberapa tahapan. Tahapan-tahapan tersebut antara lain pengumpulan data, pendefinisian karakteristik kode, perhitungan regresi dan korelasi data, dan analisa hasil. Tahapan-tahapan tersebut dilakukan secara berurutan karena hasil dari tahapan yang mendahului adalah masukan untuk tahapan berikutnya. Tahapan pengerjaan penelitian dapat dijelaskan pada Gambar 1.

4.1 Pengumpulan Data

Penelitian ini menggunakan data dari project pengembangan perangkat lunak ArgoUML, Ant, jEdit dan jHotDraw. Data tersebut merupakan data kode sumber yang ditulis dengan bahasa pemrograman Java. Menurut indek TIOBE maret 2011 yang diterangkan dalam Tomas et al. [11], bahasa Java menduduki posisi sebagai bahasa yang paling banyak digunakan yang memiliki prosentase sebesar 19,711%. Oleh sebab itu, penelitian ini menggunakan obyek teliti bahasa Java.

Penentuan perangkat lunak obyek teliti ini adalah sebagai awalan untuk melakukan penelitian yang lebih besar. Pada penelitian selanjutnya, obyek teliti sangat mungkin untuk berkembang tidak hanya dari kode sumber yang dibangun dengan bahasa pemrograman Java. Perangkat lunak dengan bahasa pemrograman yang lain juga memungkinkan untuk dijadikan obyek dalam penelitian selanjutnya, seperti C/C++ dan yang lain.

4.2 Pendefinisian Karakteristik Kode

Karakteristik kode adalah beberapa atribut yang melekat pada kode. Pendefinisian karakteristik kode menggunakan perangkat pembantu bernama SonarQube. SonarQube merupakan perangkat lunak berbasis web yang dapat digunakan untuk menjaga kualitas kode program dalam sebuah proses pengembangan perangkat lunak. Variabel yang akan dihitung dalam tahap ini adalah kloning, kompleksitas, dan jumlah kode yang kemungkinan menimbulkan cacat.

Kloning merupakan duplikasi kode yang terdapat pada seluruh file kode sumber perangkat lunak yang diteliti. Perhitungan kloning kode berdasarkan pada kesamaan sintak kode. Selain kloning, variabel yang dipertimbangkan adalah kompleksitas kode. Kompleksitas kode terdiri dari *LOC* dan *Cyclomatic Complexity*. *LOC* adalah jumlah keseluruhan baris kode pada seluruh file kode sumber. *Cyclomatic Complexity* adalah sebuah perhitungan kompleksitas kode berdasarkan dari rumusan yang dirumuskan oleh *McCabe* [5].

4.3 Perhitungan Regresi dan Korelasi Data

Perhitungan regresi dan korelasi data dilakukan dengan menggunakan pendekatan statistik. Perhitungan regresi dan korelasi dilakukan pada data hasil dari tahapan sebelumnya. Data yang akan diproses dalam tahap ini adalah kloning, kompleksitas (*LOC* dan *Complexity*), dan cacat. Kloning, kompleksitas akan menjadi variabel bebas, sedangkan cacat akan menjadi variabel terikat.

Skenario perhitungan regresi dan korelasi antara variabel bebas dan variabel terikat dilakukan dengan melakukan kombinasi antara variabel bebas terhadap variabel terikat. Kombinasi tersebut dijelaskan sebagai berikut:

- LOC terhadap cacat;
- Complexity terhadap cacat;
- Clone terhadap cacat;
- LOC dan Complexity terhadap cacat;
- Complexity dan Clone terhadap cacat;
- LOC dan Clone terhadap cacat; dan
- LOC, Complexity dan Clone terhadap cacat.

Pembuatan kombinasi tersebut digunakan untuk melihat kombinasi dari variabel bebas yang paling

mempengaruhi cacat. Selanjutnya, dari data angka yang didapat dari proses pertama dihitung koefisien determinasi dari masing-masing kombinasi. Lalu, hasil perhitungan dianalisa berdasarkan masing-masing kombinasi.

4.4 Analisa Hasil

Analisa hasil dilakukan dengan melihat hasil perbandingan koefisien determinasi tiap-tiap kombinasi variabel data. Representasi data hasil pemrosesan dan perhitungan akan ditampilkan dalam bentuk tabular dan grafik. Analisa dilakukan dengan melihat data grafik dan tabular yang sudah dibuat, sehingga kemudian dapat ditarik sebuah kesimpulan.

Pembuatan grafik dan tabel dilakukan dengan membedakan obyek teliti. Pada penelitian ini digunakan 4 obyek teliti yaitu ArgoUML, Ant, jEdit dan jHotDraw. Grafik dan tabel dibedakan tiap obyek bermaksud untuk melihat karakteristik tiap-tiap obyek.

5. STUDI KASUS

Studi kasus dalam penelitian ini dilakukan dengan menggunakan data dari repositori kode pengembangan beberapa perangkat lunak. Perangkat lunak yang diteliti adalah ArgoUML, Ant, jEdit dan jHotDraw. Masing-masing perangkat lunak memiliki kode sumber yang berkembang dari versi paling awal hingga versi paling akhir.

ArgoUML memiliki 18 (delapan belas) versi kode sumber. Dengan menggunakan SonarQube didapatkan beberapa nilai dari karakteristik kode seperti dijelaskan pada bagian sebelumnya, yaitu *LOC*, *Complexity*, *Clone* dan *defect*. Hasil perhitungan karakteristik kode sumber ArgoUML dapat dijelaskan pada Gambar 2.a. Ant adalah perangkat lunak yang memiliki versi kode yang paling banyak diantara kode perangkat lunak lain di penelitian ini. Ant memiliki 25 (dua puluh lima) versi yang datanya dijelaskan pada Gambar 2.b. Perangkat lunak ketiga adalah jEdit, dimana jEdit juga memiliki nilai karakteristik kode seperti dua perangkat lunak sebelumnya. jEdit memiliki 23 versi yang karakteristiknya dapat dijelaskan dalam Gambar 2.c. Perangkat lunak keempat adalah jHotDraw. jHotDraw adalah perangkat lunak yang dibangun dengan bahasa Java. jHotDraw memiliki 16 versi untuk dijadikan obyek teliti. Gambaran grafik karakteristik kode jHotDraw dijelaskan pada Gambar 2.d.

Gambar 2 (a-d) menggambarkan karakteristik masing-masing kode sumber dari masing-masing perangkat lunak. Karakter yang dilihat sesuai dengan beberapa variabel yang didefinisikan di sub bab sebelumnya. Dari gambar tersebut, dapat terlihat ritme perkembangan dari keempat variabel pada masing-masing perangkat lunak. Sedikit banyak hampir memiliki bentuk grafik yang sama. Namun, analisa tidak bisa dilakukan hanya dengan melihat bentuk grafik saja. Analisa membutuhkan hasil angka agar hasil analisa dapat dipertanggungjawabkan. Proses selanjutnya adalah dengan menggunakan analisa regresi dan korelasi untuk mendapatkan nilai pasti hubungan antara variabel bebas dan variabel terikat yang didefinisikan.

6. ANALISA HASIL

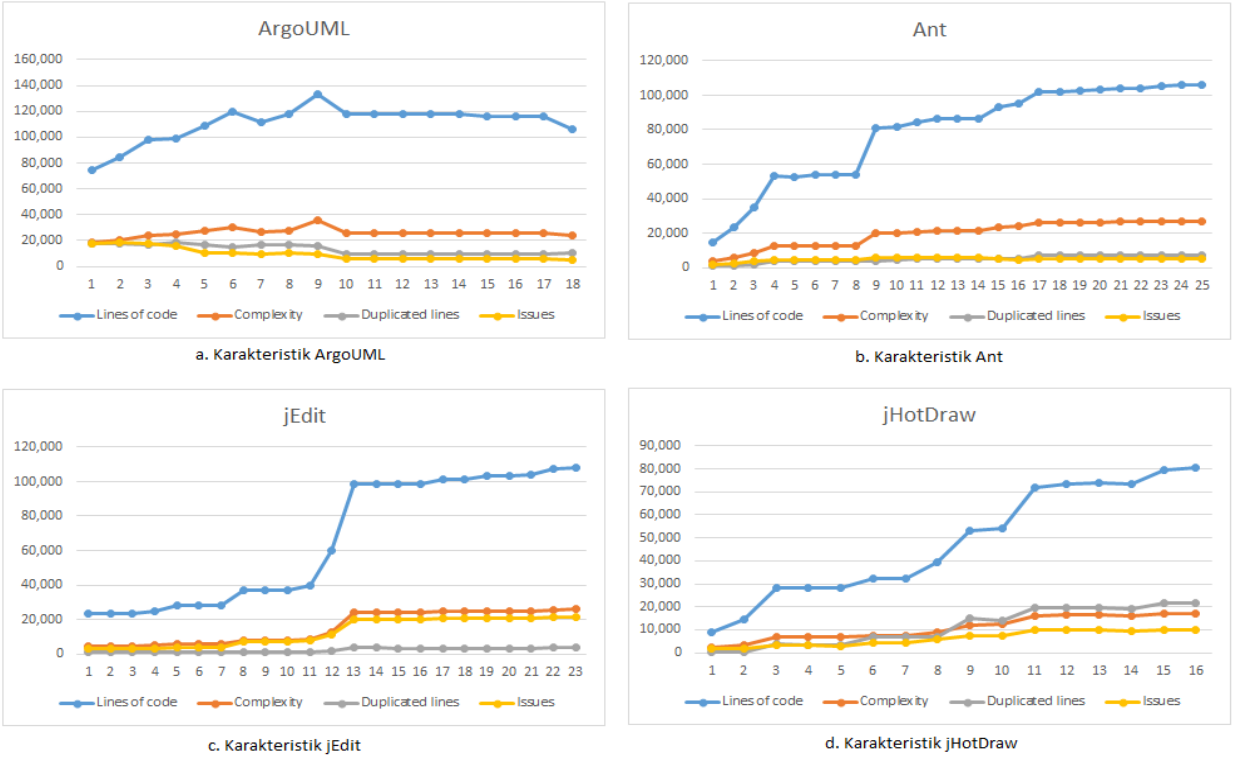
Proses analisa data dilakukan pada hasil perhitungan regresi dan korelasi data. Perhitungan regresi dan korelasi menggunakan sebuah perangkat lunak pembantu untuk perhitungan statistika, seperti SPSS atau PSPP. Hasil dari perhitungan tersebut dijelaskan pada Tabel 1. Nilai R^2 dalam Tabel 1 adalah koefisien determinasi dari tiap-tiap kombinasi variabel bebas terhadap variabel terikat.

Pada kode ArgoUML dijelaskan bahwa hubungan LOC, *Complexity* dan *Clone* terhadap *Defect* adalah 0.90. Nilai 0.90 disini dapat diartikan bahwa LOC, Complexity dan Clone mempengaruhi sebanyak 90% terhadap terjadinya cacat, dimana 10% sisanya dipengaruhi oleh faktor lain. Disamping itu terdapat kombinasi lainnya seperti LOC mempengaruhi sebanyak 57%, *Complexity* mempengaruhi sebanyak 14%, *Clone* mempengauhi sebanyak 76%. Untuk kombinasi yang lain yaitu LOC dan *Complexity* mempengaruhi sebanyak 87%, *Complexity* dan *Clone* mempengaruhi sebanyak 88% dan, LOC dan *Clone* mempengaruhi sebanyak 90% terhadap cacat. Dari kasus ArgoUML didapatkan bahwa kombinasi ketiga variabel bebas memiliki pengaruh yang sangat besar terhadap cacat. Untuk Ant, jEdit dan jHotDraw juga memiliki karakteristik yang sama, dimana pada kombinasi penggabungan ketiga variabel akan memiliki pengaruh terhadap cacat paling besar. Tabel 1 dapat direpresentasikan dalam bentuk grafik yang tergambar pada Gambar 3.

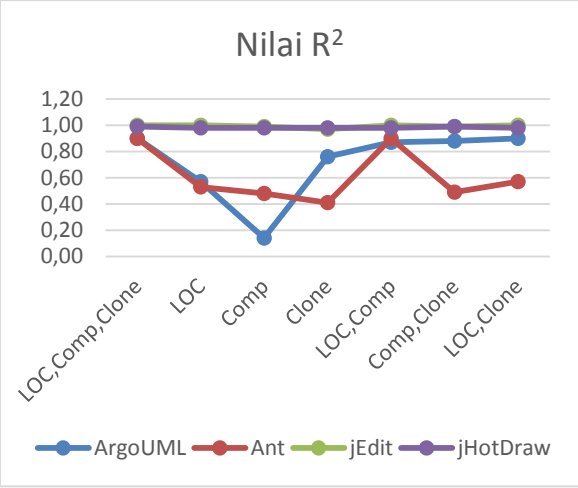
Pada grafik yang tergambar pada Gambar 3 dapat dilihat bahwa pola pengaruh variabel bebas

Tabel 1. Nilai R^2 dari Tiap Variabel Terhadap *Defect*

Variabel	Perangkat Lunak			
	ArgoUML	Ant	jEdit	jHotDraw
LOC,Comp,Clone	0.90	0.90	1.00	0.99
LOC	0.57	0.53	1.00	0.98
Comp	0.14	0.48	0.99	0.98
Clone	0.76	0.41	0.97	0.98
LOC,Comp	0.87	0.90	1.00	0.98
Comp,Clone	0.88	0.49	0.99	0.99
LOC,Clone	0.90	0.57	1.00	0.98



Gambar 2. Karakteristik kode setiap aplikasi



Gambar 2. Grafik perbandingan nilai R^2 tiap kombinasi variabel bebas

terhadap cacat pada masing-masing perangkat lunak. Dari Gambar 3 terlihat bahwa, pola yang ada pada ArgoUML, Ant, jEdit dan jHotDraw berbeda. Kombinasi LOC, *Complexity* dan *Clone* merupakan faktor yang mempengaruhi cacat paling dominan, diikuti dengan kombinasi LOC dan *Complexity* terhadap cacat. Pengaruh LOC, *Complexity* dan *Clone* secara masing-masing terhadap cacat memiliki pola yang berbeda antara perangkat lunak. Pada perangkat lunak ArgoUML, *Clone* memiliki pengaruh paling besar dibandingkan dengan LOC

Tabel 2. Rata-rata nilai pengaruh atribut kode terhadap cacat perangkat lunak

Variabel	Rata-Rata
LOC,Comp,Clone	0.95
LOC	0.77
Comp	0.65
Clone	0.78
LOC,Comp	0.94
Comp,Clone	0.84
LOC,Clone	0.86

dan *Complexity*. Untuk perangkat lunak Ant dan jEdit, pengaruh terbesar ada pada LOC, sedangkan *Clone* menduduki posisi yang paling sedikit pengaruhnya terhadap cacat. Aplikasi jHotDraw me miliki nilai pengaruh masing-masing kombinasi yang rata-rata sama. Hal ini menunjukkan bahwa adanya kloning pada kode perangkat lunak tidak selalu menjadi faktor terbesar yang mempengaruhi cacat perangkat lunak. Dengan kata lain, kloning mempengaruhi terjadinya cacat perangkat lunak, namun kloning tidak selalu menjadi sebab paling besar terhadap terjadinya cacat.

Nilai pengaruh antara atribut-atribut kode tersebut dapat dirata-rata sehingga akan terlihat rata-

rata pengaruh setiap atribut. Nilai rata-rata pengaruh atribut-atribut kode terhadap kemungkinan terjadinya cacat perangkat lunak dijelaskan pada Tabel 2. Dari nilai rata-rata tersebut, gabungan ketiga atribut memiliki pengaruh yang sangat besar terhadap terjadinya cacat perangkat lunak sebesar 0.95.

Hasil perhitungan regresi dan korelasi antara variabel bebas dan terikat dalam fase sebelumnya, menghasilkan nilai yang beragam. Khususnya pada hasil nilai dari masing-masing variabel yang memiliki nilai yang berbeda tiap aplikasi. Hal ini menimbulkan munculnya asumsi tentang kondisi tersebut. Perbedaan nilai pengaruh masing-masing variabel diasumsikan adanya perbedaan jenis dan karakteristik dari tiap variabel tersebut pada masing-masing aplikasi. Jenis dan karakteristik dari masing-masing variabel adalah hal menarik untuk dibahas.

7. KESIMPULAN

Kode sumber pada proses pengembangan perangkat lunak memiliki atribut. Atribut tersebut antara lain adalah kloning, kompleksitas dan LOC. Ketiga atribut tersebut memiliki pengaruh terhadap kemungkinan terjadinya cacat pada perangkat lunak yang dikembangkan.

Nilai pengaruh masing-masing atribut kode tersebut dapat dihitung dengan menggunakan pendekatan statistika. Metode tersebut bernama regresi dan korelasi, dimana metode tersebut dapat mengetahui nilai pengaruh antara variabel bebas (yang mempengaruhi) dengan variabel terikat (yang dipengaruhi).

Nilai perhitungan regresi dan korelasi terhadap atribut-atribut kode menghasilkan sebuah kesimpulan bahwa gabungan antara kloning, kompleksitas dan LOC memiliki pengaruh paling tinggi terhadap terjadinya cacat. Nilai tersebut mencapai rata-rata 0.95 atau dapat diartikan memiliki pengaruh sebesar 95% terhadap terjadinya cacat perangkat lunak. Kloning, kompleksitas dan LOC masing-masing memiliki pengaruh yang beragam terhadap cacat.

Pada empat studi kasus yang digunakan pada penelitian ini, pengaruh masing-masing atribut terlihat memiliki nilai yang beragam. Kloning tidak selalu menjadi pencetus utama cacat yang akan terjadi pada perangkat lunak. Nilai pengaruh kloning terhadap terjadinya cacat pada perangkat lunak dari satu proyek dengan proyek lain memiliki perbedaan. Perbedaan nilai tersebut tidak menunjukan bahwa kloning adalah penyebab tersbesar terjadinya cacat pada perangkat lunak. Pengaruh terbesar selain gabungan antara kloning, kompleksitas dan LOC terhadap cacat adalah gabungan antara LOC dan kompleksitas yang mencapai nilai 0.94 atau memiliki pengaruh sebanyak 94%.

Pada penelitian mendatang, akan dilakukan kajian yang lebih mendalam terhadap fenomena nilai pengaruh masing-masing atribut yang berbeda. Perbedaan tersebut menimbulkan pertanyaan kapan atau pada kondisi apa kloning, kompleksitas dan LOC dapat menjadi pencetus cacat terbesar. Perbedaan nilai pengaruh masing-masing variabel diasumsikan adanya perbedaan karakteristik dari variabel-variabel tersebut pada masing-masing aplikasi.

Penelitian mendatang akan melakukan pendalaman analisa terhadap karakteristik masing-masing variabel secara mendalam. Dataset yang lebih banyak dibutuhkan sehingga nilai pengaruh juga akan lebih terlihat. Selain itu, perluasan obyek yang tidak hanya perangkat lunak yang dibangun dengan bahasa Java dirasa perlu dilakukan untuk melakukan perbandingan karakteristik tiap bahasa pemrograman.

8. DAFTAR PUSTAKA

[1] Rattan, D., Bhatia, R., Singh, M. (2013). "Software Clone Detection : A Systematic Review". Information and Software Technology 55. Hal. 1165 – 1199.

[2] Roy, C.K., Cordy, J.R., Koschke, R. (2009). "Comparison and Evaluation of Code Clone Detection Techniques and Tools : A Quanlitative Approach". Science of Computer Programming 74. Hal. 470 – 495.

[3] Selim, G.M.K., Babour L., Shang, W., Adams, B., Hassan A.E., Zou Y. (2010). Studying the Impact of Clones on Software Defects. 17th Working Conference on Reverse Engineering.

[4] Zhang, H. (2009). "An Investigation of the Relationships Between Lines of Code and Defects". ICSM 2009.

[5] Zhang, H., Zhang, X., Gu, M. (2007). "Predicting Defective Software Components from Code Complexity Measures". 13th IEEE International Sysmposion on Pacific Rim Dependable Computing.

[6] Lozano, A., Wermelinger, M., Nuseibeh, B. (2008). "Evaluating the Relation Between Changeability Decay and the Characteristics of Clones and Methods". 4th International ERCIM Workshop on Software Evolution and Evolvability.

[7] Sugiyono. (2005). "Statistika Untuk Penelitian". Bandung : Alfabeta.

[8] Wirawan, Nata. (2002). "Cara Mudah Memahami Statistika 2 (Statistika Inferensia)". Denpasar : Keraras Emas.

[9] Mens, T., Demeyer, S. (2008). "Software Evolution". Berlin : Springer.

[10] ____. (1990). "IEEE Standard Glossary of Software Engineering Terminology". IEEE std 610.12-1990.

[11] Tomas, P., Escalona, M.J., Mejias, M. (2013). "Open source tools for measuring the Internal

Quality of Java software products. A survey". Computer Standards & Interfaces 36 2013. Hal. 244–255.