LAPORAN PENELITIAN

REVISI INDEKS MODULARITAS UNTUK PENGUKURAN MODULARITAS PROYEK PERANGKAT LUNAK *OPEN SOURCE* DENGAN STUDI KASUS *FREEMIND*

Dr. Andi Wahju Rahardjo Emanuel, BSEE, MSSE (720003)

/ Daniel Jahja Surjawan, S.Kom, M.T. (720212)

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS KRISTEN MARANATHA

BANDUNG

2013

LEMBAR IDENTITAS DAN PENGESAHAN LAPORAN PENELITIAN

1. Judul penelitian	:	Revisi Indeks Modularitas untuk Pengukuran Modularitas Proyek Perangkat Lunak *Open Source* dengan Studi Kasus *Freemind*
2. Ketua Peneliti		
- Nama lengkap	:	Dr. Andi Wahju Rahardjo Emanuel, BSEE, MSSE
- Pangkat / Golongan / NIK	:	720003
- Jabatan Fungsional	:	Lektor / IIID
- Fakultas / Jurusan	:	Teknologi Informasi / Teknik Informatika
- Bidang Keahlian	:	Rekayasa Perangkat Lunak
- Email	:	awreman@gmail.com
3. Anggota Peneliti		
- Nama lengkap dan NIK	:	Daniel Jahja Surjawan, S.Kom, M.T / 720212
4. Luaran yang Ditargetkan	:	Publikasi di Jurnal Internasional IJCA
5. Waktu Penelitian	:	Oktober 2012 – Maret 2013
6. Biaya Penelitian	:	
- Dari UKM	:	Rp. 6.170.000,-
- Dari Institusi Lain	:	Tidak Ada

Bandung, 20 Februari 2013

Menyetujui,
Dekan Fakultas Teknologi Informasi
UK. Maranatha

Ketua Peneliti,

Dr., Ir., Mewati Ayub, MT

Dr. Andi Wahju R.E, BSEE, MSSE

Mengetahui,
Ketua LPPM UK. Maranatha

Prof. Dr. Ir. Benjamin Soenarko, MSME

Daftar Isi

Daftar Gambar

Daftar Tabel

ABSTRACT

Software system has developed significantly to accommodate the increasingly more demanding requirements. The programming languages has also developed from the machine language that depends on the types of hardware into more flexible object-oriented programming language, and eventually the languages must adopt the modularity principle to accommodate the always changing requirements and more complex challenges. Some studies have identified that modularity is one of the key success factors of the object-oriented software project and the main researcher of this research have previously published a software metrics called Modularity Index which is a metrics to measure quantitively the modularity level of java-based Open Source Software Projects. This research is aimed to revise the formulation of Modularity Index based of some observed weaknesses so that the new formulation becomes more effective. The case study to analyze the effectiveness of this revised Modularity Index is the analysis of the evolution of Freemind software project, and the new measure is able to indicate the strengths and weaknesses of this project.

Keywords: *Modularity, Java, package cohesion, package coupling, Open Source, software architecture, Freemind*

ABSTRAK / INTISARI

Sistem perangkat lunak telah berkembang pesat seiring dengan berkembangnya tantangan dan permasalahan yang perlu diakomodasi oleh sistem tersebut. Bahasa pemrograman dalam sistem perangkat lunak telah berkembang dari bahasa mesin yang sangat bergantung pada jenis mesinnya ke bahasa pemrograman berorientasi obyek yang lebih fleksibel, dan kemudian berkembang lagi ke penerapan prinsip modularitas untuk meningkatkan kemampuan sistem tersebut dalam menghadapi perubahaan persyaratan dan tantangan yang lebih kompleks. Beberapa studi telah mengidentifikasikan bahwa modularitas adalah salah satu kunci keberhasilan dari proyek-proyek berorientasi obyek dan peneliti utama dari penelitian ini telah mempublikasikan sebelumnya sebuah metriks perangkat lunak yang dinamakan *Modularity Index* (Indeks Modularitas) yang merupakan sebuah metriks yang dapat mengukur tingkat modularitas suatu proyek perangkat lunak *Open Source* berbasis Java secara kuantitatif. Penelitian ini bertujuan untuk merevisi lebih lanjut formulasi dari Indeks Modularitas ini berdasarkan kelemahan – kelemahan yang ditemukan menjadi suatu formulasi baru Indeks Modularitas yang lebih efektif. Studi kasus yang dipergunakan untuk menganalisa keefektifan dari revisi Indeks Modularitas ini adalah dengan menganalisa evolusi dari proyek perangkat lunak Freemind dan berhasil menunjukkan kekuatan dan kelemahan dari proyek ini.

Kata kunci : Modularitas, *Java*, *package cohesion*, *package coupling*, *Open Source*, arsitektur perangkat lunak, Freemind

I. PENDAHULUAN

Dokumen ini adalah laporan pertanggungjawaban pelaksanaan penelitian Revisi Indeks Modularitas Untuk Pengukuran Modularitas Proyek *Open Source Software* Dengan Studi Kasus Freemind, yang dilaksanakan pada bulan Oktober 2012 – Januari 2013.

I.1. Latar Belakang Penelitian

Saat ini proyek *Open Source Software* (OSS) sudah mendapatkan popularitas di seluruh dunia. Upaya membangun perangkat lunak yang pada awalnya dianggap sebagai bahan eksperimen oleh kalangan akademis dan *hobbyist* sekarang ini telah berkembang menjadi salah satu arus utama metodologi pengembangan perangkat lunak dan bahkan dapat bersaing dengan metodologi rekayasa perangkat lunak yang sudah mapan. Sudah banyak contoh sukses proyek OSS seperti Sistem Operasi Linux, Mozilla Browser, Apache Web Server, dan masih banyak lagi yang setara secara kualitas atau bahkan lebih bagus dari perangkat lunak yang sejenis. Keberhasilan dari proyek-proyek yang ada dipelajari dan salah satu faktor kunci keberhasilan yang dipelajari adalah modularitas dari kode sumber (DeKoenigsberg, 2008).

Modularitas dianggap sebagai salah satu faktor kunci keberhasilan proyek OSS saat berkorelasi dengan kualitas (Stamelos *et al*, 2002). Proyek OSS yang sangat modular dianggap memiliki kualitas yang tinggi, dan pada gilirannya proyek OSS dengan kualitas tinggi juga tergantung pada komunitas yang berkelanjutan (Aberdour, 2007). Sebelumnya peneliti utama telah mengusulkan ukuran kuantitatif modularitas yang disebut sebagai Indeks Modularitas (Emanuel *et al*, 2011). Seiring dengan berjalannya waktu, dipandang perlu untuk merevisi formulasi dari Index Modularitas ini untuk memperbaiki kelemahan – kelemahan yang ditemukan, maka pada penelitian ini diusulkan revisi dari Indeks Modularitas. Sebagai sarana pembuktian dari formulasi baru dari Indeks Modularitas, formulasi baru ini dipergunakan untuk menganalisa evolusi dari salah satu proyek OSS berbasis Java yang populer yaitu FreeMind.

Laporan penelitian ini disusun menjadi empat bagian utama. Bagian pertama menjelaskan penelitian saat ini yang berhubungan dengan perangkat lunak modularitas, modularitas pada proyek OSS, dan tantangan dalam mengukur perangkat lunak. Bagian kedua menjelaskan detail formulasi dari Indeks Modularitas yang telah direvisi. Bagian ketiga menjelaskan

formula baru yang digunakan untuk mengukur dan menganalisis evolusi FreeMind. Bagian terakhir berisi kesimpulan dan saran pengembangan selanjutnya.

I.2. Identifikasi dan Rumusan Masalah

Dari berbagai macam paparan, petunjuk dan fakta yang telah dikemukakan di atas, maka dapat diketahui identifikasi masalah yang perlu dikaji lebih jauh dalam penelitian ini, yaitu:

1. Bagaimana menentukan formulasi baru dari Indeks Modularitas berdasarkan temuan – temuan kelemahan dari formulasi sebelumnya?
2. Bagaimana mengukur keefektifan dari formulasi baru dari Indeks Modularitas dalam menganalisa evolusi proyek OSS?

I.3. Tujuan dan Manfaat Penelitian

Adapun tujuan dari penelitian revisi Indeks Modularitas ini yaitu:

1. Menentukan formulasi baru Indeks Modularitas yang nantinya digunakan untuk mengukur tingkat modularitas dari proyek OSS berbasis *Java*
2. Menganalisis evolusi dari FreeMind untuk menunjukkan efektivitas dalam rangka mendeteksi kekuatan dan kelemahan, sehingga para pemangku kepentingan (komunitas) dari proyek Freemind ini dapat menentukan langkah selanjutnya dari hasil pengukuran yang dihasilkan

I.4. Batasan Penelitian

Batasan – batasan penelitian agar lebih terfokus pada beberapa permasalahan penting yaitu:

- Data untuk analisa adalah data dari 50 proyek perangkat lunak Open Source berbasis Java yang dianggap berhasil yang sebelumnya dipergunakan untuk formulasi Indeks Modularitas versi awal. Proyek – proyek ini merupakan proyek perangkat lunak Open Source berskala kecil sampai menengah (NCLOC $\leq$ 170K) yang diambil dari portal *sourceforge* (http://www.sourceforge.net). Semua proyek perangkat lunak yang dikembangkan secara *Open Source* akan dimulai dari skala kecil, pada fase ini adalah fase paling kritis yang akan menentukan tingkat keberhasilan proyek perangkat lunak tersebut (Lehman, 1996).
- Tambahan data untuk analisa dilakukan terhadap berbagai versi dari proyek perangkat lunak Freemind untuk mengetahui keefektifan dari formulasi baru Indeks Modularitas

dan sekaligus juga mengetahui kekuatan dan kelemahan dari proyek ini jika dilihat dari proses evolusinya.

I.5. Metoda Penelitian

Metodologi penelitian ini dibagi menjadi 5 tahapan yaitu: Pengumpulan Informasi, Pengembangan Formulasi Baru, Analisis, Pengujian, dan Publikasi.

I.5.1. Pengumpulan Informasi

Pengumpulan informasi yang dilakukan akan dititikberatkan pada pengumpulan informasi yang berasal dari jurnal – jurnal dan prosiding Internasional yang berhubungan dengan pengembangan aplikasi berorientasi obyek, *metrics* perangkat lunak, arsitektur perangkat lunak, dan modularitas perangkat lunak.

I.5.2. Pengembangan Formulasi Baru dari Indeks Modularitas

Setelah publikasi awal dari Indeks Modularitas, pada perkembangan selanjutnya telah diketemukan kelemahan – kelemahan dari formulasi metriks perangkat lunak ini dan metriks – metriks pendukungnya. Beberapa kelemahan yang ditemukan antara lain:

1. Pada analisa modularitas dari proyek – proyek perangkat lunak di tingkat *class*, formulasi class quality sebelumnya menggabungkan 3 buah parameter yaitu *LOC Quality* (LOC_Q), *Function Quality* (F_Q), dan LCOM4. Nilai dari LOC_Q dan F_Q adalah nilai ternormalisasi sedangkan nilai dari LCOM4 bukan merupakan nilai ternormalisasi. Maka dipandang perlu untuk memformulasikan nilai baru yang ternormalisasi dari kohesi.
2. Pada analisa modularitas dari proyek – proyek perangkat lunak di tingkat *system*, formulasi dari Indeks Modularitas (M_I) sebelumnya adalah nilai yang selalu meningkat seiring dengan peningkatan jumlah *packages* dari sistem tersebut. Hal ini akan menyulitkan komparasi dari berbagai macam sistem perangkat lunak yang berbeda nilai *packages*-nya. Diperlukan formulasi ulang yang membuat nilai Indeks Modularitas yang ternormalisasi yang memungkinkan komparasi dari sistem – sistem perangkat lunak yang berbeda ukuran dan *packages*-nya.

I.5.3. Analisis

Tahapan – tahapan dari analisi yang dilakukan dalam rangka formulasi ulang dari Indeks Modularitas adalah:

- Analisis ulang secara statistik terhadap 50 proyek – proyek perangkat lunak berbasis *Open Source* dari portal *Sourceforge* yang dianggap berhasil. Proyek – proyek perangkat lunak *Open Source* ini adalah proyek – proyek yang sebelumnya dipergunakan dalam formulasi awal Indeks Modularitas.
- Perumusan ulang *Modularity Index* untuk memperbaiki berbagai kelemahan dari formulasi awal Indeks Modularitas yang sebelumnya telah dipublikasikan. Perangkat yang digunakan untuk mengambil nilai – nilai metrics yang dibutuhkan adalah dengan menggunakan SONAR (http://www.sonarsource.org).

I.5.4. Pengujian

Pengujian baik terhadap sistem yang dikembangkan dan terutama terhadap formulasi baru dari Indeks Modularitas yang telah dirumuskan. Adapun pengujian yang dilakukan adalah dengan melakukan studi kasus evolusi dari proyek *Open Source Freemind* untuk menguji kemampuan dari *Modularity Index* dalam mengindentifikasi kelemahan dan kekuatan dari proyek- proyek tersebut, sekaligus juga untuk mengetahui efektifitas dari formulasi baru dari Indeks Modularitas.

I.5.5. Publikasi

Hasil dari revisi Indeks Modularitas ini kemudian dipublikasikan dalam sebuah jurnal internasional IJCA (*International Journal of Computer Applications*) Volume 59 nomor 12 edisi Desember 2012 dengan judul "*Revised Modularity Index to Measure Modularity of OSS Projects with Case Study of Freemind*". Jurnal Internasional ini memiliki situs web di http://www.ijcaonline.com . Paper hasil publikasi terdapat pada lampiran laporan penelitian ini.

II. LANDASAN TEORI

II.1. Pemrograman Berorientasi Obyek

Bahasa pemrograman berorientasi obyek merupakan pengembangan revolusioner dari proses perkembangan bahasa pemrograman. Bahasa pemrograman berkembang dari bahasa mesin, bahasa *assembler*, bahasa pemrograman prosedural (C, Basic, dan lain - lain), dan pada akhirnya berkembang menjadi bahasa pemrograman berorientasi obyek. Contoh bahasa pemrograman berorientasi obyek yang sekarang masih populer adalah Java dan C#. Bentuk bahasa pemrograman berorientasi obyek ini masih bertahan meskipun sekarang ini sudah mulai populer bentuk bahasa pemrograman berbasis skrip seperti PHP, ASP, dan lain – lainnya.

Pemrograman berorientasi obyek adalah bentuk bahasa pemrograman yang berpusat pada 'obyek'. Obyek (yang dibentuk dari sebuah *class*) ini memiliki karakterisik tertentu (Rosenschein, 2012), yaitu:

- Dapat memiliki sebuah kondisi atau *state* yang dinyatakan dengan nilai –nilai yang dimiliki oleh atributnya.
- Dapat melakukan suatu aksi atau metode (*method*) pada kondisi – kondisi ini.
- Dapat berkomunikasi dengan obyek lainnya dengan cara pertukaran pesan (*message passing*).

Dalam dunia *Open Source*, bahasa pemrograman berorientasi obyek yang paling populer adalah Java. Bahasa pemrograman ini dikembangkan pada tahun 1990an oleh James Gosling dari perusahaan *Sun Microsystem* (http://www.freejavaguide.com/history.html). Perusahaan *Sun Microsystem* sekarang ini telah menjadikan bagian dari perusahaan *Oracle*.

II.2. Proyek *Open Source Software (OSS)*

Open Source merupakan suatu metode pengembangan perangkat lunak yang dilakukan secara bersama – sama oleh banyak orang yang tersebar di berbagai penjuru dunia dimana setiap orang berhak untuk mengunduh, mengembangkan, dan memodifikasi kode sumber untuk kepentingan bersama. Pelopor dari metode ini adalah *Richard Stallman* dalam tulisannya "*Why Software Should be Free*" (Stallman, 1992) dan *Eric Raymond* dengan tulisannya "*The Cathedral and the Bazaar*" (Raymond, 2000) pada awal tahun 1990an. Metode ini kemudian berkembang menjadi suatu gerakan yang menghasilkan aplikasi – aplikasi besar seperti

Apache Web Server, Sistem Operasi *Linux* dengan segala *distro*-nya (Debian, FreeBSD, OpenBSD dan lain – lain), browser web *Firefox* dan lain – lain.

Metode pengembangan perangkat lunak secara *Open Source* memiliki beberapa karakteristik yang berbeda bila dibandingkan dengan metode pengembangan perangkat lunak secara tertutup (*closed source*) atau komersial (*proprietary*). Beberapa karakteristik penting dari metode pengembangan secara *Open Source* adalah:

- Kode sumber (*Source Code*) yang dapat diunduh, dimodifikasi, dan dikembangkan oleh pengguna (Raymond, 2000). Pengguna yang sebelumnya hanya ditempatkan di sisi konsumen sekarang ditempatkan juga di sisi pengembang (*developer*). Para pengembang ini pada akhirnya akan membentuk kelompok yang disebut komunitas, yang pada proyek *Open Source* yang semakin kompleks, komunitas pengembangnya akan terstruktur menjadi beberapa posisi sosial yang telah diidentifikasikan memiliki waktu keanggotaan yang terbatas (Christley & Madey, 2007).
- Pengembang *(developer)* mendaftar atau direkrut secara sukarela. Seiring dengan perkembangannya, baik komunitas dan juga sistem aplikasi *Open Source* mengalami perubahan atau ber-evolusi. Komunitas suatu aplikasi *Open Source* akan terstruktur dengan pusatnya adalah beberapa developer inti (*core developers*) yang dikelilingi oleh banyak pengembang tambahan (*periphery developers*), dan juga lebih banyak lagi pelapor bug (*bug reporter*) yang oleh beberapa studi dikatatakan berbentuk lapisan bawang (Crowston *et al*, 2006). Pengembangan dilakukan secara kolaborasi dengan menggunakan media penghubung yaitu *Internet*.
- Dalam proses pengembangan perangkat lunak secara *Open Source* akan mengikuti beberapa tahapan. Lehman dalam papernya berjudul "*Laws of Software Evolution Revisited*" mengidentifikasikan 8 tahapan penting pengembangan software secara *Open Source* mulai dari tingkat yang paling sederhana sampai yang paling kompleks dimana inisiator proyek pada fase ini sudah tidak mampu lagi untuk mengelola proyeknya sendiri dan membutuhkan orang lain dan komunitas (Lehman, 1996).
- Tidak atau sedikit sekali mempergunakan metodologi formal (Christley & Madey, 2007). Fokus dari pengembangan secara kode terbuka hanya pada dua hal yaitu menambah fitur dan perbaikan terhadap bug yang dilaporkan. Selama proses pengembangan, proyek – proyek perangkat lunak berbasis kode terbuka menggunakan perangkat pendukung yang didesain khusus seperti perangkat pengelola versi secara

bersamaan (*concurrent*) seperti CVS (*Concurrent Versioning System*) dan yang sejenisnya, perangkat untuk melaporkan kesalahan (*bug reporting*) yang jenisnya beragam seperti *bugzilla* di Mozilla, sarana komunikasi antar developer dan pengguna dilakukan dengan menggunakan *mailing – list* dan *forum*, sebuah situs untuk proyek tersebut juga diperlukan sebagai sarana bagi para user untuk mengunduh, menempatkan dokumentasi, pengumuman dan lain – lain. Situs portal yang dikhususkan untuk proyek-proyek *Open Source* sekarang ini cukup banyak misalnya *sourceforge.net*, *freshmeat.net, launchpad.net*, *code.google.com* dan lain – lain.

Di sisi aplikasinya, sistem aplikasi *Open Source* akan berkembang dari suatu sistem yang dikembangkan oleh satu orang saja yang kemudian berkembang menjadi aplikasi yang besar dan kompleks. Suatu studi bahkan menekankan bahwa salah satu kunci utama keberhasilan suatu proyek *Open Source* adalah modularitas (DeKoenigsberg, 2008). Agar suatu proyek *Open Source* agar terus menerus berkembang maka proyek ini harus senantiasa bisa menarik banyak kontributor pemula yang nantinya bakal menjadi kandidat dari pengembang utama (*core developer*). Dengan sifat komunitas yang membolehkan seseorang dating dan pergi kapan saja, maka akan sangat mungkin bagi seorang pengembang (baik yang utama dan tambahan) untuk meninggalkan komunitas karena berbagai sebab (misalnya sudah kurang minat, mengembangkan proyek *Open Source* baru, dan lain – lain), maka kesinambungan dari jumlah orang yang baru harus senantiasa dipelihara agar proyek *Open Source* ini tetap berkembang.

Beberapa studi sebelumnya juga telah mencoba memahami kunci keberhasilan proyek – proyek *Open Source* dengan mencoba memahami proses kerjanya. Beberapa studi mempelajari kasus – kasus proyek *Open Source* besar yang dikategorikan berhasil seperti Debian GNU/Linux (Spaeth & Stuermer, 2007), FreeBSD (Dinh-Trong & Bieman, 2004), Apache (Mockus *et al*, 2000), OpenBSD (Li *et al*, 2005). Beberapa studi telah mencoba mencari keseragaman pola dengan mempelajari lebih dari satu proyek *Open Source* saja seperti Apache dan Mozilla (Mockus *et al*, 2002), 15 Proyek *Open Source* (von Krogh *et al*, 2005), dua proyek *Open Source* (Capiluppi & Ramil, 2004). Sementara studi yang lain mencoba menghubungkan pengembangan proyek *Open Source* dengan praktek rekayasa perangkat lunak modern seperti rekayasa persyaratan (Paech & Reuschenbach, 2006), kesalahan kode (Li *et al*, 2006), pola desain (Hahsler, 2005), model kehandalan (Zhou & Davis, 2005), tahapan pengembangan (Stewart *et al*, 2005) dan praktek kerja (Crowston *et al*, 2004). Beberapa juga telah mempelajari siklus hidup proyek – proyek *Open Source* untuk

mengetahui pola penggunaan kembali (von Krogh *et al*, 2005) (Twindale & Nichols, 2005), dan evolusinya itu sendiri (Alsmadi & Magel, 2006).

Meskipun terdapat beberapa aplikasi *Open Source* yang telah berhasil dan populer, lebih banyak lagi yang tidak berkembang dan gagal yang dikarenakan berbagai macam hal. Beberapa studi telah mengidentifikasikan beberapa kelemahan metode pengembangan ini misalnya jumlah developer yang masih sedikit, pola pengembangan aplikasi yang masih tidak terstruktur (*ad-hoc)*, dan juga dibutuhkan kemampuan pemrograman yang cukup tinggi bagi seseorang untuk dapat berkontribusi pada suatu pengembangan aplikasi *Open Source*. Untuk itu kiat – kiat ataupun strategi – strategi perlu ditemukan dan kemudian diterapkan bagi seorang calon inisiator proyek *Open Source* agar proyeknya dapat berhasil. Studi oleh Stewart juga menunjukkan bahwa apabila suatu perangkat lunak yang dikembangkan secara *Open Source* tidak dikelola secara aktif akan menjadi semakin kompleks (Stewart *et al*, 2005).

II.3. Metrics Perangkat Lunak (*Software Metrics*)

Software Metrics didefinisikan sebagai sebuah nilai yang diekspresikan dalam unit yang berhubungan dengan perangkat lunak (Meyer *et al*, 2009). *Software metrics* berguna untuk mengindikasikan bagaimana keadaan dari sebuah sistem perangkat lunak dan juga memungkinkan perbandingan dan perkiraan tentang apa yang telah dicapai oleh sebuah sistem perangkat lunak. Beberapa kategori *software metrics* yaitu:

- *Metrics* yang berhubungan dengan *size* (ukuran). Contohnya adalah LOC (*Lines of Code*), jumlah kelas atau berkas header, jumlah metode per kelas, jumlah atribut per kelas, ukuran kode yang terkompilasi, dan jejak memori.
- *Metrics* yang berhubungan dengan kualitas / kompleksitas. Contohnya adalah *Cyclomatic Complexity*, jumlah *state*, jumlah *bug* setiap LOC, *Coupling metrics*, dan *Inheritance metrics*.
- *Metrics* yang berhubungan dengan proses. Contohnya *Failed Builds*, *Defect per Hour*, *Requirement Changes*, *Programming Time*, dan *Patches after Release*.

Dalam dunia pengukuran perangkat lunak atau software metrics, terdapat beberapa perintis yang karyanya sangat berpengaruh seperti McCabe dan Chidamber – Kemerer. McCabe dalam papernya yang berjudul "*A Complexity Measure*" memperkenalkan sebuah metrics yang mengukur tingkat kompleksitas dari sebuah program yang kemudian dinamakan

McCabe's Cyclomatic Complexity (McCabe, 1976). Metrics ini merupakan metrics kompleksitas yang paling banyak dipakai oleh banyak peneliti sampai sekarang. *Metrics* untuk perangkat lunak yang berorientasi obyek pertama kali diperkenalkan oleh Chidamber dan Kemerer (Chidamber & Kemerer, 1994). Metrics – metrics tersebut adalah:

1. *Weighted Method Per Class* (WMC): pengukuran kompleksitas dari *class* berdasarkan jumlah methods dalam *class* tersebut.
2. *Depth of Inheritance Tree* (DIT): panjang maksimum dari sebuah *node* ke *root* dalam sebuah pohon pewarisan.
3. *Number of Children* (NOC): jumlah *subclass* langsung dari sebuah *class* dalam suatu hirarki.
4. *Coupling Between Object Classes* (CBO): jumlah dari *class* – *class* lain yang terhubung dengan sebuah class.
5. *Response For a Class* (RFC): jumlah absolut dari *response set* dari sebuah *class*.
6. *Lack of Cohesion in Methods* (LCOM): ukuran ketidaksatuan dari sebuah *class*. Metrics ini kemudian disempurnakan lagi menjadi metrics LCOM2, LCOM3, dan akhirnya versi terbaru adalah LCOM4.

Penelitian di bidang pengukuran perangkat lunak terus berkembang sampai dengan sekarang. Jumlah metrics perangkat lunak terus bertambah untuk menjawab tantangan perkembangan perangkat lunak itu sendiri yang semakin besar dan kompleks dengan segala dinamikanya. Secara keseluruhan sekarang ini terdapat lebih dari 200 *Software Metrics* dengan tujuan pengukuran yang berbeda beda (Meyer *et al*, 2009).

II.4. Modularitas Perangkat Lunak

Pada perangkat lunak, modularisasi dilakukan dengan memecah – mecah sebuah sistem perangkat lunak menjadi bagian yang lebih kecil dan lebih independen yang disebut modul. *Booch* telah mendefinisikan modularitas sebagai properti dari sebuah sistem dimana modul – modulnya kohesif dan *looselycoupled* (Melton & Tempero, 2007). Fenton menyatakan bahwa modularitas adalah atribut kualitas internal dari sebuah sistem perangkat lunak (Melton & Tempero, 2007). Juga diketahui bahwa modularitas berhubungan langsung dengan arsitektur perangkat lunak, karena modularitas adalah pemisahan dari sebuah sistem perangkat lunak ke dalam modul – modul yang independen dan berkolaborasi yang dapat diorganisir dalam sebuah arsitektur perangkat lunak (Nakagawa *et al*, 2008). Modularitas memiliki beberapa

keuntungan dalam hal pemeliharaan (*maintainability*), pengaturan (*manageability*), dan pemahaman (*comprehensibility*) (Munelly *et al*, 2007).

Terdapat empat atribut yang berhubungan erat dengan modularitas dalam sistem perangkat lunak yaitu keterkaitan (*coupling / dependency*), kompleksitas (*complexity*), kohesi (*cohesion*), dan penyembunyian informasi (*information hiding*). Atribut pertama adalah keterkaitan (*coupling / dependency*) yang terdiri dari keterkaitan sintaktik / langsung yang bisa didapatkan dengan komposisi, penandaan metode (*method signatures*), instantiasi kelas, dan pewarisan; dan semantik atau keterkaitan secara tidak langsung (Matos *et al*, 2007). Atribut kedua adalah kompleksitas yang dapat diukur dengan menggunakan *software metrics* seperti *McCabe's Cyclomatic Complexity*, *Halstead's Software Metrics*, *Function Points*, atau ukuran fisik seperti LOC atau SLOC (Wang & Shao, 2003). Atribut ketiga adalah kohesi yang mengukur integritas dari kode sumber di dalam setiap modul. Terminologi yang dipergunakan untuk mengukur secara kualitatif dari kohesi adalah kohesi tinggi (*high cohesion*) atau kohesi rendah (*low cohesion*) (Lee *et al*, 2007). Atribut terakhir adalah penyembunyian informasi (*information hiding*) yang meliputi penyembunyian detil dari implementasi dari modul eksternal (Capra *et al*, 2008).

Untuk merumuskan tingkat modularitas secara kuantitatif, pendekatan awal yang dilakukan adalah dengan mengidentifikasikan ukuran kualitatif dari modularitas itu sendiri. Atribut – atribut ini kemudian digunakan sebagai tolok ukur terhadap teknik – teknik yang tersedia untuk mencapai modularitas. Teknik – teknik yang sesuai untuk proyek perangkat lunak berbasis *Open Source* yang diinginkan. Untuk mendapatkan sebuah sistem perangkat lunak yang modular secara ideal, sistem tersebut harus memiliki atribut – atribut berikut ini:

- Keterkaitan (*coupling / dependency*) yang rendah (Huynh & Cai, 2007): peminimalan atau standardisasi dari coupling / dependency sebagai contoh melalui format standar seperti API – API yang terpublikasi, penghapusan dari keterkaitan semantik, dll.
- Kompleksitas yang rendah: hirarki dari modul – modul yang lebih condong ke keterkaitan yang datar / horisontal daripada yang tinggi / vertikal (Melton & Tempero, 2007)(Aruna *et al*, 2008).
- Kohesi yang tinggi: integritas yang tinggi dari struktur internal dari modul modul perangkat lunak yang biasanya dinyatakan dalam bentuk kohesi yang tinggi atau kohesi yang rendah, dan kohesi yang tinggi mengindikasikan integritas yang tinggi dari suatu *class* (Lee *et al*, 2007).

- Terbuka untuk ekstensi (Huynh & Cai, 2007): kemampuan dari modul – modul yang sudah ada untuk dikembangkan sehingga menjadi modul yang lebih kompleks.
- Tertutup untuk modifikasi (Huynh & Cai, 2007): menghindari perubahan kode – kode yang sudah di-*debug*. Pembuatan dari modul – modul baru harus disarankan menggunakan ekstensi dan tidak memodifikasi modul – modul yang sudah dites.

Terdapat banyak *Software Metrics* yang dapat dikumpulkan dengan menggunakan perangkat SONAR, namun hanya beberapa dari *metrics* tersebut yang dipilih karena *metrics – metrics* tersebut sudah diidentifikasikan sebagai indikator tingkat modularitas dari proyek perangkat lunak berorientasi obyek.

II.5. SONAR

Sumber: http://www.sonarsource.org

SONAR adalah sebuah platform terbuka untuk mengelola kualitas dari kode perangkat lunak berbasis web (dipasang pada server web berbasis Apache, PHP dan MySQL). Perangkat SONAR ini mampu menganalisis berbagai macam proyek perangkat lunak yang dibuat dengan bahasa pemrograman Java, C, C#, Flex, Natural, PHP, PL/SQL, Cobol dan Visual Basic 6. Fleksibilitas dan kemampuan SONAR didasarkan pada plugin – plugin yang ditambahkan ke dalam perangkat ini yang bisa mengukur berbagai macam metrics dan parameter – parameter lainnya. Adapun plugin – plugin penting yang terdapat dalam SONAR adalah:

- *checkstyle*: perangkat untuk meneliti pelanggaran terhadap standar koding bahasa pemrograman Java yang telah ditetapkan
- *cobertura*: perangkat untuk menganalisis jangkauan kode (code coverage) di bahasa pemrograman Java.
- *cpd:* perangkat pendeteksi *copy* dan *paste* pada kode sumber.
- *pmd*: perangkat untuk menganalisis bug potensial, kode mati, kode duplikasi dan tidak optimal.
- *squid*: perangkat untuk mengukur tingkat kohesi pada class
- *surefire*: perangkat untuk melakukan unit testing.
- dll (total sekitar 50 plugin yang tersedia).

III. FORMULASI BARU INDEKS MODULARITAS

Formulasi dari revisi Indeks Modularitas dimulai dari tingkat *Class*, kemudian berkembang ke tingkat *Package* (modul) dan akhirnya ke tingkat *System*. Indeks modularitas ini dimaksudkan untuk menentukan tingkat modularitas pada tingkat sistem secara keseluruhan.

III.1. Modularitas Tingkat *Class*

Ada tiga komponen utama yang menjadi bagian dalam modularitas tingkat *Class* yaitu *LOC Quality (LOC_Q), Function Quality (F_Q), dan Cohesion Quality (H_Q).* Dua komponen pertama (LOC_Q dan F_Q) sudah dibahas sebelumnya dan komponen ketiga (H_Q) adalah pengukuran yang baru. LOC_Q adalah nilai normal yang menentukan kualitas sebuah *Class* berdasarkan jumlah *Non-Commenting Lines of Code (NCLOC)* pada *Class*. Nilai ini berdasar pada pengamatan terpilih 50 proyek OSS berbasis *Java* (Emanuel *et al*, 2011) dan formulasi tersebut ditampilkan seperti berikut:

$$LOC_Q = 0.0138\, NCLOC + 0.310 \; for \; NCLOC \leq 50 \quad \ldots(1)$$

$$LOC_Q = \frac{1}{(NCLOC - 50)^{1.969}} \; for \; NCLOC > 50 \quad \ldots(2)$$

Dimana:

LOC_Q = *LOC Quality*

NCLOC = *Non-Commenting Lines of Code*

Demikian pula F_Q adalah nilai normal yang menentukan kualitas sebuah *Class* berdasarkan jumlah *Function* (metode) di *Class*. Nilai ini juga didasarkan pada pengamatan terpilih 50 proyek OSS berbasis *Java* (Emanuel *et al*, 2011).

$$F_Q = 0.1836\, F + 0.0820 \; for \; F \leq 5 \quad \ldots(3)$$

$$F_Q = \frac{1}{(F - 4.83)^{2.691}} \; for \; F > 5 \quad \ldots(4)$$

Dimana:

F_Q = *Function Quality*

F = *Function / method*

Komponen ketiga adalah *Cohesion Quality* (H_Q) yang merupakan nilai normal yang menentukan kualitas dari sebuah *Class* berdasarkan nilai kohesi (LCOM4). *Cohesion Quality* menggantikan nilai LCOM4 yang digunakan pada formulasi kualitas *Class* (Emanuel *et al*, 2011). Dari 50 terpilih proyek OSS berbasis *Java* dan menggunakan *polynomial square fit* terbalik, formulasi *Cohesion Quality* ditunjukkan sebagai berikut:

$$H_Q = \frac{1}{LCOM4^{2.216}} \qquad \dots (5)$$

Dimana:

H_Q = *Cohesion Quality*

LCOM4 = *Lack of Cohesion Metrics 4*

Akhirnya, formulasi c_Q terdiri dari tiga komponen dengan bobot masing-masing komponen tersebut dibedakan berdasarkan pada kasus bahwa NCLOC dan F yang memiliki korelasi yang tinggi, dan LCOM4 tidak bergantung karena memiliki korelasi yang rendah dengan NCLOC dan F .

$$c_Q = 0.25\ LOC_Q + 0.25\ F_Q + \ 0.5\ H_Q \qquad \dots (6)$$

Dimana:

c_Q = *class Quality*

LOC_Q = *LOC Quality*

F_Q = *Function Quality*

H_Q = *Cohesion Quality*

Class Quality adalah nilai normal yang menentukan kualitas suatu *Class* tertentu. Nilai maksimum *class Quality* dicapai ketika kelas memiliki 50 NCLOC, 5 Function (metode) dan kohesi yang sempurna (LCOM4 setara dengan 1).

Tabel 1 menunjukkan ringkasan perbedaan antara versi awal dan revisi dari modularitas tingkat *class*.

Tabel 1. Perbandingan tingkat *class*

Parameter	**Versi Awal**	**Versi Revisi**
c_Q	Tidak Berubah	

Parameter	Versi Awal	Versi Revisi
F_Q	Tidak Berubah	
H_Q	Tidak Digunakan	Menggantikan LCOM4
c_Q	Gabungan dari LOC_Q, F_Q dan LCOM4	Gabungan dari LOC_Q, F_Q dan H_Q seperti ditunjukkan formulasi 6

III.2. Modularitas Tingkat *Package*

Tidak ada perbedaan untuk formulasi *Package Quality*. *Package Quality* adalah rata-rata kualitas *class* dalam *package* itu:

$$P_Q = \frac{\sum_{i=1}^{j} c_{Qi}}{\sum_{i=1}^{j} c_i} \quad \ldots (7)$$

Dimana:

P_Q = *Package Quality*

c_{Qi} = *i-th class Quality*

c_i = *i-th class*

Meskipun tidak ada pengamatan yang serupa dengan jumlah *class* dalam tiap *package*, namun hal ini dapat diamati dari 50 proyek OSS terpilih dimana proyek tersebut memiliki 10 sampai 16 *class* per *package*.

III.3. Modularitas Tingkat *System*

Komponen untuk modularitas tingkat *system* adalah *System Architecture* (S_A) dan *Package Quality* (P_Q). Nilai S_A menentukan kualitas dari arsitektur sistem, nilai ini ditentukan oleh dua parameter yaitu *Package Coupling* dan *Package Cohesion*. *Package Coupling* (Cij) didefinisikan sebagai jumlah ketergantungan dari kelas dalam satu paket untuk kelas dalam paket lain dalam sistem. Sedangkan *Package Cohesion* (Cii) didefinisikan sebagai jumlah ketergantungan dari kelas dalam satu paket untuk kelas dalam paket termasuk dependensi ke dalam kelas itu sendiri. Formulasi S_A didasarkan pada rumusan nilai yang sama tetapi menggunakan langkah-langkah entropi (Ammar *et al*, 2008).

$$S_A = \sqrt{\frac{\sum_{i=1}^{d} c_{ii}^2}{\sum_{i=1}^{d}\sum_{j=1}^{d} c_{ij}^2}} \quad \ldots (8)$$

Dimana:

S_A = *System Architecture*

C_{ii} = *Package Cohesion*

C_{ij} = *Package Coupling* (jika $i \neq j$)

Nilai S_A adalah nilai normal dari 0 sampai 1. Nilai 0 berarti bahwa sistem tidak memiliki *Package Cohesion*, sedangkan nilai maksimum S_A tercapai saat memiliki banyak *Package Cohesion* dan tidak ada *Package Coupling*.

III.4. Formulasi Indeks Modularitas

Indeks modularitas didefinisikan sebagai produk dari System Architecture dan rata-rata dari *Package Quality*.

$$M_I = S_A \frac{\sum_{i=1}^{j} P_{Qi}}{\sum_{i=1}^{j} P} \qquad \ldots(9)$$

Dimana:

M_I = *Modularity Index*

S_A = *System Architecture*

P_{Qi} = *i-th Package Quality*

P_i = *i-th Package*

Indeks modularitas adalah nilai normal dengan nilai antara 0 sampai 1. Nilai maksimum M_I tercapai ketika S_A dan rata-rata P_Q adalah sama dengan1. Rata-rata P_Q sama dengan 1 dapat dicapai ketika semua *Package Quality* dalam sistem juga sama dengan 1.

Formulasi dari Indeks Modularitas ini berbeda dari rumusan sebelumnya dimana nilai M_I selalu meningkat saat jumlah *package* meningkat (Emanuel *et al*, 2011). Formulasi yang baru seharusnya dapat meningkatkan pemahaman dari perangkat lunak metrik dan juga memungkinkan untuk membandingkan Indeks Modularitas proyek OSS berbasis *Java* dengan ukuran yang berbeda (*class* dan *package*).

Tabel 2 di bawah ini menunjukkan perbandingan antara versi awal dan versi revisi dari modularitas tingkat *system*.

Tabel 2. Perbandingan Tingkat *Class*

Parameter	Versi Awal	Versi Revisi
Package Coupling	Tidak Berubah	
Package Cohesion	Tidak Berubah	
S_A	Tidak Berubah	
M_I	Nilai selalu meningkat (tidak dibagi dengan jumlah paket)	Nilai ternormalisasi seperti ditunjukkan persamaan 9

IV. STUDI KASUS PADA PROYEK FREEMIND

Untuk menunjukkan efektivitas dari revisi Indeks Modularitas, pengukuran dan komponennya digunakan untuk menganalisis proses evolusi dari proyek OSS berbasis *Java* yang memiliki jumlah *download* yang cukup banyak di portal *sourceforge.net* yaitu perangkat lunak pemetaan *FreeMind*. Analisis ini didasarkan pada pengukuran metrik perangkat lunak yang menunjukkan modularitas dari *NCLOC, package, class*, rata-rata *class per package*, *function*, rata-rata *function per class*, rata-rata *NCLOC per class,* $S_{A,}$ rata-rata $P_{Q,}$ dan $M_{I.}$

FreeMind adalah proyek OSS berbasis Java yang digunakan untuk menggambar *mind mapping* yang merupakan cara alternatif untuk melakukan *brainstorming* dengan menggunakan representasi grafis dalam dokumen kertas atau digital, yang dipopulerkan oleh Tony Buzan. Versi terbaru dari FreeMind adalah versi 0.9 dengan banyak ketidakstabilan sebelum versi 1.0.0 sebagai tonggak versi yang belum tercapai. Studi tentang evolusi FreeMind seharusnya memberikan beberapa wawasan tentang proses evolusi sebelum proyek OSS mencapai versi 1.0.0 yang merupakan tonggak penting pertama proyek.

Proses pengumpulan data menggunakan SONAR (http://www.sonarsource.org) dimana aplikasi ini sangat cocok untuk mengumpulkan beberapa metrik perangkat lunak pada proyek berbasis Java. Pada versi tertentu dari proyek yang akan dianalisis oleh SONAR, kode sumber harus dapat dikompilasi menggunakan ANT dimana pengukurannya dapat dipanggil dengan menggunakan kode *build.xml*. Terdapat 22 versi yang dikumpulkan dan disusun mulai dari versi 0.4 pada tanggal 7 Juli 2001 sampai versi 1.0.0 Beta 8 pada tanggal 7 Oktober 2012. Tabel 3 menunjukkan detail dari versi FreeMind yang dikumpulkan dan diukur.

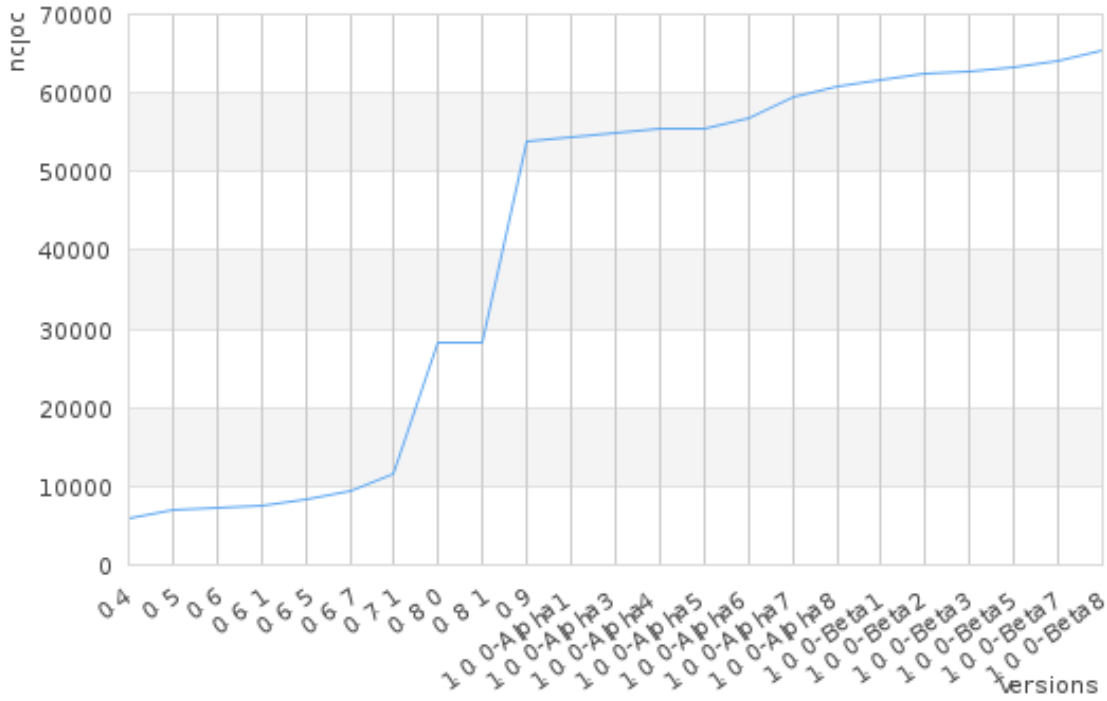
Tabel 3. Daftar Proyek Freemind

No.	Proyek	Versi	Tanggal rilis
1	FreeMind	0.4	7 Juli 2001
2	FreeMind	0.5	24 Agustus 2002
3	FreeMind	0.6	1 Februari 2003
4	FreeMind	0.6.1	8 Februari 2003

5	FreeMind	0.6.5	4 September 2003
6	FreeMind	0.6.7	25 Oktober 2003
7	FreeMind	0.7.1	21 Maret 2005
8	FreeMind	0.8.0	7 September 2005
9	FreeMind	0.8.1	26 Februari 2008
10	FreeMind	0.9	18 Februari 2011
11	FreeMind	1.0.0 Alpha 1	26 Maret 2011
12	FreeMind	1.0.0 Alpha 3	14 April 2011
13	FreeMind	1.0.0 Alpha 4	21 Mei 2011
14	FreeMind	1.0.0 Alpha 5	26 Juni 2011
15	FreeMind	1.0.0 Alpha 6	30 September 2011
16	FreeMind	1.0.0 Alpha 7	8 November 2011
17	FreeMind	1.0.0 Alpha 8	18 Desember 2011
18	FreeMind	1.0.0 Beta 1	17 Februari 2012
19	FreeMind	1.0.0 Beta 2	29 April 2012
20	FreeMind	1.0.0 Beta 3	9 Mei 2012
21	FreeMind	1.0.0 Beta 5	10 Juni 2012
22	FreeMind	1.0.0 Beta 7	5 September 2012
23	FreeMind	1.0.0 Beta 8	7 Oktober 2012

IV.1. Evolusi NCLOC

Metrik perangkat lunak yang digunakan untuk menunjukkan ukuran proyek adalah *Non-Commenting Lines of Codes* (NCLOC). Gambar 1 menunjukkan evolusi NCLOC dari 23 versi FreeMind.

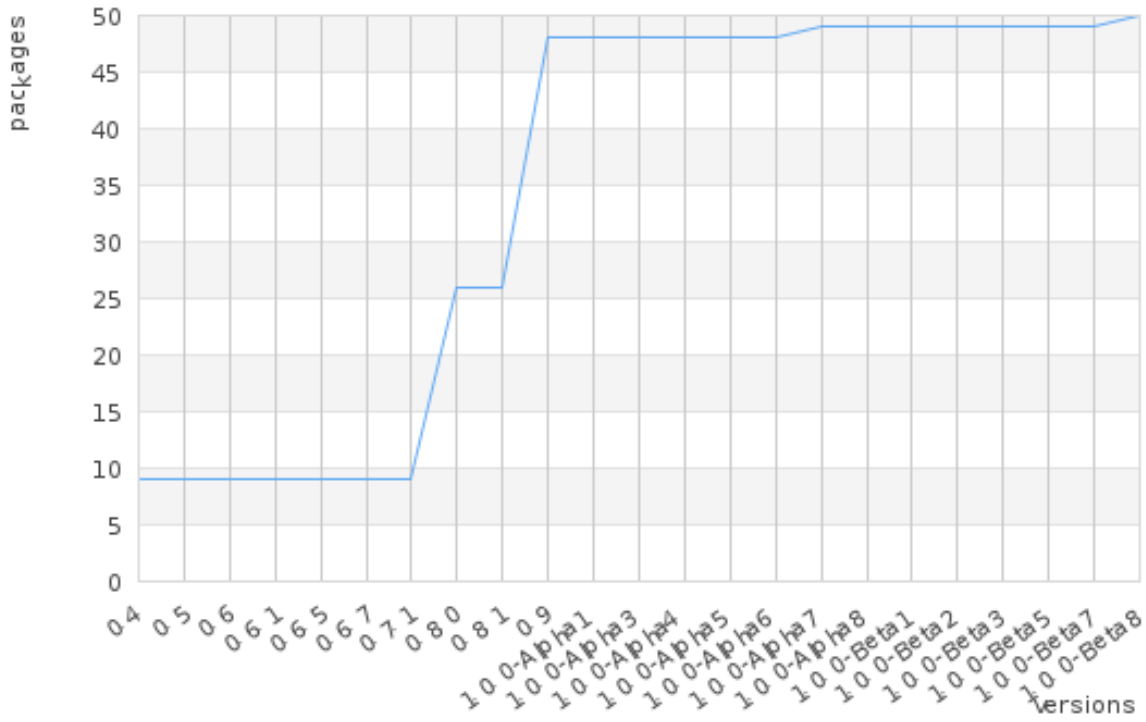


Gambar 1. Evolusi NCLOC Freemind

Gambar di atas menunjukkan bahwa NCLOC dari FreeMind mengalami pertumbuhan 11 kali lipat dari 5835 pada versi 0,4 sampai 65.408 pada versi 1.0.0 Beta 8. Pada versi 0.8.0 ke 0.8.1 tidak mengalami pertumbuhan (metrik lain juga menunjukkan nilai yang sama) menunjukkan masa vakum dari proyek selama sekitar 2,5 tahun. Pertumbuhan NCLOC pada FreeMind versi beta mendekati 65K yang menunjukkan peningkatan kematangan dari kode sumber.

IV.2. Evolusi *Packages*

Indeks modularitas mendefinisikan 'modul' pada proyek OSS berbasis *Java* yaitu *Package*. Gambar 2 menunjukkan evolusi *Package* di FreeMind.

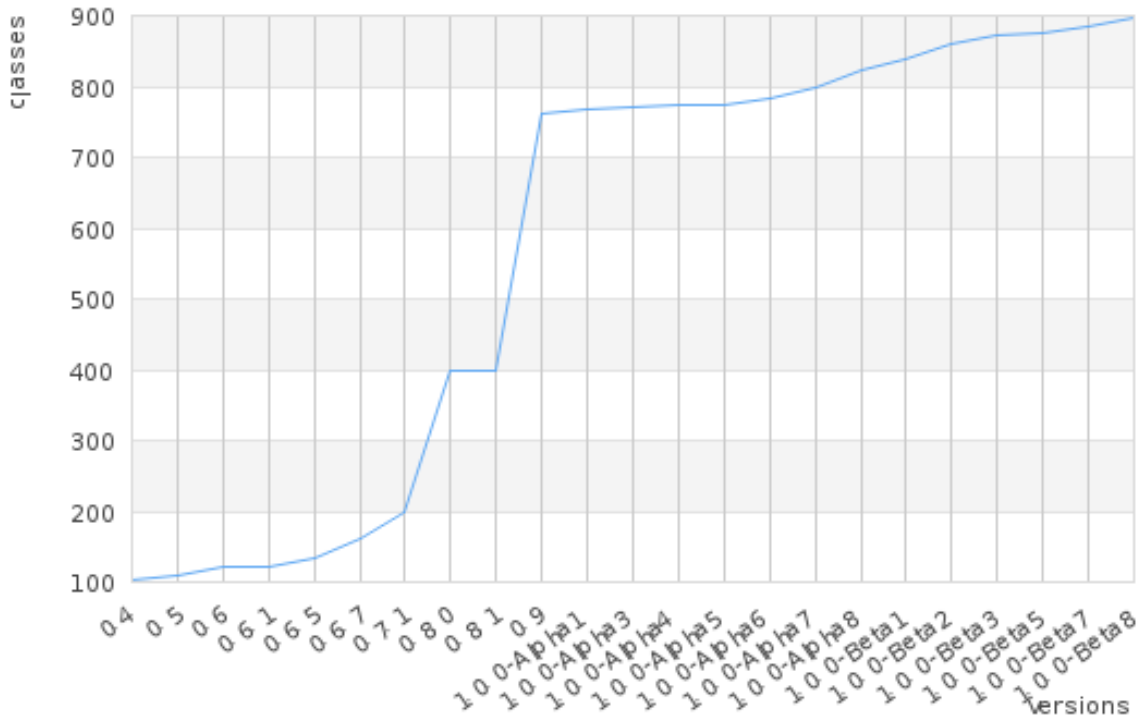


Gambar 2. Evolusi *Package* Freemind

Jumlah *package* telah berkembang dari 9 pada versi 0,4 menjadi 50 pada versi 1.0.0 Beta 8. Hal ini menunjukkan peningkatan fitur yang signifikan dari proyek. Pertumbuhan *package* pada versi beta menunjukkan kestabilan pada jumlah modul yang menunjukkan juga stabilitas sejumlah fitur dalam proyek.

IV.3. Evolusi *Class*

Gambar 3 di bawah ini menunjukkan evolusi *class* pada FreeMind.

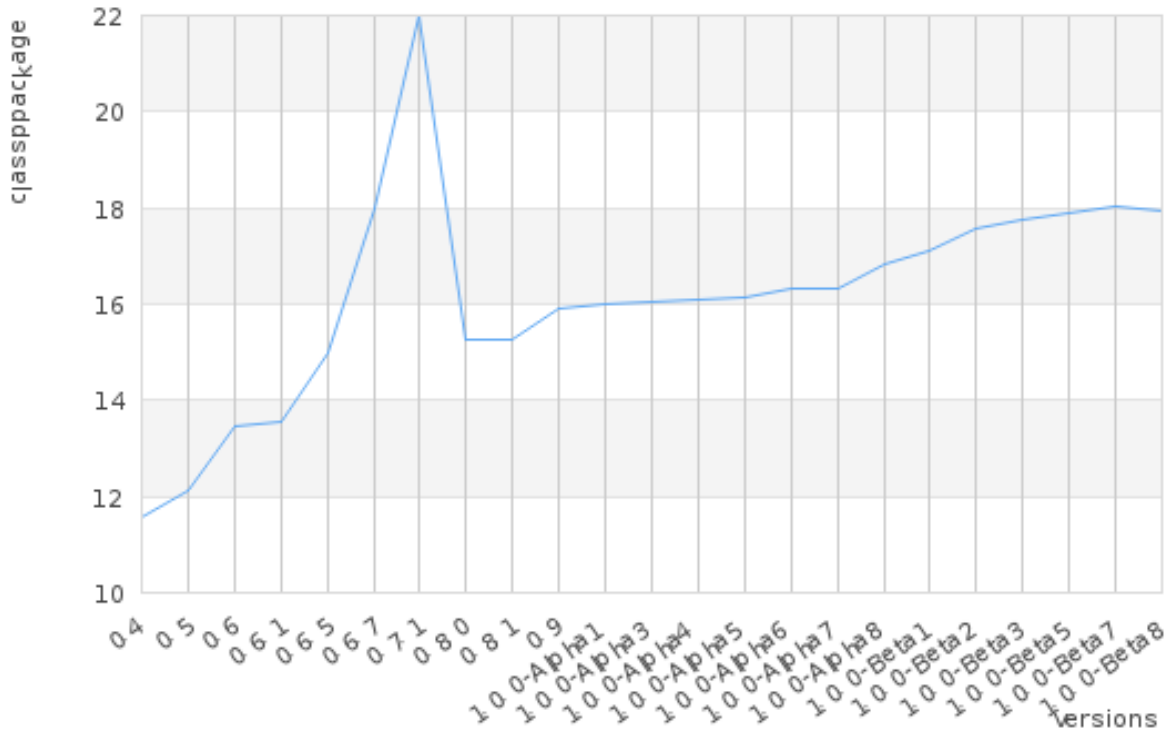


Gambar 3. Evolusi *Class* Freemind

Jumlah *class* telah meningkat dari 104 pada versi 0,4 menjadi 898 pada versi 1.0.0 Beta 8. Jumlah *class* juga menunjukkan titik stabilitas di versi beta dari FreeMind yang menunjukkan stabilitas kode sumber.

IV.4. Evolusi Rata-rata *Class* per *Package*

Gambar 4 di bawah ini menunjukkan evolusi dari rata-rata *class* per *package* di FreeMind.

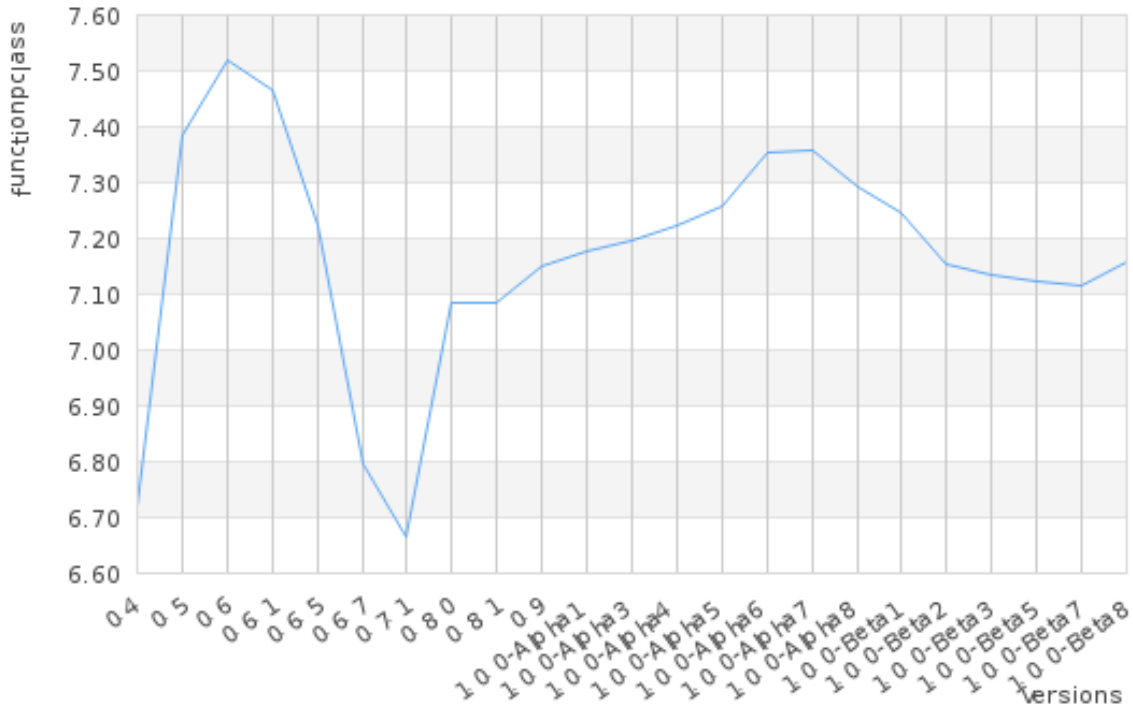


Gambar 4. Evolusi Rata-rata *Class* per *Package* Freemind

Rata-rata *class* per *package* menunjukkan hasil yang menarik dimana nilainya meningkat secara signifikan dari versi 0.4 ke versi 0.7.1. Mulai versi 0.8.0 dan seterusnya, jumlah *class* per *package* naik mencapai nilai stabil pada 18 *class* per *package* di 1.0.0 versi beta. Jumlah ini sedikit lebih tinggi dibandingkan dengan nilai-nilai pengamatan sebelumnya (Emanuel *et al*, 2011) yaitu antara 10 - 16 *class* per *package*.

IV.5. Evolusi Rata-rata Function per Class

Gambar 5 di bawah ini menunjukkan evolusi rata-rata *function* (metode) per *class* di FreeMind.

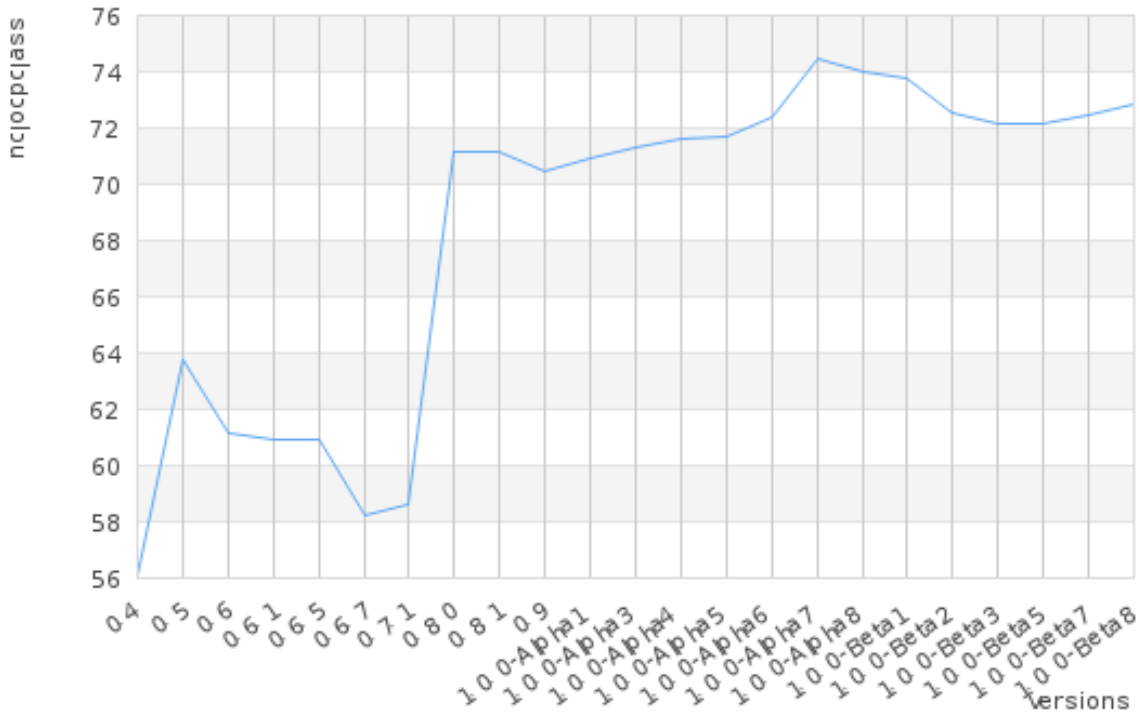


Gambar 5. Evolusi *Function* per *Class* Freemind

Gambar di atas menunjukkan bahwa rata-rata *function* per *class* bervariasi seiring dengan meningkatnya nomor versi. Pada versi 0.8.0 dan seterusnya, jumlah rata-rata *function* per *class* telah mencapai kestabilan pada 7 *function* per *class*. Nilai optimal dari *function* per *class* berdasarkan pengamatan sebelumnya (Emanuel *et al*, 2011) harus sekitar 5, sehingga nilai yang dihasilkan lebih tinggi yang menunjukkan kurang optimalnya praktek *coding*.

IV.6. Evolusi NCLOC per *Class*

Gambar 6 di bawah ini menunjukkan evolusi rata-rata NCLOC per *class* di FreeMind.

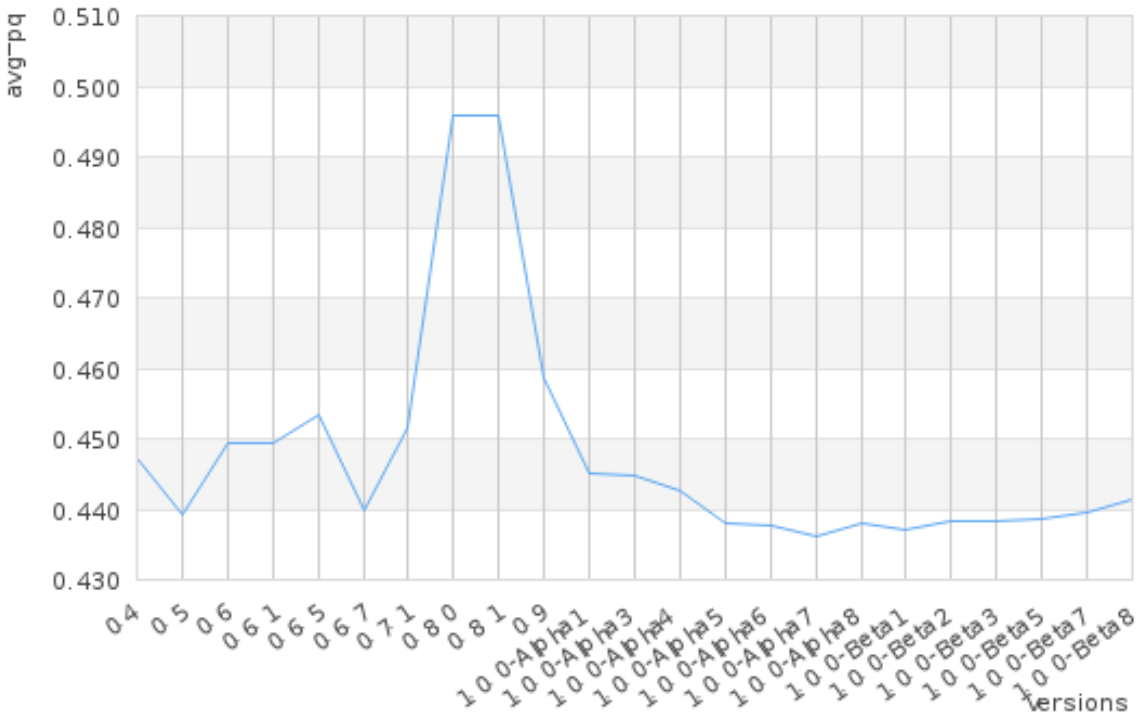


Gambar 6. Evolusi Rata-rata NCLOC per *Class* Freemind

Gambar di atas menunjukkan bahwa rata-rata NCLOC per *class* di FreeMind cenderung meningkat seiring dengan bertambahnya nomor versi. Rata-rata NCLOC per *class* berada pada nilai yang dapat diterima yaitu sekitar 80 NCLOC per kelas.

IV.7. Evolusi Rata-rata P_Q

Gambar 7 di atas menunjukkan evolusi rata-rata *Package Quality* di FreeMind.

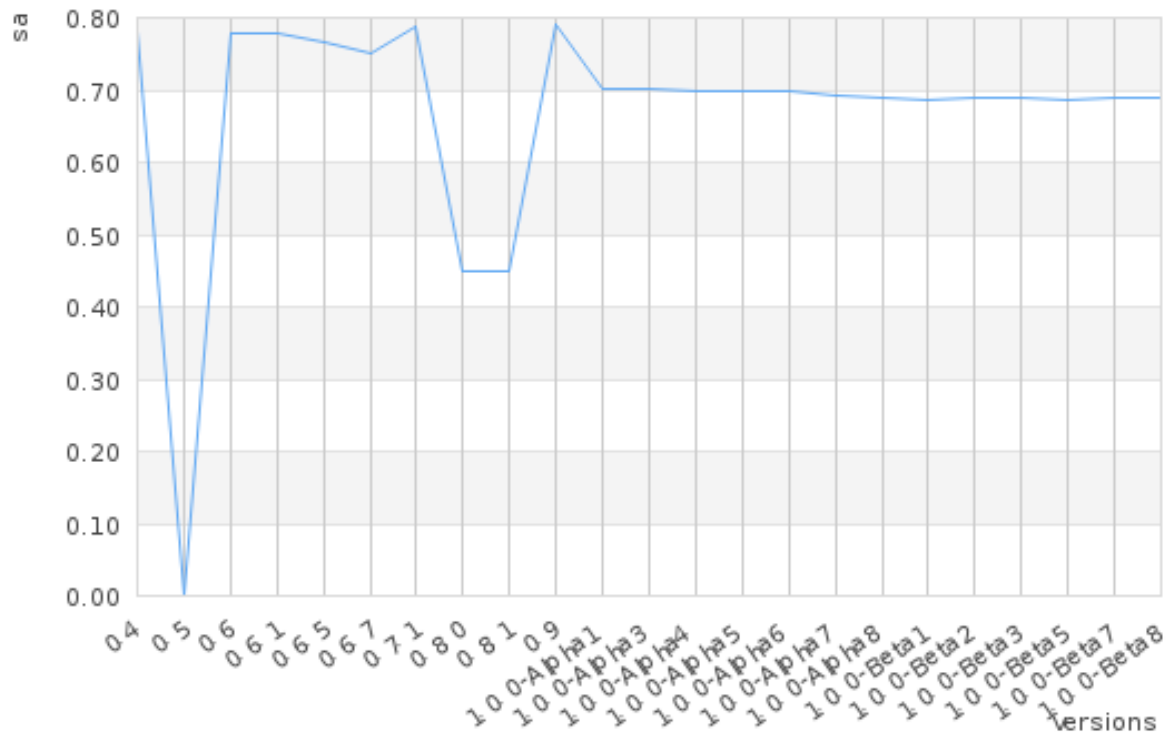


Gambar 7. Evolusi Rata-rata *Package Quality* Freemind

Gambar di atas menunjukkan bahwa rata-rata *Package Quality* pada banyak versi FreeMind relatif stabil pada sekitar 0,4 dengan beberapa pengecualian yang ditunjukkan pada versi 0.8.0 dan 0.8.1 dimana nilainya mendekati 0,5. Nilai ini rendah karena jumlah maksimum rata-rata *Package Quality* harus mendekati 1. Hal ini disebakan karena jumlah *function* per *class* mendekati 7 meskipun jumlah NCLOC per *class* sudah berada pada nilai optimal yaitu sekitar 80.

IV.8. Evolusi S_A

Gambar 8 di atas menunjukkan evolusi nilai *System Architecture* di FreeMind.

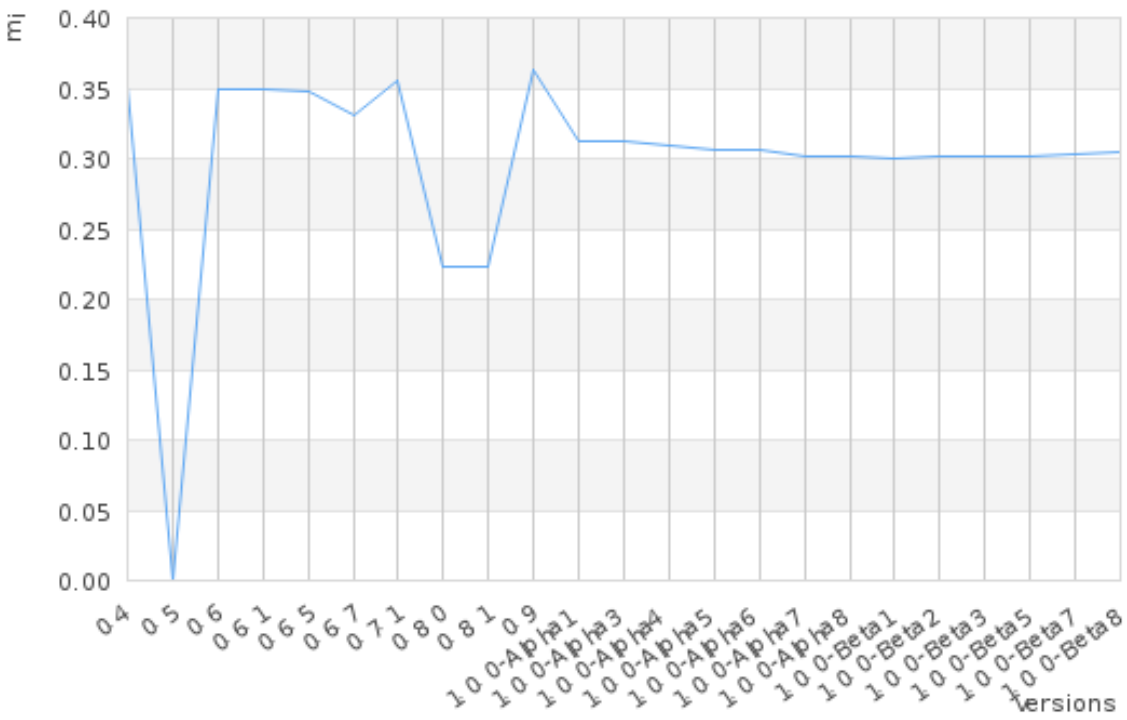


Gambar 8. Evolusi *System Architecture* Freemind

Gambar di atas menunjukkan bahwa nilai S_A di FreeMind mulai stabil pada nilai 0,7 di versi terbaru FreeMind. Nilai ini terbilang tinggi dari *system architecture* yang menunjukkan struktur ketergantungan proyek sudah menerapkan prinsip "memaksimalkan *cohesion* dan meminimalkan *coupling*".

IV.9. Evolusi M_I

Gambar di bawah ini menunjukkan evolusi Indeks Modularitas di FreeMind.



Gambar 9. Evolusi Indeks Modularitas Freemind

Gambar 9 di atas menunjukkan bahwa nilai Indeks Modularitas pada versi terbaru dari FreeMind stabil pada nilai 0,3. Karena Indeks Modularitas adalah produk dari S_A dan rata-rata $P_{Q,}$ jumlah yang rendah dari Indeks Modularitas sebagian besar disebabkan oleh rendahnya jumlah rata-rata P_Q walaupun jumlah S_A tinggi.

IV.10.Analisis Evolusi Pada Freemind

Dengan menganalisis nilai Indeks Modularitas dan semua metrik perangkat lunak pendukung, evolusi proyek OSS FreeMind memiliki beberapa kekuatan dan kelemahan yang dapat diamati. Kekuatan yang terutama ditunjukkan pada *system architecture* dimana nilai S_A terutama pada versi sekitar Beta 0,7 yang memiliki nilai tinggi. Proyek-proyek FreeMind sudah menerapkan prinsip "memaksimalkan *cohesion* dan meminimalkan *coupling*". Kekuatan lainnya adalah nilai rata-rata NCLOC per *class* mendekati nilai ideal yaitu sekitar 72-74 NCLOC per *class*.

Kelemahan dari proyek FreeMind terutama ditampilkan dalam praktek *coding*. Nilai rata-rata *class* per *package* adalah sekitar 18 *class* per *package* yang dianggap terlalu tinggi dibandingkan dengan nilai standar 10 - 16 *class* per *package*. Jumlah *function* (metode) per *class* dari sekitar 7 juga terlalu tinggi dibandingkan dengan nilai standar 5 *function* per *class*.

V. KESIMPULAN DAN SARAN

V.1. Kesimpulan

Kesimpulan yang dapat diperoleh dari penelitian ini yaitu :

1. Revisi Indeks Modularitas merupakan komposit metrik perangkat lunak untuk mengukur tingkat modularitas dari proyek OSS berbasis *Java*.
2. Terdapat beberapa modifikasi dari formulasi sebelumnya, seperti pengenalan *Cohesion Quality* (H_Q) daripada menggunakan LCOM4 metrik kohesi secara langsung.
3. Formulasi baru Indeks Modularitas merupakan produk dari *System Architecture* (S_A) dan rata - rata *Package Quality* (P_Q). Perbedaan dari formulasi sebelumnya adalah versi sebelumnya dari Indeks Modularitas merupakan produk dari *System Architecture* (S_A) dan jumlah total dari *Package Quality* (P_Q).
4. Rumusan baru menghasilkan nilai Indeks Modularitas adalah nilai ternormalisasi dengan nilai antara 0 (tidak ada modularitas) sampai dengan 1 (modularitas yang sempurna). Formulasi sebelumnya dari Indeks Modularitas memiliki nilai maksimal tidak terbatas yang menyulitkan perbandingan antara suatu proyek perangkat lunak dengan proyek – proyek yang lainnya yang memiliki dimensi (ukuran, jumlah *class*, atau jumlah *packages*) yang berbeda.
5. Penggunaan revisi Indeks Modularitas ini pada beberapa versi dari proyek OSS FreeMind menunjukkan bahwa Freemind ini memiliki nilai *System Architecture* (S_A) yang tinggi dan nilai yang rendah pada rataan *Package Quality* (P_Q) sehingga menyebabkan rendahnya jumlah Indeks Modularitas secara keseluruhan.

V.2. Saran

Saran yang dapat diberikan dari penelitian ini yaitu:

1. Melakukan pengembangan penelitian yang mencakup analisis lebih lanjut evolusi lain proyek OSS berbasis *Java* menggunakan Indeks Modularitas.
2. Selain itu dapat juga menerapkan aplikasi Indeks Modularitas pada perangkat lunak yang berorientasi obyek seperti menggunakan C#, C++, dan OOP PHP.

Daftar Pustaka

[1] DeKoenigsberg G., “How Successful Open Source Projects Work, and How and Why to Introduce Students to the Open Source World”, 21st IEEE Conference on Software Engineering Education and Training 2008.

[2] Stamelos I., Angelis L., Oikonomou A., and Bleris G.L., “Code Quality Analysis in Open Source Software Development”, Information Systems Journal vol. 12 no. 1, pp 43 – 60, 2002.

[3] Aberdour M., "Achieving Quality in Open Source Software", IEEE Software, vol. 24 no. 1, pp 58 – 64, 2007.

[4] Emanuel A.W.R., Wardoyo R., Istiyanto J.E., and Mustofa K., "Modularity Index Metrics for Java-Based Open Source Software Projects", International Journal of Advanced Computer Science and Applications (IJACSA) Vol. 2 No. 11, November 2011.

[5] Capiluppi A., and Ramil J.F., “Studying the Evolution of Open Source Systems at Different Levels of Granularity: Two Case Studies”, IEEE IWPSE 2004.

[6] Cai Y., and Huynh S., “An Evolution Model for Software Modularity Assessment”, Proceeding of the Fifth International Workshop on Software Qualty 2007 (WoSQ'07). Minneapolis, Minnesota, 20 - 26 May 2007, pp 3.

[7] Gurbani V. K., Garvert A., and Herbsleb J. D., “A Case Study of a Corporate Open Source Development Model”, Proceeding of the 28th International Conference on Software Engineering 2006, pp 472 - 481.

[8] Chidamber S.R., and Kemerer C.F., “A Metrics suite for Object Oriented Design”, IEEE Transaction on Software Engineering, Vol. 20 No. 6 June 1994, pp 476 – 493.

[9] Asundi, J., “The Need for Effort Estimation Models for Open Source Software Projects”, Proceeding of Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE), 17 May 2005.

[10] Aruna M., M.P. Suguna Devi M.P, and Deepa M., “Measuring the Quality of Software Modularization using Coupling-Based Structural Metrics for an OOS System”, Proceeding of the First International Conference on Emerging Trends in Engineering and Technology. 16 - 18 July 2008, pp 1130 – 1135.

[11] Fiondela L., and Gokhale S.S., "Importance Measures for a Modular Software System", Proceeding of the Eighth International Conference on Quality Software, 2008, pp 338 – 343.

[12] Ammar H., Shereshevsky M., Mili A., Rabie W., and Radetsky N., "Software Architecture Metrics", Seminar Presentation, Faculty of Information Science & Engineering, Management & Science University, Shah Alam, Malaysia, May 12, 2008. Available: http://www.docstoc.com/docs/6802629/Software-Architecture-Metrics

[13] ____, "History of Mind Mapping", The Mind Mapping Site, accessed: 15 November 2012. Available: http://www.mindmappingsite.com/history/80-history-of-mind-mapping.

Lampiran

Lampiran A: Paper Publikasi di IJCA

Lampiran B: Anggaran Penelitian

Lampiran C: Data Evolusi NCLOC, Packages, class, Rata – Rata class per Package, Rata – Rata Function per class, NCLOC per class, Rata – Rata PQ, SA, dan MI pada berbagai versi Freemind

Lampiran A: Paper Publikasi di IJCA

Lampiran B: Anggaran Penelitian

Berikut adalah rincian pengeluaran untuk penelitian ini yang menghabiskan dana Rp. 6.170.000,0 (enam juta seratus tujuh puluh ribu rupiah)

No	Item	Harga	Keterangan
1	Honor Peneliti Utama	Rp. 3.000.000,-	
2	Honor Peneliti Tambahan	Rp. 2.250.000,-	
3	Casing CPU Power Logic	Rp. 260.000,-	Casing untuk pengganti casing CPU lama yang kondisinya kurang baik dan cepat panas
4	Duplikasi Jurnal IJCA	Rp. 196.000,-	Duplikasi Jurnal IJCA sebanyak 4 buah
5	Flash Disk Sony 16 GB (2)	Rp. 206.400,-	Flash disk sebagai tempat penyimpanan data untuk peneliti utama dan peneliti tambahan
6	Cetak dan Jilid Laporan Penelitian	Rp. 124.000,-	Fotocopy 4 jidid laporan penelitian dan hardcover 5 bundel laporan penelitian
7	Catridge Tinta Warna 802	Rp. 100.000,-	Catridge Warna tipe 802 untuk tinta printer
8	Kertas Bola Dunia 1 rim A4 70 gram	Rp. 33.600,-	Kertas untuk cetak laporan dan dokumen lainnya
	TOTAL	**Rp. 6.170.000,-**	

Lampiran C: Data Evolusi dari berbagai versi *Freemind*

Project Name	Version	Mod Idx	S_A	avg P_Q	packages	classes	ncloc	lines	statements	files	functions	complexity
Freemind	0.4	0.3502	0.7829	0.4473	9	104	5,835	10,547	2,739	66	699	1,450
Freemind	0.5	0	0	0.4393	9	109	6,951	12,303	3,507	69	805	1,816
Freemind	0.6	0.35	0.7786	0.4495	9	121	7,396	12,701	3,886	71	910	2,019
Freemind	0.6.1	0.35	0.7786	0.4495	9	122	7,429	12,735	3,906	71	911	2,030
Freemind	0.6.5	0.347	0.7655	0.4533	9	135	8,229	13,876	4,520	71	975	2,331
Freemind	0.6.7	0.3303	0.751	0.4398	9	162	9,430	15,653	5,238	74	1,101	2,664
Freemind	0.7.1	0.3555	0.7874	0.4515	9	198	11,606	19,057	6,470	92	1,320	3,252
Freemind	0.8.0	0.2223	0.4483	0.4959	26	397	28,239	47,568	13,926	237	2,812	6,707
Freemind	0.8.1	0.2223	0.4483	0.4959	26	397	28,238	47,567	13,925	237	2,812	6,707
Freemind	0.9	0.3628	0.791	0.4587	48	763	53,734	86,244	25,496	444	5,454	12,328
Freemind	1.0.0-Alpha1	0.3122	0.7015	0.445	48	768	54,477	87,278	25,836	447	5,512	12,497
Freemind	1.0.0-Alpha3	0.312	0.7012	0.4449	48	770	54,919	87,825	26,056	449	5,540	12,566
Freemind	1.0.0-Alpha4	0.3091	0.6985	0.4425	48	773	55,375	88,463	26,301	451	5,582	12,669
Freemind	1.0.0-Alpha5	0.3056	0.6975	0.4381	48	774	55,478	88,607	26,358	450	5,616	12,728
Freemind	1.0.0-Alpha6	0.3054	0.6978	0.4377	48	784	56,768	90,625	26,942	457	5,764	12,986
Freemind	1.0.0-Alpha7	0.3021	0.6928	0.436	49	800	59,590	96,949	27,590	465	5,887	13,258
Freemind	1.0.0-Alpha8	0.3016	0.6885	0.438	49	824	60,969	98,853	28,203	467	6,008	13,553
Freemind	1.0.0-Beta1	0.3004	0.6874	0.4371	49	838	61,787	99,969	28,603	468	6,073	13,717
Freemind	1.0.0-Beta2	0.3016	0.6882	0.4383	49	861	62,464	101,173	28,844	475	6,160	13,876
Freemind	1.0.0-Beta3	0.3017	0.6885	0.4383	49	871	62,858	101,840	29,005	477	6,213	13,950
Freemind	1.0.0-Beta5	0.3014	0.6873	0.4386	49	876	63,203	102,346	29,154	478	6,241	14,016
Freemind	1.0.0-Beta7	0.3026	0.6885	0.4396	49	884	64,027	103,497	29,498	481	6,291	14,144
Freemind	1.0.0-Beta8	0.3039	0.6885	0.4414	50	898	65,408	105,653	30,088	490	6,429	14,419