KAJIAN *VERSION CONTROL* DALAM MENDUKUNG KINERJA *DEVELOPER* PADA PT. JAWASOFT

Nur Shobi Mabrur
Program Studi Informatika
Fakultas Teknik Universitas Muhammadiyah Tangerang
Jl. Perintis Kemerdekaan 1/33 Cikokol Kota Tangerang

Abstract - Version Control is a system that keeps track of the historical changes and usage of a resource within an integrated storage media. Version Control (also widely known as a Source Code Management System) is a crucial component for any team working on software development. Version control will keep a record of every single source code change to a file or document, such as historical data, comparisons between different versions, and even access rights to that file or document. A wise choice on which Version Control system to use will have a massive effect on the performance of the Software Developers. Hence many factors must be acknowledged in the process of making the decision of which product to use. The objective of this research is to find which features/attributes of a Version Control System are most influential in the aforementioned decision making process. The research explores the features/attributes inherent in three well known Version Control systems – Subversion, Mercurial and Git. This is done using ISO 9126 as a criteria. Influential factors in the decision making process are gauged using a descriptive analysis technique, and the instrument used in this case is the Analytical Hierarchy Process (AHP). Extensive questionnaire data was 'fed' into the AHP technique, and based on the results obtained, it was calculated that Subversion is the more favorable Version Control system compared to Mercurial and Git. It can thus be concluded that Subversion is the best Version Control system to support the productivity of software developers.

Keywords: Version Control, Subversion, Mercurial, Git, Analytical Hierarchy Process.

I. PENDAHULUAN

A. Latar Belakang

Dalam suatu Sistem Informasi berskala besar, susunan baris kode yang dihasilkan sudah pasti memiliki jumlah yang sangat besar pula, ribuan bahkan jutaan. Baris kode yang banyak ini membuat para *Software Developer* kewalahan dalam me-*maintain* jika suatu saat terjadi perubahan baik yang bersifat kecil ataupun besar. Terlebih lagi jika mereka bekerja dalam tim.

Dengan *Version Control* memudahkan *Software Developer* dalam memanage *source code* mereka. Fitur-fitur dasar yang harus ada dari *Version Control* seperti data historikal mengenai suatu dokumen, komparasi dari tiap-tiap perubahan hingga hak akses dari dokumen tersebut. Tak hanya itu, dokumen lain selain dokumen yang memuat *source code* dapat dikontrol dengan baik. Kendala waktu, jarak dan lokasi pun sudah dapat diatasi karena pusat penyimpanannya yang terintegrasi dalam satu lokasi. Sehingga dengan demikian para *Software Developer* yang berbeda lokasi dan waktu dapat berkolaborasi tanpa memperhitungkan masalah-masalah tersebut.

PT.Jawasoft merupakan perusahaan yang sedang berkembang pesat dalam mengembangkan perangkat lunak. *Developer* yang bekerja di perusahaan tersebut tidak hanya saja berada di Indonesia, tetapi juga terdapat di beberapa negara seperti Cina, Vietnam dan Amerika Serikat. Keadaan ini menjadi kendala bagi perusahaan tersebut dalam proses pengembangan perangkat lunak yang dapat menyebabkan *code conflict*. Oleh karena itu, *Version Control* yang tepat sangat diperlukan.

B. Batasan Masalah

1. Meneliti semua teknik *Version Control* dengan batasan tenaga, waktu, dan biaya dibatasi dengan *Version Control* yang sudah ada.
2. *Version Control* yang diteliti difokuskan pada 3 (tiga) jenis *Version Control*, yaitu *Subversion*, *Mercurial* dan *Git*.
3. Penelitian ini menerapkan ISO 9126 sebagai kriteria dengan menggunakan teknik analisa deskriptif dan pendekatan yang digunakan *Analytical Hierarchy Process* (AHP).

C. Rumusan Masalah

Rumusan masalah pada penulisan ini adalah "Dari ketiga alternatif *Subversion, Mercurial,* dan *Git*, *Version Control* apa yang menjadi alternatif pilihan terbaik yang dapat mendukung kinerja *Software Developer* pada PT.Jawasoft?"

D. Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini secara garis besar adalah memberikan alternatif *version control* terbaik dalam menunjang kinerja para *Software Developer* sesuai dengan pola yang ada pada PT.Jawasoft sehingga *source code* ataupun dokumen lain yang dihasilkan dapat didokumentasikan dan dikontrol dengan baik.

Dengan penelitian ini diharapkan memberikan manfaat dalam pertimbangan manajemen dalam menentukan *version control* yang terbaik.

II. TINJAUAN PUSTAKA

A. Definisi Source Code

Dalam ilmu komputer, *source code* (atau disebut juga *source*) adalah kumpulan pernyataan atau deklarasi bahasa pemrograman komputer yang ditulis dan dapat dibaca manusia. *Source code* memungkinkan *developer* untuk berkomunikasi dengan komputer menggunakan beberapa perintah yang telah didefinisikan terlebih dahulu.

Source code merupakan input dalam sebuah proses yang menghasilkan program yang dapat dieksekusi. Belakangan ini, *source code* juga merupakan metode berkomunikasi antara satu *programmer* dengan *programmer* lainnya untuk menyampaikan algoritma yang ada didalamnya [2].

Cara bagaimana sebuah perangkat lunak ditulis dalam bentuk *source code* memiliki konsekuensi yang penting bagi *developer* terutama dalam proses perawatan. Aturan-aturan seperti kemudahan pembacaan struktur algoritma dan beberapa aturan spesifik dari bahasa pemrograman tertentu harus diperhatikan pada saat perawatan *source code* dari sebuah perangkat lunak. Hal ini juga akan mempengaruhi proses *debugging* dan update.

Prioritas lainnya, seperti kecepatan eksekusi atau proses kompilasi dari beberapa arsitektur yang ada dalam perangkat lunak sering membuat penulisan *source code* tidak diperhatikan. Kualitas dari sebuah *source code* sangat tergantung sepenuhnya dari tujuan *source code* itu dibuat [3].

B. Software Development

Pengembangan perangkat lunak (juga dikenal sebagai *Application Development, Software Design, Software Engineering, Software Application Development, Enterprise Application Development*) adalah pengembangan produk perangkat lunak dalam proses yang sistematis, terencana dan terstruktur [1].

Ada beberapa tahap umum yang terjadi pada saat pengembangan perangkat lunak terlepas dari metodologi yang dimulai dengan proses mendapatkan *requirement* hingga proses pemeliharaan. Beberapa fase tersebut antara lain [1] :

1. *Recognition of Need*
   Yaitu mendefinisikan masalah-masalah yang terjadi pada sistem.
2. *Analysis*
   Yaitu proses detail tentang berbagai kegiatan yang dilakukan oleh sebuah sistem dan hubungan mereka kedalam dan keluar dari sistem itu sendiri.
3. *Design*
   Desain menggambarkan sebuah akhir sistem dan proses yang dikembangkan.
4. *Coding*
   Coding merupakan proses men-terjemahkan desain dari sistem ke dalam *source code* yang ada dalam bahasa pemrograman.

5. *Testing*
   Mengukur kualitas kontrol utama yang digunakan selama pengembangan perangkat lunak. Fungsinya adalah untuk mendeteksi kesalahan-kesalahan yang ada dalam perangkat lunak.
6. *Implementasi*
   Tidak seperti pada fase disain, fase implementasi ini berkosentrasi pada pelatihan user, pemilihan lokasi, serta persiapan dan pengkonversian *file*.
7. *Maintenance*
   Maintenance merupakan fase yang penting dalam tahap pengembahan perangkat lunak. Banyak dari proses pemeliharaan dapat mengkonsumsi lebih banyak waktu dari waktu yang dikonsumsi dalam pengembangan.

Beberapa metodologi yang sudah banyak digunakan antara lain: *Waterfall*, *Spiral* dan metodologi berorientasi objek.

C. Kinerja Developer

Kinerja karyawan merupakan sesuatu yang mempengaruhi seberapa banyak para karyawan memberikan kontribusi dari segi kuantitas dan kualitas *output* dari pekerjaan yang mereka lakukan, lamanya waktu yang dibutuhkan untuk menghasilkan *output*, kehadiran karyawan dan lain sebagainya. *Output* yang dihasilkan berupa *source code* yang merupakan bahan utama dalam proyek perangkat lunak.

Untuk menghasilkan *code* yang baik tentu saja beberapa hal harus diperhatikan. Secara teknis, suatu sistem yang dapat digunakan untuk me-*manage code* yang dihasilkan sangat diperlukan, yaitu sistem yang dapat melihat histori dari sebuah *code*, mampu melakukan *trackback* jika terjadi kesalahan dan *backup* secara periodik.

D. Version Control

Version Control dikenal dengan banyak istilah. Ada yang menyebutnya sebagai *Configuration Management Tool*, *Revision Control, Source Control* atau *Source Code Management System* [CIO, 2007].

Version Control dapat membantu *programmer* baik yang bekerja individual ataupun dalam tim pengembang perangkat lunak dengan menyediakan akses kepada setiap anggota tim tanpa harus saling menimpa pekerjaan anggota tim yang lain, seperti yang terjadi jika sebuah tim pengembang menggunakan *sharing folder*.

Hal-hal yang mampu dilakukan oleh *Version Control* adalah [6] :

1. Mencatat perubahan *code* dan pembuat perubahan.
2. Menyediakan fungsi *undo* untuk mengembalikan keadaan *code* ke titik tertentu.
3. Melihat riwayat perubahan code, dari pertama dibuat hingga keadaan yang sekarang.
4. Memungkinkan penulisan *code* secara paralel tanpa ada kejadian anggota tim menimpa pekerjaan anggota tim yang lain.

Ada banyak aplikasi *Version Control* yang tersedia, beberapa aplikasi yang cukup terkenal antara lain [6] Visual Source Safe, CVS, Subversion, Mercurial dan Git.

E. Analytical Hierarchy Process

Salah satu model yang dapat digunakan sebagai proses pengambilan keputusan adalah dengan menggunakan Proses Hirarki Analitik atau yang dikenal dengan istilah *Analytical Hierarchy Process* (AHP) yang dikembangkan oleh Dr. Thomas L. Saaty dari *Wharton School of Business* pada tahun 1970-an untuk mengorganisasikan informasi dan *judgement* dalam memilih alternatif yang paling disukai [10].

Dengan menggunakan AHP, suatu persoalan yang akan dipecahkan dalam suatu kerangka berpikir yang terorganisir, sehingga memungkinkan dapat diekspresikan untuk mengambil keputusan yang efektif atas persoalan tersebut. Persoalan yang kompleks dapat disederhanakan dan dipercepat proses pengambilan keputusannya. Prinsip kerja AHP adalah menyederhanakan suatu persoalan kompleks yang tidak terstruktur, strategik, dan dinamik menjadi bagian-bagiannya, serta menata dalam suatu hirarki. Kemudian tingkat kepentingan setiap variabel diberi nilai numerik secara subyektif tentang arti penting variabel

tersebut secara relatif dibandingkan dengan variabel lain.

Adapun prinsip kerja AHP adalah sebagai berikut:

1. Penyusunan Hirarki
   Persoalan yang akan diselesaikan diuraikan menjadi unsur-unsurnya, yaitu kriteria dan alternatif, kemudian disusun menjadi struktur hirarki.
2. Penilaian Kriteria dan Alternatif
   Kriteria dan alternatif dinilai melalui perbandingan berpasangan. Skala 1 sampai 9 adalah skala terbaik dalam mengekspresikan pendapat. Nilai dan definisi pendapat kualitatif dari skala perbandingan Saaty dapat dilihat pada tabel berikut :

Tabel 1. Skala Perbandingan Saaty [10]

NILAI	KETERANGAN
1	Kriteria/Alternatif A sama penting dengan kriteria/alternatif B
3	A sedikit lebih penting dari B
5	A jelas lebih penting dari B
7	A sangat jelas lebih penting dari B
9	A mutlak lebih penting dari B
2,4,6,8	Apabila ragu-ragu antara dua nilai yang berdekatan

3. Penentuan Prioritas
   Untuk setiap kriteria dan alternatif, perlu dilakukan perbandingan berpasangan (*pairwise comparisons*). Nilai-nilai perbandingan relatif kemudian diolah untuk menentukan peringkat relatif dari seluruh alternatif.
4. Konsistensi Logis
   Semua elemen dikelompokkan secara logis dan diperingkatkan secara konsisten sesuai dengan suatu kriteria yang logis.

Penyelesaian metode pengambilan keputusan dengan AHP dapat menggunakan perangkat lunak *Expert Choice 2000* untuk perhitungan pemecahan persoalan dengan AHP yang sudah teruji kehandalannya.

F. ISO 9126

Salah satu tolak ukur kualitas perangkat lunak adalah ISO 9126, yang dibuat oleh International Organization for Standardization (ISO) dan International Electrotechnical Commission (IEC). ISO 9126 mendefinisikan kualitas produk perangkat lunak, model, karakteristik mutu, dan metrik terkait digunakan untuk mengevaluasi dan menetapkan kualitas sebuah produk software.

Dalam ISO 9126 menetapkan 6 karakteristik kualitas yaitu :

1. Fungsionalitas (*Functionality*): Kemampuan menutupi fungsi produk perangkat lunak yang menyediakan kepuasan kebutuhan *user*.
2. Kehandalan (*Reliability*)*:* Kemampuan perangkat lunak untuk dapat diandalkan, seyogyanya dapat berfungsi 100% waktu.
3. Penggunaan (*Usability*)*:* Kemampuan yang berhubungan dengan penggunaan perangkat lunak.
4. Efisiensi (*Efficiency*)*:* Kemampuan yang berhubungan dengan sumber daya fisik yang digunakan ketika perangkat lunak dijalankan.
5. Pemeliharaan (*Maintainability*)*:*Kemampuan yang dibutuhkan untuk membuat perubahan perangkat lunak.
6. Portabilitas (*Portability*)*:* Kemampuan yang berhubungan dengan kemampuan perangkat lunak untuk dapat dikirim ke lingkungan berbeda.

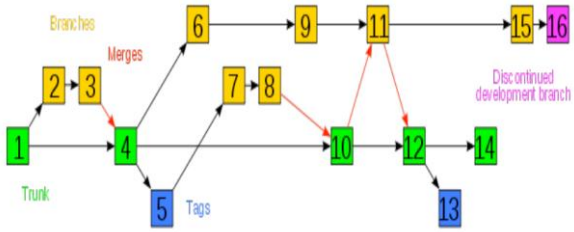
G. Version Control Subversion

Subversion (sering disingkat svn) adalah *version control* yang diperkenalkan pada tahun 2000 dan disponsori oleh CollabNet Inc. sebagai pengganti dari CVS yang memiliki kekurangan antara lain:

1. Tidak mendukung *atomic commit*
2. Tidak mendukung penyimpanan file *binary*
3. Tidak mendukung rename file atau folder
4. Tidak dapat menyimpan perubahan pada file yang sudah dihapus
5. Ijin akses tidak dapat diatur per folder.

Subversion menggunakan model file bercabang untuk menangani *branches* dan *tags*. *Branches* adalah bagian yang terpisah dari

pengembangan. Sedangkan *tagging* merupakan *snapshot* dari semua isi repositori yang tidak seperti cabang, tidak akan berubah dalam proses pengembangan selanjutnya.



Gambar 1. Visualisasi *branch* dan *tag* dalam subversion [14]

Subversion merupakan salah satu *version control* yang sifatnya gratis dan legal (*open source*). Selain itu *Subversion* sudah menjadi standard *de facto version control* di dunia *open source.*

H. Version Control Mercurial

Mercurial adalah sebuah *version control* untuk *developer* perangkat lunak yang bersifat terdistribusi dan *cross-platform*. *Mercurial* awalnya ditulis untuk berjalan pada platform Linux. Tetapi saat ini telah di-*porting* hampir ke dalam semua Sistem Operasi seperti Windows, Mac OS X, FreeBSD dan sistem Unix.

Disain dari tujuan utama Mercurial meliputi [SULLIVAN, 2009] :

1. Sangat mudah dipelajari dan digunakan
2. Sangat ringan
3. Sangat mudah diskalakan
4. Sangat mudah untuk disesuaikan

Mercurial pertama kali diperkenalkan pada tanggal 19 April 2005 oleh Mackall atas dorongan karena pada awal bulan tersebut, *Bitmover* menarik versi gratis dari *BitKeeper*. *Mercurial* pada dasarnya adalah program yang berbasis *command line*. Semua operasi *Mercurial* dieksekusi dengan menggunakan pilihan kata kunci "hg".

I. Version Control Git

Git adalah *version control* yang terdistribusi dengan penekanan pada kecepatan. Setiap repositori pada direktori kerja *Git* penuh dengan histori lengkap dan kemampuan pelacakan penuh dari sebuah revisi. *Git* memiliki perintah yang sangat besar, dengan versi 1.5.0 terdapat 139 perintah individu. Beberapa keunggulan *Git* antara lain [4] :

1. Dukungan penuh untuk pengembangan yang bersifat non-linear sperti branch dan merging
2. Proses pengembangan yang terdistribusi, namun tidak seperti mercurial, Git memiliki kemampuan untuk menyimpan revisi yang terakhir dimana suatu file telah di copy
3. Kompatibilitasnya dengan sistem/protokol yang sudah ada seperti HTTP, FTP, rsync
4. juga emulasi CVS bagi proyek yang telah menggunakan CVS
5. Penanganan yang efisien untuk proyek yang besar
6. Dan beberapa keunggulan lainnya.

III. METODE PENELITIAN

A. Metode Penelitian

Metode penelitian ini adalah penelitian kualitatif dengan langsung membandingkan obyek yang diteliti dan kuantitatif dengan menggunakan survei. Penelitian kuantitatif dapat diartikan sebagai proses pemecahan masalah yang diselidiki dengan memberikan bobot pada kriteria, subkriteria dan alternatif pada obyek penelitian.

B. Metode Pengumpulan Data

Proses pengumpulan data dimulai dengan mencari data primer, dengan melakukan survei sesuai dengan kebutuhan dan kondisi yang ada. Pada saat yang bersamaan peneliti juga mencari data sekunder guna memperkaya pengetahuan dan literatur. Setelah data yang diperoleh memadai, maka peneliti melakukan analisa kebutuhan dan membuat model dalam bentuk kuesioner. Selanjutnya kuesioner ini diberikan kepada beberapa responden yang terkait.

C. Instrumentasi Penelitian

Penelitian ini menggunakan angket atau kuisioner yang digunakan sebagai pembanding dari hasil observasi langsung obyek yang diteliti dengan pendekatan ISO 9126 guna

memperoleh data dalam proses *penentuan Version Control* terbaik yang mendukung kinerja *developer*.

D. Teknik Analisis Data

Analisis yang digunakan dalam penelitian ini adalah analisis deskriptif dan *Analytical Hierarchy Process* (AHP). Analisis deskriptif dilakukan dengan meneliti langsung obyek penelitian yaitu dengan menggunakan dan membandingkan obyek-obyek yang diteliti melalui penyajian rangkuman hasil *survey* dan identifikasi dalam bentuk tabulasi dan/atau grafik. Secara garis besar, teknik analisis data yang digunakan adalah:

1. Menentukan besarnya bobot yang dimulai dari kasus khusus yang sederhana sampai dengan kasus-kasus umum dengan menggunakan penyelesaian persamaan matematik.
2. Pengolahan data horisontal untuk menyusun prioritas elemen keputusan setiap tingkat hierarki keputusan.
3. Pengolahan vertikal untuk menyusun prioritas setiap elemen dalam hierarki terhadap sasaran utama.
4. Hasil penelitian diolah dan dibandingkan dengan menggunakan metode *Analytical Hierarchy Process* (AHP) dengan *software Expert Choice 2000*.

IV. ANALISIS DAN PEMBAHASAN

A. Kriteria dalam Version Control

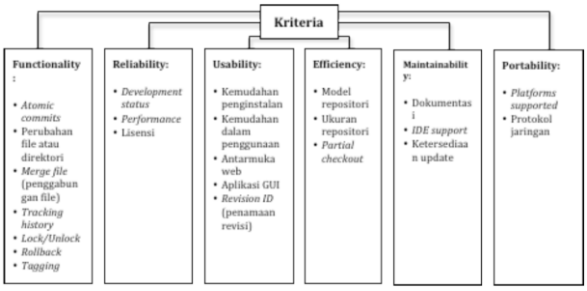
Berdasarkan ISO 9126 yang dibuat oleh *International Organization for Standardization* (ISO) dan *International Electrotechnical Commission* (IEC), berikut ini beberapa hal yang nantinya akan dilakukan pengujian berdasarkan kriteria tersebut:

1. *Functionality*, antara lain :
   a. *Atomic Commits*, yaitu sebuah operasi dimana lebih dari satu perubahan diaplikasikan dalam satu kali operasi.
   b. Perubahan file atau direktori, yaitu kemampuan untuk memin-dahkan letak (*move*), pengubahan nama (*rename*), menghapus isi (*delete*), menduplikasi (*copy*), ataupun menggabungkan isi (*merge*) dari file atau direktori dalam repositori.
   c. *Merge file*, yaitu kemampuan untuk mengga-bungkan perubahan yang dibuat pada satu *branch* kedalam file atau direktori yang sama pada *branch* yang berbeda atau sebaliknya.
   d. *Tracking history*, yaitu kemampuan untuk melacak perubahan-perubahan yang telah terjadi antara cabang yang satu dengan cabang yang lain.
   e. *Lock*/*Unlock*, yaitu kemampuan mengunci suatu file atau direktori.
   f. *Rollback*, yaitu kemampuan untuk mengembalikan sebuah revisi ke revisi tertentu.
   g. *Tagging*, yaitu kemampuan untuk memberikan nama khusus pada revisi tertentu. Beberapa *version control* menyebutnya label.
   h. File *unicode*, yaitu kemampuan untuk mendukung sistem file yang memiliki pengkodean karakter yang berbeda.
2. *Reliability*, antara lain :
   a. *Development status***,** yaitu status dari pengembangan *version control* hingga saat ini.
   b. *Performance*, yaitu level yang digunakan untuk mengukur performa fungsionalitas dari *version control*.
   c. Lisensi, yaitu model lisensi yang digunakan
3. *Usability*, antara lain :
   a. Kemudahan penginstalan, yaitu sebuah ukuran dimana *version control* mudah dalam penginstalan.
   b. Kemudahan dalam penggunaan, yaitu sebuah ukuran dimana perangkat lunak mudah dipelajari dan digunakan dalam proses pengembangan.
   c. Antarmuka web, yaitu keter-sediaan antarmuka web oleh sebuah *version control*.
   d. Aplikasi *GUI*, yaitu ketersediaan aplikasi yang bersifat *GUI* yang dimiliki oleh *version control*.
   e. *Revision ID* (penamaan revisi), digunakan untuk mengidentifikasi versi tertentu dari sebuah file dalam repositori.

4. *Efficiency*, antara lain :
   a. Model repositori, yaitu model repositori yang digunakan.
   b. Ukuran repositori, menggambarkan tingkat pertumbuhan repositori jika setiap saat perubahan dilakukan.
   c. *Partial checkout*, yaitu kemampuan untuk mengambil atau mengklon sebagian direktori tertentu dari repositori.
5. *Maintainability*, antara lain :
   a. Dokumentasi, yaitu level keter-sediaan dokumentasi yang dimiliki dari *version control* yang digunakan oleh pengembang.
   b. *IDE support* (*Integtated Development Environment*), yaitu level dukungan yang diberikan oleh IDE untuk tiap-tiap *version control*.
   c. Ketersediaan update, yaitu update yang tersedia dari *version control* untuk mengatasi perubahan kebutuhan dari para *developer.*

6. *Portability*, antara lain :
   a. Platforms supported, yaitu sistem operasi yang didukung oleh sebuah *version control.*
   b. Protokol jaringan, yaitu protokol jaringan yang didukung oleh *version control.*

Dari 6 kriteria kualitas tersebut dapat digambarkan dalam Gambar IV-1 sebagai berikut:



Gambar 2 Model Spesifikasi *version control*

B. Pembahasan Hasil Pengamatan

1. Berdasarkan kriteria *functionality*

Berdasarkan kriteria *functionality*, maka didapatkan hasil seperti berikut:

Tabel 2. Hasil pengamatan *functionality*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Atomic commits	Ada (*branch*)	Ada	Ada
Perubahan file atau direktori	Semua fitur dasar, kecuali *merge file*	Semua fitur dasar	Semua fitur dasar
Merge file	Tidak ada	Ada	Ada
Tracking history	Ada	Ada	Ada
Lock/ Unlock	Ada	Tidak ada	Tidak ada
Rollback	Tidak ada (*revert*)	Ada	Ada
Tagging	Ada (*branch*)	Ada	Ada
File unicode	Didukung semua *OS*	Sebagi-an *OS*	Sebagi -an *OS*

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 3. Hasil pembobotan *functionality*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Atomic commits	4	3	3
Perubahan file atau direktori	3	3.5	3.5
Merge file	0	5	5
Tracking history	3.5	3.5	3
Lock/ Unlock	10	-	-
Rollback	1	4.5	4.5
Tagging	3	3.5	3.5
File unicode	5	2.5	2.5
Total	29.5	25.5	25

Berdasarkan Tabel 3, *Subversion* memiliki nilai bobot yang paling tinggi.

2. Berdasarkan kriteria *reliability*

Berdasarkan kriteria *reliability*, maka didapatkan hasil seperti berikut:

Tabel 4. Hasil pengamatan *reliability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Development status	Aktif dikembangkan (*stable*)	Aktif dikembangkan (*stable*)	Aktif (*stable*)
Performance	Baik (berdasarkan infrastruktur dan jaringan)	Baik (berdasarkan infrastruktur dan jaringan)	Baik (berdasarkan infrastruktur dan jaringan)
Lisensi	Apache	GPL	GPL

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 5. Hasil pembobotan *reliability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Development status	3	4	3
Performance	3	3.5	3.5
Lisensi	3	3.5	3.5
Total	9	11	10

Berdasarkan Tabel 5, *Mercurial* memiliki nilai bobot yang paling tinggi.

3. Berdasarkan kriteria *usability*

Berdasarkan kriteria *usability*, maka didapatkan hasil seperti berikut:

Tabel 6. Hasil pengamatan *usability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Kemudahan penginstalan	Mudah	Sedang (adanya dependensi)	Mudah
Kemudahan penggunaan	Mudah	Sedang (banyak perintah dasar)	Mudah
Antarmuka Web	Ada (plugin)	Ada (disertakan dalam penginstalan)	Ada (plugin)
Aplikasi GUI	Ada	Ada	Ada
Revision ID	Numerik	Numerik/*Hash*	*Hash*

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 7. Hasil pembobotan *usability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Kemudahan penginstalan	4	2	4
Kemudahan penggunaan	3.5	3	.3.5
Antarmuka Web	2.5	5	2.5
Aplikasi GUI	4	3	3
Revision ID	3.5	4	2.5
Total	17.5	17	15.5

Berdasarkan Tabel 7, *Subversion* memiliki nilai bobot yang paling tinggi.

4. Berdasarkan kriteria *efficiency*

Berdasarkan kriteria *efficiency*, maka didapatkan hasil seperti berikut:

Tabel 8. Hasil pengamatan *efficiency*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Model repositori	*Trunk, branch* dan *tag*	Distributif	Distributif
Ukuran repositori	*Patch* (*branch*)	Revisi (*clone*)	Revisi (*clone*)
Partial checkout	Ya	Tidak	Tidak

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 9. Hasil pembobotan *efficiency*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Model repositori	4	3	3
Ukuran repositori	5	2.5	2.5
Partial checkout	10	-	-
Total	19	5.5	5.5

Berdasarkan Tabel 9, *Subversion* memiliki nilai bobot yang paling tinggi.

5. Berdasarkan kriteria *maintainability*

Berdasarkan kriteria *maintainability*, maka didapatkan hasil seperti berikut:

Tabel 10. Hasil pengamatan *maintainability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Dokumen-tasi	Ada (*online* dan dapat diunduh)	Ada (*online* dan dapat diunduh)	Ada (*online*, diunduh)
IDE *Support*	*Netbeans, Eclipse, Visual Studio, Komodo*	*Netbeans, Eclipse, Visual Studio*	*Netbeans, Eclipse, Visual Studio*
Update	Aktif	Sangat aktif	Aktif

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 11. Hasil pembobotan *maintainability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Dokumentasi	3.5	3.5	3
IDE *Support*	3.5	3.5	3
Update	3.5	3.5	3
Total	10.5	10.5	9

Berdasarkan Tabel 11, *Subversion* dan *Mercurial* memiliki nilai bobot yang sama.

6. Berdasarkan kriteria *portability*

Berdasarkan kriteria *portability*, maka didapatkan hasil seperti berikut:

Tabel 12. Hasil pengamatan *portability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Platforms supported	Unix, Windows, Mac OS X	Unix, Windows, Mac OS X	POSIX, Windows, Mac OS X
Protokol jaringan	SSH, HTTP dan SSL (LDAP)	SSH, HTTP (LDAP)	SSH, HTTP/ HTTPS

Berdasarkan hasil tersebut dilakukan pembobotan seperti berikut:

Tabel 13. Hasil pembobotan *portability*

Version control / Fitur	*Subversion*	*Mercurial*	*Git*
Platforms supported	3.5	3.5	3
Protokol jaringan	3.5	3	3.5
Total	7	6.5	6.5

Berdasarkan Tabel 13, *Subversion* memiliki nilai bobot yang paling tinggi.

C. Bobot Keseluruhan

Dari hasil pembobotan yang dilakukan pada masing-masing kriteria, maka dapat dibuat matriks bobot keseluruhan alternatif seperti pada Tabel 14.

Tabel 14. Bobot keseluruhan alternatif

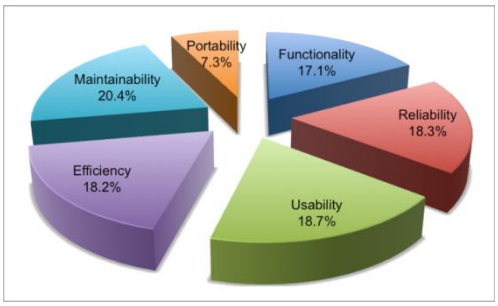
Fitur \ *Version control*	*Subversion*	*Mercurial*	*Git*
Functionality	29.5	25.5	25
Realibility	9	11	10
Usability	17.5	17	15.5
Efficiency	19	5.5	5.5
Maintainability	10.5	10.5	9
Portability	7	6.5	6.5
Total	92.5	76	71.5

Berdasarkan hasil ini didapat bahwa *version control Subversion* adalah alternatif terbaik dari ketiga *version control* yang dibandingkan. Hasil ini terbukti sesuai dengan hipotesa penelitian ini yang menduga bahwa *version control Subversion* diduga lebih baik dari *Mercurial* dan *Git*.

D. Kuisioner dengan AHP

1. Bobot Kriteria

Analisis pendapat gabungan para responden menunjukkan kriteria *maintainability* merupakan kriteria yang paling penting dari kriteria yang dimiliki oleh sebuah *version control* dengan bobot 0.204 atau 20.4% dari total kriteria.

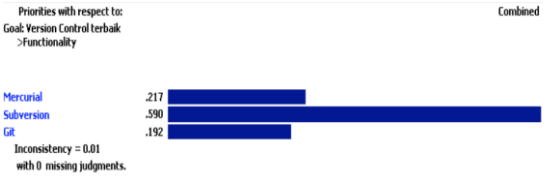


Gambar 3. Grafik nilai bobot kriteria

2. Bobot Alternatif pada Kriteria

a. Bobot alternatif pada *functionality*

Berdasarkan kriteria *functionality* didapatkan hasil sebagai berikut:



Gambar 4. Bobot alternatif *functionality*

Dari grafik tersebut didapat bahwa *Subversion* mengungguli *version control* lainnya dengan nilai bobot 0.590 atau 59%.

b. Bobot alternatif pada *reliability*

Berdasarkan kriteria *reliability* didapatkan hasil sebagai berikut:

Gambar 5. Bobot alternatif *reliability*

Dari grafik tersebut didapat bahwa *Subversion* mengungguli *version control* lainnya dengan nilai bobot 0.539 atau 53.9%.

c. Bobot alternatif pada *usability*

Berdasarkan kriteria *usability* didapatkan hasil sebagai berikut:

Gambar 6. Bobot alternatif *usability*

Dari grafik tersebut didapat bahwa *Subversion* mengungguli *version control* lainnya dengan nilai bobot 0.661 atau 66.1%.

d. Bobot alternatif pada *efficiency*

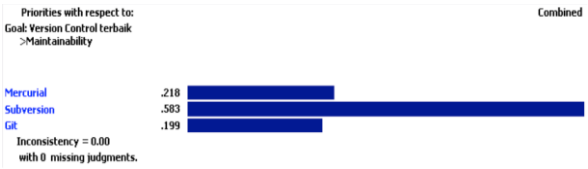
Berdasarkan kriteria *efficiency* didapatkan hasil sebagai berikut:

Gambar 7. Bobot alternatif *efficiency*

Berdasarkan Gambar 7 didapat bahwa *Subversion* masih mengungguli *version control* lainnya dengan nilai bobot 0.576 atau 57.6%.

e. Bobot alternatif pada *maintainablity*

Berdasarkan kriteria *maintainablity* didapatkan hasil sebagai berikut:



Gambar 8. Bobot alternatif *maintainablity*

Dari grafik tersebut didapat bahwa *Subversion* masih mengungguli *version control* lainnya dengan nilai bobot 0.583 atau 58.3%.

f. Bobot alternatif pada *portability*

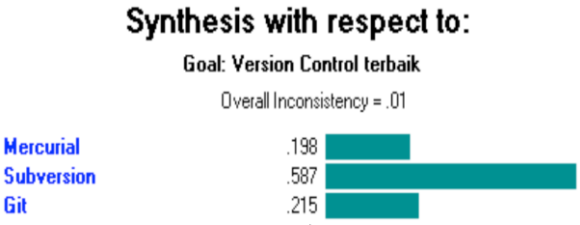
Berdasarkan kriteria *portability* didapatkan hasil sebagai berikut:

Gambar 9. Bobot alternatif *portability*

Dari grafik tersebut didapat bahwa *Subversion* tetap mengungguli *version control* lainnya dengan nilai bobot 0.582 atau 58.2%.

3. Bobot Alternatif pada Kriteria

Setelah melalui proses pengolahan data kuisioner diperoleh nilai bobot global prioritas alternatif sebagai berikut:



Gambar 10. Nilai bobot global prioritas alternatif berdasarkan sasaran

Hasil yang diperoleh dari pengolahan data responden ahli didapat bahwa prioritas utama alternatif *version control* adalah *Subversion*.

4. Perbandingan Analisa Deskriptif dengan Kuisioner

Hasil perbandingan analisa deskriptif dengan Kuisioner AHP didapat hasil sebagai berikut:

Tabel 15. Perbandingan alternatif analisa deskriptif dengan kuisioner AHP

Version control / Fitur	*Subversion*	*Mercurial*
Functionality	*Subversion*	*Subversion*
Realibility	*Mercurial*	*Subversion*
Usability	*Subversion*	*Subversion*
Efficiency	*Subversion*	*Subversion*
Maintainability	*Mercurial, Subversion*	*Subversion*
Portability	*Subversion*	*Subversion*
Keseluruhan	*Subversion*	*Subversion*

Berdasarkan Tabel 15, didapat kesamaan hasil baik dari teknik analisa deskriptif maupun hasil kuisioner yaitu alternatif *version control Subversion*. Sehingga dapat disimpulkan bahwa hasil ini terbukti sesuai dengan hipotesa penelitian ini yang menduga bahwa *version control Subversion* diduga lebih baik dari *version control Mercurial* dan *version control Git.*

V. KESIMPULAN

Berdasarkan pembahasan dari bab sebelumnya mengenai kajian *version control* baik dengan menggunakan teknik analisa deskriptif dan kuisioner AHP, maka didapat

version control Subversion merupakan alternatif terbaik yang dapat digunakan oleh tim pengembang PT.Jawasoft sebagai *version control* yang mendukung kinerja *developer* dalam proses melihat histori dari sebuah *code* atau file, baik perubahannya, siapa yang melakukan perubahan tersebut, sehingga dapat dipertanggungjawabkan dengan jelas.

Trackback pada kondisi dimana perangkat lunak tersebut dalam keadaan baik (*rollback*) jika seandainya terjadi kesalahan atau kegagalan dalam sebuah sistem akibat dari perubahan yang dibuat. *Backup* semua *source code* yang ada secara berkala pada lokasi yang berbeda.

REFERENSI

[1] B. B. Agarwal, S. P. Tayal, M. Gupta, "Software Engineering & Testing", Jones And Bartlett Publishers, Massachusets, 2008

[2] Firman Gunajaya, "Pengertian dari source code", http://blogvrman.blogspot.com/2010/03/

[3] Wikipedia, "Source Code", http://en.wikipedia.org/wiki/ Source_code (diakses 14 Mei 2011)

[4] Wikipedia,"Git (Software)", http://en.wikipedia.org/wiki/Git_(software) (diakses 14 Mei 2011)

[5] Google Trend, "Mercurial, SVN, Git", http://www.google.com/trends?q=mercurial%2C+svn%2C+git&ctab=0&geo=all&date=all&sort=0 (diakses 13 Mei 2011)

[6] Hari Mulyadi, "Manajemen Source Code dengan Subversion",http://harymulyadi.wordpress.com/2009/12/09/manajemen-source-code-dengansubversion/ (diakses 20 Mei 2011)

[7] Wikipedia, "Mercurial", http://en.wikipedia.org/wiki/Mercurial (diakses 13 Mei 2011)

[8] Kevin H.Naw, "Comparison of Version Control Systems for Software Maintenance", California, 2006

[9] Lukman Azhari, "Pemilihan Framework Aplikasi Web Berbasis Java dengan Menggunakan Analytical Hierarchy Process (AHP): Studi Komparasi Komunitas Knowledge Sharing Group (KSG) dan PT.Jawasoft", Fakultas Pascasarjana Universitas Budi Luhur, 2011

[10] Marimin, "Teknik dan Aplikasi Pengambilan Keputusan Kriteria Majemuk", PT. Gramedia Widiasarana Indonesia, Jakarta, 2005.

[11] Ratih Hafsarah Maharrani, Abdul Syukur, Tyas Catur P, "Penerapan Metode Analytical Hierarchy Process Dalam Penerimaan Karyawan Pada PT.Pasir Besi Indonesia", Jurnal Teknologi Informasi, 2010

[12] Rohit Dhiman, Christian, Sieghl, Jorg Dorr, "ISO/IEC 9126 Standard", Dept. of Computer Science ISO, 2011

[13] Brian O'Sullivan, "Mercurial: The Definitve Guide", O'Reilly Media, Pittsburg, 2009

[14] Wikipedia, "Apache Subversion", http://en. wikipedia.org/wiki/Apache_Subversion (diakses 13 Mei 2011)