BAB III

KEGIATAN SELAMA KERJA PRAKTEK

3.1 Jadwal Kerja Praktek

Penulis melaksanakan Kerja Praktek selama 30 hari mulai dari tanggal 12 Juli sampai dengan 25 Agustus 2004. Adapun waktu pelaksanaan Kerja Praktek dari hari senin sampai dengan hari jumat, dengan jam masuk disesuaikan dengan jadwal masuk kerja pembimbing kerja praktek

3.2 Data Kerja Praktek

3.2.1 Konsep dasar CVS

CVS (*Concurrent Versioning System*) adalah sebuah sistem kendali versi, atau suatu program manajemen versi yang mengatur perubahan versi file pada suatu proyek yang melibatkan banyak file dan atau banyak developer. Suatu program manajemen versi memungkinkan banyak developer mengerjakan suatu proyek secara bersama dan juga memungkinkan satu developer yang mengerjakan suatu proyek untuk mengontrol versi file - file yang sedang dikembangkannya.. CVS umumnya digunakan sebagai salah satu bagian penting dalam pengembangan perangkat lunak di mana sistem ini berfungsi melacak dan mengkoordinasikan setiap perubahan *source code* yang dilakukan oleh banyak developer. Karena kegunaannya, CVS saat ini sangat populer di dunia pengembangan perangkat lunak dengan metodologi *open source*

Sebagai contoh, dalam hal pemrograman, sebuah program biasanya tidak akan dikembangkan sekaligus tapi bertahap, mungkin ada fungsi-fungsi yang belum diimplementasikan, yang penting programnya sudah mencapai tahap tertentu. Setelah yakin bahwa semua sudah benar baru kemudian ditambahkan fungsi - fungsi yang lain. Dari banyak pengalaman, bahwa kadangkala program yang dimodifikasi menjadi *error*, tentunya kita akan mem-*back up* terlebih dahulu file itu, sebelum masuk tahap selanjutnya. Dalam hal ini sebenarnya kita telah membuat versi baru dari file kita dan kita telah melakukan manajemen versi dengan mengatur versi - versi file yang kita miliki, sehingga kita dapat memakai file (versi) yang lama dengan mudah jika terjadi kesalahan. Manajemen versi file tidak hanya dilakukan untuk *source code* program, hal ini juga bisa dilakukan untuk file apa saja.

Manajemen versi dapat dilakukan secara manual namun hal ini akan sangat menyulitkan, karena kita tidak akan berkonsentrasi pada tugas kita tapi pada proses manajemen versi itu sendiri. Program manajemen versi memungkinkan kita melakukan proses manajemen versi secara mudah dan sederhana, Kita bisa saja menyimpan setiap file yang pernah dibuat atau diubah. Namun hal ini tentu saja akan menghabiskan banyak ruang di harddisk. Hal tersebut bukan masalah jika menggunakan CVS karena CVS menyimpan setiap versi file yang diciptakan di dalam satu buah file. File yang dihasilkan oleh CVS tersebut hanya menyimpan perbedaan antara setiap versi file yang pernah dibuat, maka dengan

menyimpan hanya perubahan dari file yang kita buat di setiap versi, kita cukup mengatur hak akses terhadap file yang disimpan itu.

CVS adalah program manajemen versi yang dirancang untuk menangani proyek dengan file teks murni seperti program, dokumentasi dalam format teks (misalnya dalam *txt*, *html*, *sgml*; atau *LaTex*) bukan file atau dokumen biner seperti dokumen milik Microsoft Office

Bug biasanya muncul pada saat sebuah perangkat lunak sedang dimodifikasi, dan mungkin tidak disadari keberadaan *bug* tersebut untuk waktu yang lama atau sampai ada orang yang melaporkan kesalahan pada program itu. Dengan menggunakan CVS, kita dapat dengan mudah mendapatkan versi lama *software* untuk mengetahui perubahan apa yang dilakukan, yang menyebabkan timbulnya *bug*. Hal ini akan sangat membantu bagi para developer *software*.

.CVS juga dapat membantu jika kita adalah bagian dari sekelompok orang yang bekerja pada suatu proyek yang sama. Akan sangat mudah terjadi proses saling menimpa (*overwrite*) program yang dikerjakan oleh satu orang dengan yang lainnya sehingga kita harus sangat berhati-hati. Beberapa editor, (misalnya *GNU Emacs*) tidak mengijinkan sebuah file yang sama dimodifikasi oleh dua orang atau lebih di saat yang sama. Namun, jika orang kedua menggunakan editor selain *Emacs* fungsi tersebut menjadi tidak berfungsi. CVS dapat mengatasi masalah tersebut dengan cara memisahkan / mengisolasi developer yang satu dengan lainnya. Setiap developer bekerja di direktori kerja masing - masing,

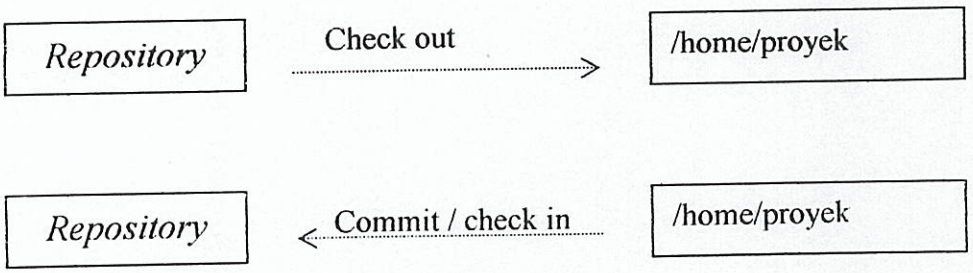
kemudian CVS akan menggabungkan proyek saat semua developer telah menyelesaikan pekerjaannya.

UNIT PELAKSANA TEKNIS PERPUSTAKAAN UNIKOM

3.2.2 Prinsip Kerja CVS

CVS merupakan suatu sistem yang berjalan secara *Client / Server*, jadi bila kita ingin menjalankan CVS kita harus terhubung dengan mesin server, tetapi CVS bisa juga berjalan secara *standalone* atau menggunakan server *localhost*

Dalam CVS, suatu file dalam proyek disimpan di suatu tempat yang disebut dengan *repository*. Jika ada yang ingin mengakses file dalam repository maka harus melakukan proses *login* ke dalam CVS setelah itu melakukan *checkout* yaitu menyalin file dari repository ke direktori tempat bekerja, setelah pekerjaannya selesai maka file – file tadi disalin kembali ke repository, disebut juga melakukan *checkin* atau *commit*.



Gambar 3.1 Prinsip kerja CVS

3.2.3 Fitur CVS

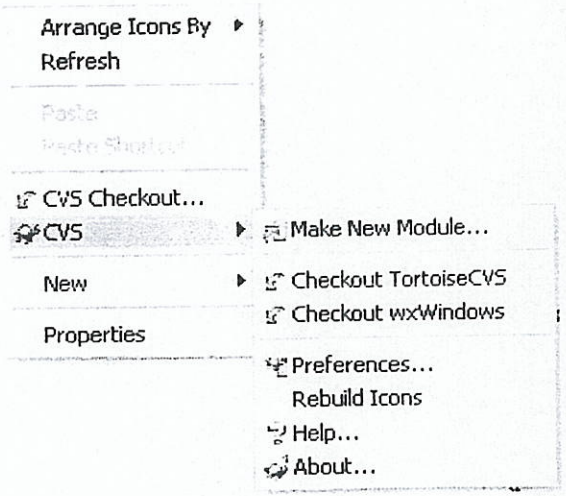
- Dapat mengambil kembali versi lama software.
- Dapat menyimpan semua versi dari tiap file yang dibuat.

- Dapat mengatur hak akses file
- Memungkinkan akses terhadap file secara bersama (konkuren) tidak hanya untuk dibaca tapi untuk ditulisi
- Menyimpan hanya perubahan file (tidak keseluruhan file)
- Dapat melakukan pencatatan perubahan yang ada pada file awal, sehingga *disk space* yang digunakan lebih sedikit
- Dapat melakukan *forking* (menciptakan cabang dalam pengembangan perangkat lunak sehingga ada dua versi atau lebih yang dikembangkan dengan jalur yang berbeda, yang berasal dari perangkat lunak yang sama)
- CVS juga bermanfaat jika kita adalah anggota sekelompok orang yang bekerja pada proyek yang sama. Dengan proyek yang dikerjakan oleh banyak orang, maka kemungkinan terjadinya konflik pada file - file *source code* sangat besar, misalnya satu orang merubah file yang telah dirubah oleh orang lainnya. CVS mengatasi masalah ini dengan cara mengisolasi para pengembang satu dengan yang lain. Setiap pengembang bekerja di direktorinya, dan CVS menggabungkan pekerjaannya ketika mereka telah selesai

3.2.4 TortoiseCVS

CVS bekerja dengan *interface command line*, ini merupakan salah satu kekurangan dari CVS tetapi kebanyakan developer sekarang ini telah terbiasa bekerja dengan berbasiskan *integrated development environments* (IDE) grafis atau *Graphical User Interface* (GUI) maka untuk memenuhi

tuntutan developer yang telah terbiasa menggunakan GUI, diciptakanlah TortoiseCVS salah satu *front-end client* CVS yang bekerja berbasiskan GUI.



Gambar 3.2 TortoiseCVS

TortoiseCVS membuat penggunaan CVS menjadi lebih mudah. TortoiseCVS memungkinkan para developer untuk bekerja dengan file yang dikontrol oleh CVS langsung dari Windows Explorer. TortoiseCVS memenuhinya dengan menyediakan *environment* "point-and-click" dengan penggunaan dan konfigurasi yang sangat mudah, Kita hanya cukup klik kanan dan semua menu dapat diakses.

3.2.5 Penggunaan Dasar TortoiseCVS

1. Direktori Kerja (*Sanboxes*)

TortoiseCVS memiliki metode kerja yang unik seperti kebanyakan *version control system* yang lain di mana beberapa developer perangkat lunak dapat mengedit sebuah file secara konkuren. Caranya, dapatkan sebuah versi *source code* dari tempat penyimpanan (repository)

kemudian *source code* itu akan disalin di sebuah direktori kerja di komputer anda. Direktori kerja inilah yang dinamakan *sandboxes*.

Kita kemudian dapat mengedit file yang telah didapatkan tadi. Kita juga dapat menambahkan file baru atau menghapus file yang tidak diperlukan lagi. Setelah selesai melakukan pengeditan, kita kemudian *commit* perubahan yang telah dilakukan ke *repository*.

Jika orang lain telah mengubah file yang sama di saat kita sedang bekerja dengan file tersebut, maka *commit* akan gagal. kita kemudian harus meng-*Update* semua file *source code* kita dari *repository*. Dengan cara ini, secara otomatis perubahan yang telah dilakukan oleh developer lain akan digabungkan (*merge*) kc dalam salinan file yang anda ambil.

Terkadang CVS tidak bisa melakukan hal tersebut secara otomatis, misalnya jika kedua belah pihak mengubah kode pada baris yang sama, hal ini dinamakan *Conflict*. *Conflict* itu sendiri tidak terlalu sering terjadi. CVS menyimpan kedua versi kode yang *conflict* tersebut di dalam file, dan diberi tanda sebagai pemisah. Kemudian kita harus mengedit file secara manual untuk memecahkan masalah *conflict* sebelum kita *commit* kembali kode kita.

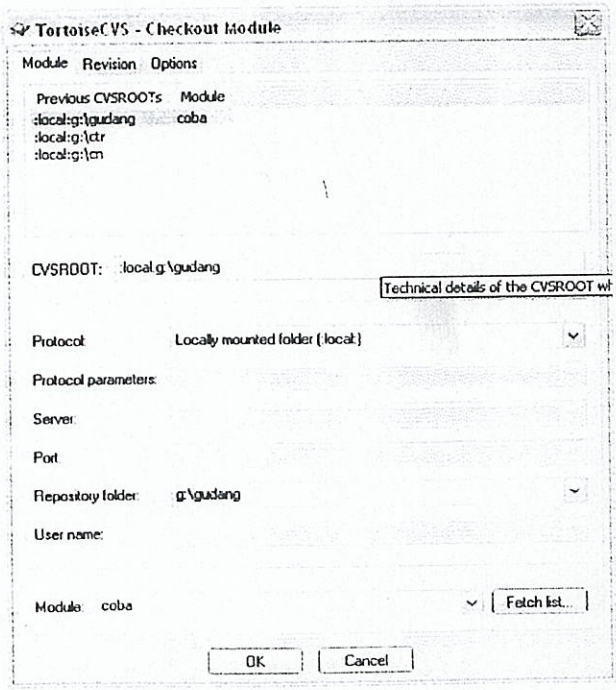
Metode kerja seperti ini memiliki banyak keuntungan. Setiap developer bekerja di *sandbox*nya masing - masing. Perubahan yang dilakukan oleh satu developer akan diisolasi dari delelover lain sampai ada developer yang akan memasukan perubahan yang telah dilakukannya. Dengan begitu *bottleneck* yang disebabkan oleh

banyaknya orang / developer yang mengantri untuk mendapatkan file yang sedang diakses oleh orang lain dapat dihindari. Setiap developer bisa bekerja pada suatu file tanpa harus mengakses langsung ke server, koneksi ke server hanya dilakukan saat akan dilakukan *update* atau *commit*.

2. Checkout Modul

Proyek yang kita kerjakan atau *source code* yang sedang kita kembangkan dalam CVS disebut modul. Modul bisa juga berbentuk suatu file

Pada saat seseorang / developer akan mendapatkan sebuah modul dari CVS server untuk pertama kalinya disebut *checkout*. Saat kita melakukan *checkout* sebuah modul dari *repository* secara otomatis akan menciptakan *sandbox* lokal dari modul tersebut.



Gambar 3.3 Checkout Modul

Untuk melakukan *checkout*, klik kanan folder di mana kita akan menyimpan modul, dan pilih **CVS Checkout...** dari menu pop-up yang muncul. Kemudian akan muncul *Checkout Dialog* dengan field - field sebagai berikut :

- **Protocol** : protokol yang digunakan saat berkomunikasi dengan *remote CVS repository*.
- **Server** : nama server yang menjadi host dari *remote CVS repository*.
- **Port** : port yang digunakan oleh *remote CVS repository*. Biasanya tidak perlu diisi.
- **Repository Directory** : lokasi CVS repository di remote server.
- **User name** : nama user dari *account* anda di *CVS repository*.
- **CVSROOT** : *connection string* secara keseluruhan, terdiri dari semua string yang anda masukkan ke field - field di atas. kita juga dapat mem-*paste connection string* yang sudah kita ketahui sebelumnya.
- **Module** : nama modul yang ingin kita *checkout*. Nama modul bersifat *case sensitive* jadi kita harus berhati-hati saat menuliskannya.

3. Repository

Repository CVS adalah suatu tempat atau folder dimana menyimpan secara lengkap salinan versi semua file dan direktori yang ada dalam *version control*

Kita tidak pernah mengakses file secara langsung dari repository, tetapi kita menggunakan perintah dari CVS untuk mendapatkan salinan dari file – file tersebut secara khusus ke dalam direktori kerja, lalu bekerja dengan file di direktori tersebut. Setelah kita selesai melakukan perubahan pada file di direktori kerja kita, kita lakukan *commit* atau *checkin* kembali ke dalam repository. Repository kemudian akan berisi perubahan yang telah dilakukan, rekaman semua perubahan, waktu peubahan dan informasi lainnya. Repository bukan subdirektori dari direktori kerja kita dan juga sebaliknya, maka keduanya harus berada di lokasi yang terpisah

CVS dapat mengakses *repository* dengan berbagai cara, dapat mengakses di komputer lokal, di computer lain dalam jaringan lokal ataupun di lingkungan jaringan *World Wide Web*. Untuk memperjelas cara aksesnya nama *repository* diawali dengan *"access method"*, contoh, metode akses `':local:'` berarti mengakses direktori *repository*, `':local:c:/src/cvsroot'` artinya repository berada di komputer lokal di direktoi `'c:/src/cvsroot'`

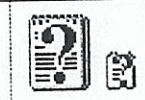
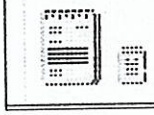
Repository dibagi menjadi dua bagian, direktoi CVSROOT berisi file – file administrative untuk CVS, sedangkan direktori lainnya berisi modul yang *user defined*

4. Windows Explorer Dan TortoiseCVS

Setelah kita *checkout* modul, kita akan mengekplorasi bagaimana TortoiseCVS bekerja dengan windows. kita akan melihat bahwa file kita

akan muncul di Windows Explorer dengan ikon yang berbeda, sebagai tambahan jika anda menggunakan Windows 2000 atau Windows XP anda juga akan melihat kolom CVS baru di *Detail View* Windows Explorer. Ikon yang terlihat menerangkan status dari file atau folder di CVS.

UNIT PELAYANAN TEKNIS PERPUSTAKAAN UNIKOM

	Unmodified	File atau folder yang belum dimodifikasi dan terhubung dengan Repository CVS
	Modified	File atau folder yang sudah dimodifikasi
	Conflict	File atau folder yang terdapat conflict di dalamnya
	Not In CVS	File atau folder yang tidak berada di bawah kontrol CVS
	CVS Watch	File atau folder yang dikontrol oleh CVS
	Ignore	File atau folder yang diabaikan oleh CVS

Gambar 3.4 Keterangan Ikon CVS

5. Update Sandbox

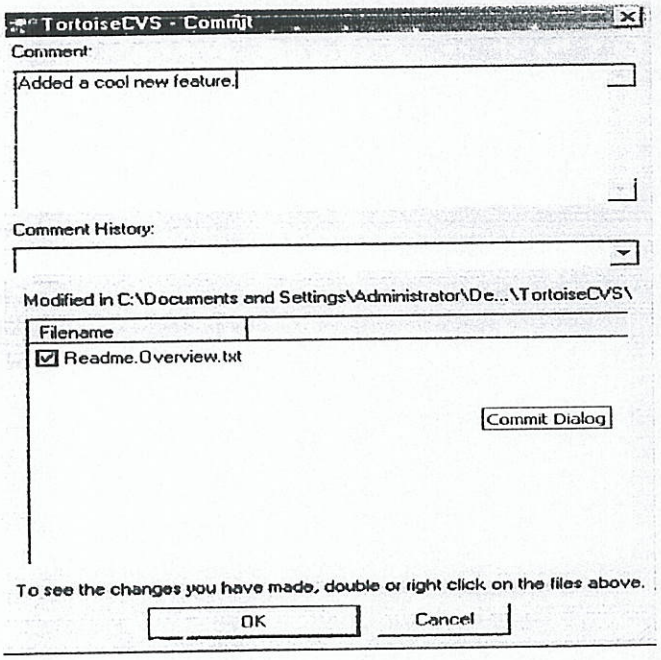
Terkadang kita ingin bekerja sama dengan orang lain di dalam direktori kerja kita. Proses perubahan dari server ke direktori kerja kita dikenal dengan istilah *Updating*. *Updating* dapat bekerja di file yang tunggal, satu set file yang dipilih, atau secara rekursif pada semua hirarki folder . Untuk meng-*update*, sorot file dan atau folder yang diinginkan,

klik kanan dan pilih **CVS Update**, Dialog proses akan muncul memperlihatkan proses *update* yang sedang berjalan. Orang lain yang melakukan *update* juga, hasilnya akan bersatu dengan file kita, kita akan menjaga setiap perubahan yang telah dilakukan terhadap file yang sama. Repository tidak akan terpengaruh oleh proses *update*, jika kita menerima laporan *conflict* selama proses *update*

6. Commit Perubahan Ke Repository

Membuat modifikasi lokal di repository dikenal dengan istilah *commit*. Sebelum *commit*, kita harus *update* untuk memastikan bahwa tidak ada *conflict*

Untuk *commit* perubahan, dimulai dengan memilih file atau folder yang ingin kita *commit*, klik kanan di file itu dan pilih **CVS Commit**



Gambar 3.5 Commit dialog

Akan muncul dialog *commit* dimana kita dapat memasukkan komentar tentang perubahan apa saja yang terjadi. Jika kita tidak ingin merubah file hasil dari proses *commit*. maka kita tidak memberi tanda cek pada *checkbox*. Setelah selesai klik **OK.** Dengan *commit* kita tidak akan otomatis menambah file baru yang telah anda buat di repository

7. Memperbaiki Conflicts

Adakalanya, server CVS akan memberikan laporan sebuah *conflict* ketika kita *update* file dari repository. *Conflict* terjadi ketika dua atau beberapa developer telah merubah sedikit baris pada file yang sama. CVS tidak mengetahui apa saja isi dari proyek kita, CVS akan menyerahkan perbaikan *conflict* ke developer. Kapanpun *conflict* dilaporkan, kita harus membuka file, dan mencari baris mana yang terjadi *conflict* dengan perintah string <<<<<<<.

Area yang telah terjadi *conflict* ditandai seperti ini :

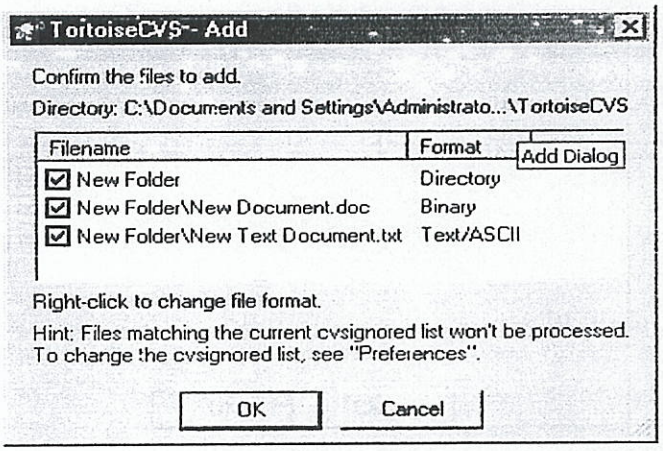
```
<<<<<<< filename
 your changes
=======
 code merged from repository
>>>>>>> revision
```

Kita harus menentukan kode mana yang harus dilihat di sini, lakukan perubahan yang diperlukan, pindahkan CVS *markup*, dan *commit* modifikasi kita ke repository

8. Menambah File dan Direktori ke Repository

Jika kita membuat file baru atau direktori, kita akan melihat bahwa file dan direktori itu muncul dengan status ikon *Not in CVS*. Untuk

memasukan file atau folder di bawah kontrol CVS, pilih item yang ingin dimasukan. Klik kanan dan pilih **CVS Add** dari menu untuk melakukan penambahan itu. Dialog *Add* akan tampil, kita dapat menverifikasi file mana saja yang ingin ditambahkan



Gambar 3.6 Add Dialog

Kita tidak perlu menentukan apakah file itu ASCII / text, Unicode / text, atau binary, karena secara otomatis TortoiseCVS mendeteksi tipe file itu

Setelah operasi penambahan ditampilkan, file atau ikon file muncul sedikit berbeda, ini terjadi karena penambahan file itu dianggap sebagai perubahan lokal dan tidak akan dikirim ke repository sampai kita melakukan *commit*

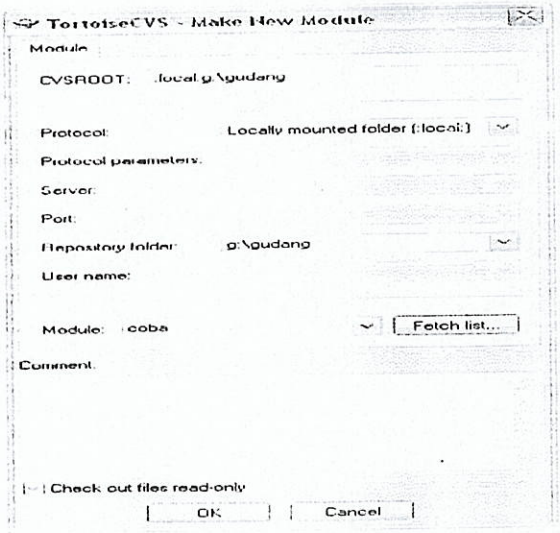
Kita dapat menambahkan beberapa file dan folder ke CVS menggunakan perintah **CVS Add All Contents.** Operasi ini dengan sendirinya akan mengubah struktur folder dan menampilkan semua file yang tidak ditambahkan di dialog *Add*, di sini kita dapat memberi tanda cek atau tidak terhadap file dan folder yang ingin ditambahkan ke CVS

3.2.6 Penggunaan lebih lanjut TortoiseCVS

1. Membuat Repository atau Modul Baru

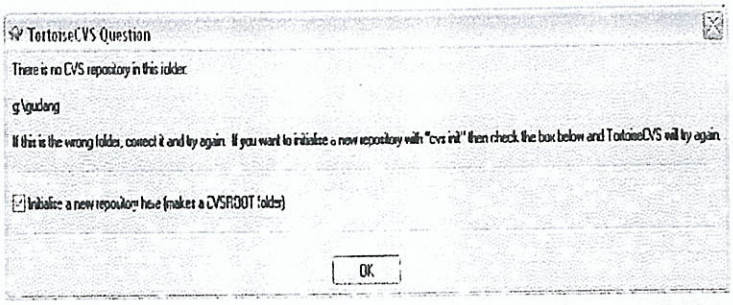
Secara khusus, seorang System Administrator akan menyusun repository CVS dalam *remote server*. TortoiseCVS dapat mengkonfigurasi repository dan modul, baik secara lokal atau *remotely*

Untuk membuat repository atau modul baru, tentukan folder dan file sebagai *sandbox*, klik kanan di folder untuk modul atau repository dan pilih **CVS --> Make New Module** dari item menu. Dialog *make new module* akan tampil. Masukan informasi detil untuk membuat repository baru atau untuk *checkout* repository CVS yang sudah ada. Lalu klik OK untuk membuat modul baru



Gambar 3.7 Make New Module Dialog

Jika remote server CVS sudah ada, tetapi repository CVS tidak ada, akan muncul pertanyaan



Gambar 3.8 Pertanyaan Repository CVS

Untuk membuat repository baru beri tanda cek pada checkbox *Initialise a new repository here* dan klik **OK** Jika kita sudah terdaftar di remote server, maka TortoiseCVS akan membuat repository baru

Jika TortoiseCVS telah membuat repository baru atau terkoneksi dengan repository CVS yang sudah ada, maka modul baru akan dibuat. Secara otomatis TortoiseCVS menggunakan nama dari folder yang dipilih untuk dijadikan nama modul yang baru, namun kita dapat menggantinya dengan merubah textbox **Module**

2. Watch, Edit dan Unedit

CVS dapat mengikuti semantic *Lock / Unlock* yang hampir digunakan oleh semua *version control system.*, di CVS dikenal dengan nama *Edit / Unedit*. Fitur ini secara otomatis belum diaktifkan oleh CVS, jika kita ingin menggunakannya kita harus mengaktifkan *Watch*

Kita dapat mengaktifkan *watch* menggunakan TortoiseCVS ketika kita membuat modul baru dengan mengaktifkan checkbox **Check files read-only.** Jika kita ingin mengaktifkan *watch* di dalam modul yang sudah ada, kita harus menjalankan beberapa perintah, (menggunakan *Command Prompt*) di bagian paling atas dari modul **CVS watch on**

Setelah *watch* diaktifkan, kita menjalankan *checkout* atau *Update* file yang kita kerjakan akan dinuat sebagai *read only*. Untuk mengubah satu atau banyak file, pilih direktori atau file yang ingin diubah. Klik kanan di file yang dipilih dan pilih **Edit** item menu. Perintah ini bekerja secara berbeda tergantung dari jenis file, text atau binary. Untuk file binary (seperti .doc), TortoiseCVS menjalankan *update* di file itu dan periksa apakah user lain sudah mengedit file itu. Jika ini terjadi kita tidak diperbolehkan untuk mengeditnya. Untuk file text (seperti .txt atau .cpp), kita dapat bekerja di file, walaupun user lain telah bekerja di file itu, dalam kasus ini TortoiseCVS akan memberitahukan keadaan itu.

Ketika kita telah selesai bekerja di sebuah file kita harus melakukan-*commit*, jika kita memutuskan untuk tidak membuat perubahan di file yang dikerjakan, klik kanan di file yang dipilih dan pilih **CVS --> Unedit** item menu. TortoiseCVS akan bertanya, apakah kita ingin mengganti file dengan versi yang orisinil (dengan catatan akan kehilangan perubahan yang telah dibuat)

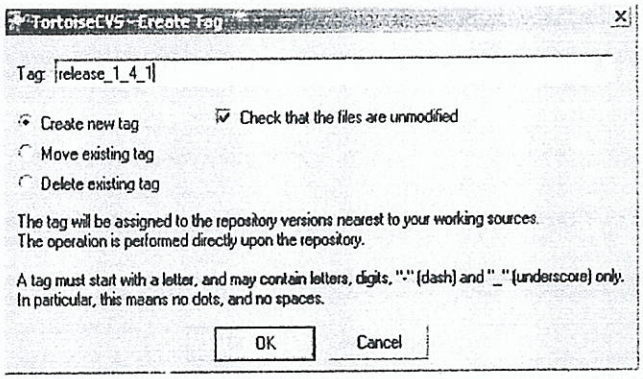
Jika kita ingin melihat file – file mana saja, yang telah diedit dan oleh siapa, klik kanan daerah mana saja di windows Explorer dan pilih **CVS --> Show edited files** item menu

3. Tagging dan Labeling

Dalam sebuah pengembangan proyek, pemberian label file untuk melihat revisinya, dikenal dengan istilah *tagging*. *Tagging* dapat digunakan di semua modul, jadi modul itu dapat direkontruksi di masa

depan. *Tagging* selalu digunakan dalam pengiriman proyek dan sebelum memulai perubahan

Untuk men-*tag* file atau direktori dengan sebuah label, pilih direktori atau file yang ingin anda *tag*, klik kanan file itu dan pilih **CVS -> Tag.** item menu



Gambar 3.9 Tag Dialog

Akan muncul dialog *tag*, di sini kita dapat memasukan label di field **Tag**. CVS membatasi karakter tertentu dari penamaan label. Label harus dimulai dengan huruf dan setelah itu harus mengandung huruf, angka, "-" (*dash* / strip), "_" (*underscore* / strip bawah), titik, sedangkan spasi tidak termasuk di dalamnya. Jika kita ingin mengikutsertakan versi nomor di sebuah label, ganti titik dengan strip. Dua nama label yang telah ada, mempunyai arti yang khusus di CVS, "HEAD" artinya versi yang baru saja diterima di repository, sedangkan "BASE" adalah revisi yang terakhir kali kita *checkout* ke dalam direktori lokal

Setelah label dimasukan, klik **OK**, tidak seperti kebanyakan operasi CVS lainnya, proses *tag* segera dilakukan di repository dan tidak membutuhkan *commit*

UNIT PELAKSANA TEKNIS PERPUSTAKAAN UNIKOM

4. Mengganti Versi lama dari File

Adakalanya Developer perlu untuk mengganti kembali perubahan yang telah di *checkin.* dan kembali lagi ke revisi awal file, caranya :

1. Klik kanan di file dan pilih **CVS --> History** untuk membuka dialog *history*
2. Klik kanan di revisi yang anda butuhkan dan pilih option "Save this revision as..."
3. Ketika dialog *Save As* muncul, jangan mengklik tombol "Save", tapi klik dua kali nama file di direktori
4. Klik "yes " ketika TortoiseCVS meminta untuk *overwrite* file

5. Branching dan Merging

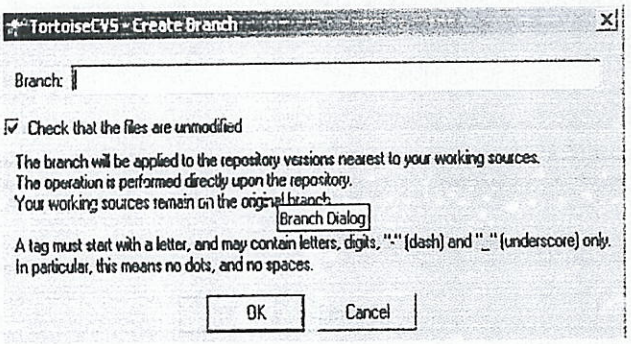
Salah satu fitur dari *version control systems* adalah kemampuan untuk mengisolasi perubahan kedalam baris yang terpisah dari suatu pengembangan. Baris ini dikenal dengan nama *Branch*

Branching sangat berguna untuk mengontrol perubahan selama peroses *lifecycle* dari suatu proyek software. Sebagai contoh kita telah membuat rilis pertama dari proyek software anda dengan versi 1.0. kita ingin menambahkan fitur baru di proyek software kita itu untuk rilis versi 2.0, diantara rilis sebelumnya dengan versi baru ditemukan *bug* di dalam software itu, revisi dari *source code* itu dilakukan secara terus - menerus

dan tidak bisa ditunda di waktu lain dan tidak ada jalan lain untuk memperbaiki *bug* berdasarkan source terbaru, daripada kita mencoba untuk memperbaiki versi itu (HEAD branch), kita iebih baik membuat *branch* di *revision tree* untuk semua file yang dibuat untuk rilis 1.0, lalu kita dapat membuat modifikasi terhadap *branch* tanpa mengganggu HEAD branch. Ketika modifikasi telah selesai anda dapat menyatukannya di HEAD branch atau memasukannya di branch rilis 1.0

1. Membuat *Branch*

Untuk membuat *branch*, pilih direktori atau file yang ingin kita *branch* dan pilih **CVS --> Create Branch**



Gambar 3.10 Branch Dialog

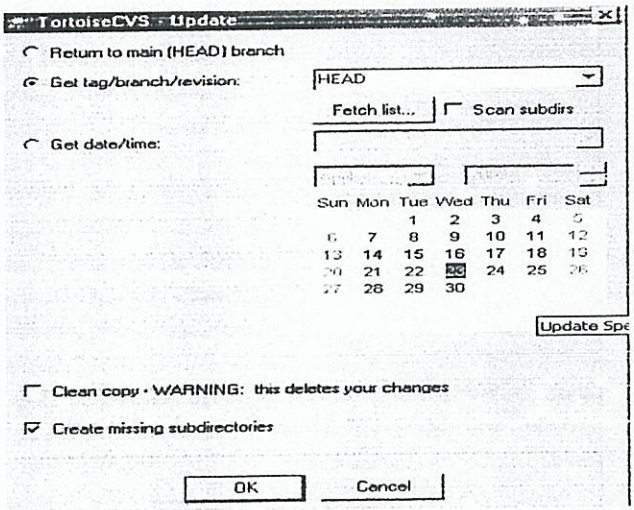
akan muncul dialog *branch*, kita dapat memasukan label di field **branch**. Pembatasan nama juga terjadi dalam *branch*, sama seperti kita menjalankan *tags*, setelah selesai memasukan nama *branch* yang diinginkan, klik **OK**

2. Memilih Branch

Untuk memulai bekerja di dalam branch, kita harus menghubungkan salinan kerja ke dalam branch. Ini diperlukan untuk

memastikan bahwa beberapa proses seperti *update, commit,* dll. dapat bekeja dengan baik di dalam branch dibandingkan di baris utama development

Untuk memindahkan salinan kerja ke *branch* yang lain, pilih folder yang paling atas dari proyek. Kita juga dapat memilih folder atau file yang merupakan bagian dari *branch* yang diinginkan. Klik kanan file yang dipilih dan pilih **CVS --> Update Special**



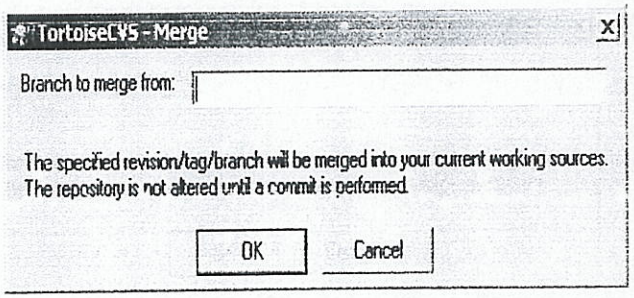
Gambar 3.11 Update Special Dialog

Kita akan melihat dialog *update special*, di sini kita dapat memasukan nama *branch* di field **Get Tag / Branch / Revision,** klik **OK** dan TortoiseCVS akan melakukan update yang diperlukan untuk memindahkan salinan kerja ke dalam branch yang diinginkan. Proses *update* juga termasuk menambah atau memindahkan file tergantung status dari *branch.* TortoiseCVS menempatkan label terhadap file yang dikenal juga *sticky tags*, yang berpengaruh pada *branch.* Untuk memindahkan *sticky tags* kita harus kembali ke HEAD *branch*

3. *Merging* Dari Sebuah *Branch*

Ketika kita sudah puas dengan perubahan yang dibuat, kita ingin perubahan itu juga dapat terjadi pada HEAD *branch* dari suatu development yang terintegrasi dari satu *branch* ke *branch* yang lain dan dikenal juga dengan nama *merging*

Untuk melakukan *merge* dari *branch*, pindahkan salinan kerja kita, ke *branch* yang ingin kita *merge*. Pilih folder yang paling atas dari proyek Kita juga dapat memilih folder atau file yang merupakan bagian dari *branch* yang diinginkan. Klik kanan file yang dipilih dan pilih **CVS --> Merge**



Gambar 3.12 Merge Dialog

Kita akan melihat dialog *merge*, di sini kita dapat memasukan nama *branch* di field **Branch to merge from,** lalu klik **OK**. TortoiseCVS akan melakukan proses *merge* dari *branch*. Perubahan yang terjadi tidak akan dapat dibuat di server repository, sampai kita meng-*commit* perubahan itu

Proses *merge* di atas akan mencoba merubah dari permulaan *branch*, jika kita mengerjakan operasi untuk kedua kalinya (untuk me-*merge* kembali *branch* setelah *merge* terakhir), kita tidak menginginkan

merging yang dimulai dari permulaan *branch* dan itu akan memberikan masalah bagi kita, untuk mengatasi masalah ini, kita harus memberikan label baru setelah kita me-*merge* dan menggunakan label baru ketika memberi nama *branch* untuk proses *merge* berikutnya

4. Kembali Ke HEAD Branch

Jika kita imgin berhenti untuk bekerja pada *branch* dan memindahkan kembali salinan kerja ke baris utama development, kita harus membuat TortoiseCVS memindahkan semua *sticky tags*

Untuk memindahkan *sticky tags*, dan *update working copy*, kita kembali ke baris utama development, pilih folder yang paling atas dari proyek. Kita dapat memilih folder atau file yang merupakan bagian dari *branch* yang diinginkan. Klik kanan file yang dipilih dan piiih **CVS --> Update Special**

Kita akan melihat dialog *update special*, pilih checkbox *Return to main (HEAD) branch*, klik **OK** dan TortoiseCVS akan melakukan *update* yang diperlukan untuk memindahkan salinan kerja ke dalam *head branch*

6. Binary Dan Unicode

TortoiseCVS secara otomatis mendeteksi tipe file yang dimasukan ke dalam CVS, CVS dapat mendeteksi tipe file *Text / ASCII*, *Text / Unicode*, atau *Binary*. TortoiseCVS pertama akan mengecek ekstensi dari file untuk memisahkan apakah file itu termasuk text atau binary. Sebagai contoh ".doc" dan ".exe", termasuk tipe file binary sedangkan ekstensi lainnya seperti ".cpp" dan ".txt", termasuk tipe text. TortoiseCVS

menjalankannya menggunakan daftar yang sudah ada dari ekstensi file binary dan text yang termasuk tipe file yang umum. Untuk mengatur daftar ini kita dapat mengubah file "FileTypes.config" di direktori instalasi TortoiseCVS

Secara otomatis TortoiseCVS akan mengecek 4000 bytes pertama dari file untuk menentukan tipe file, ini digunakan jika ekstensi file tersebut tidak diketahui, 4000 bytes pertama itu juga digunakan untuk menentukan apakah file itu termasuk tipe file Text / ASCII atau Text / Unicode.

TortoiseCVS juga menyediakan *plugin* untuk mengimplementasikan DLL untuk mendeteksi tipe file. Keterangan dari *plugin* ini juga ada di file "FileTypes.config":

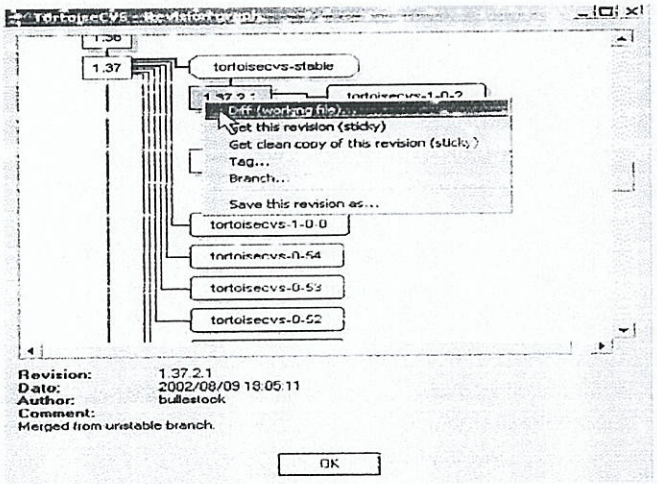
Dalam semua kasus, dialog *Add* memperlihatkan file yang ingin dimasukan berdasarkan tipenya, jadi akan mempermudah kita

7. File Revision History

TortoiseCVS meyediakan dua metode untuk me-*review* dan bekerja dengan *history* dari file. Pertama dari dialog *history*, kita dapat melihat *history* dari file dengan cara klik kanan file tersebut dan pilih **CVS --> History**

Dialog *history* akan memperlihatkan catatan revisi, *tags* dan *branch* dari file itu, dengan mengklik kanan revisi yang lebih spesifik, *tag*, atau *branch* untuk men-*diff*, men-*save* revisi atau *update* dengan *sticky tags*

TortoiseCVS juga meyediakan dialog *Revision Graph* yang secara grafis memperlihatkan *history* dari file, kita dapat melihat *revisi graph* dari file dengan cara klik kanan pada file dan pilih **CVS –> Revision Graph**



Gambar 3.13 Revision Graph Dialog

Seperti dialog *history*, dialog *Revision Graph* juga akan menampilkan keterangan dan operasi CVS pada setiap revisi, *tag* atau *branch*

8. Making a Patch File

Mungkin kita mempunyai akses membaca ke repository atau modul CVS, tetapi tidak memiliki akses menulis. Proyek *open source* secara khusus bekerja dengan cara ini untuk mengelola *quality control* untuk semua kontribusi terhadap proyek, jalan termudah untuk menambah kontribusi terhadap proyek adalah dengan membuat file *patch*

Di TortoiseCVS proses ini dilakukan dengan cara memilih folder atau file dengan perubahan yang ingin kita tambahkan ke file *patch*. Klik kanan pada file dan pilih **CVS --> Make Patch** *menu*. Kita akan melihat

dialog *save* untuk menerangkan di mana kita harus menyimpan file *patch*. TortoiseCVS akan membuka patch baru di notepad, dimana kita dapat memvalidasi perubahan.

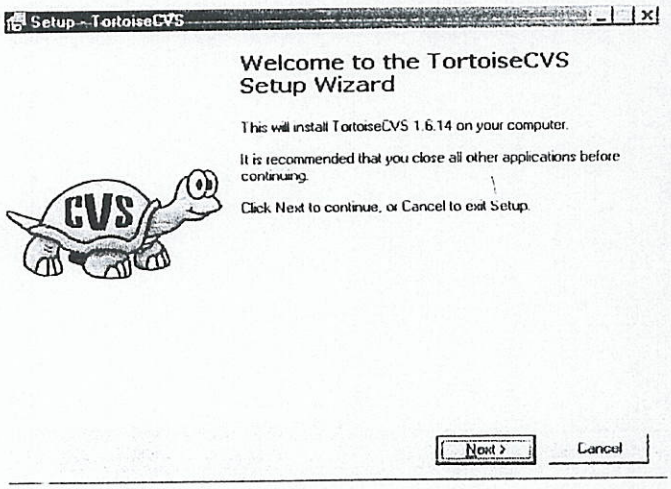
UNIT PELAKSANA TEKNIS PERPUSTAKAAN UNIKOM

3.2.7 Contoh Penggunaan CVS Memakai TortoiseCVS Yang Standalone (Localhost Server)

Kita akan mengerjakan suatu proyek pembuatan biodata penulis, menggunakan file .txt dari Notepad, caranya sebagai berikut :

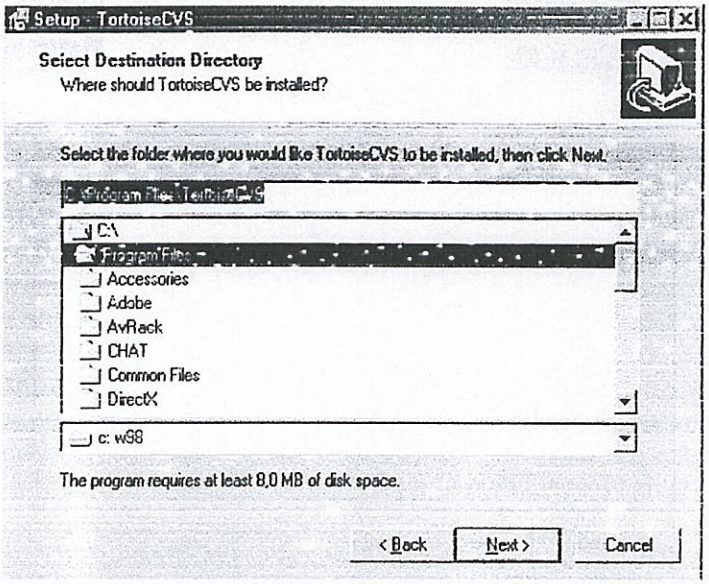
A. Menginstall TortoiseCVS

1. Download versi terbaru dari TortoiseCVS dari web site (http://tortoisecvs.org/)
2. Jalankan file *installer*. Jika kita memakai Windows NT atau 2000 kita harus mempunyai hak istimewa *local admin* untuk menginstall TortoiseCVS, muncul *Setup Wizard* TortoiseCVS



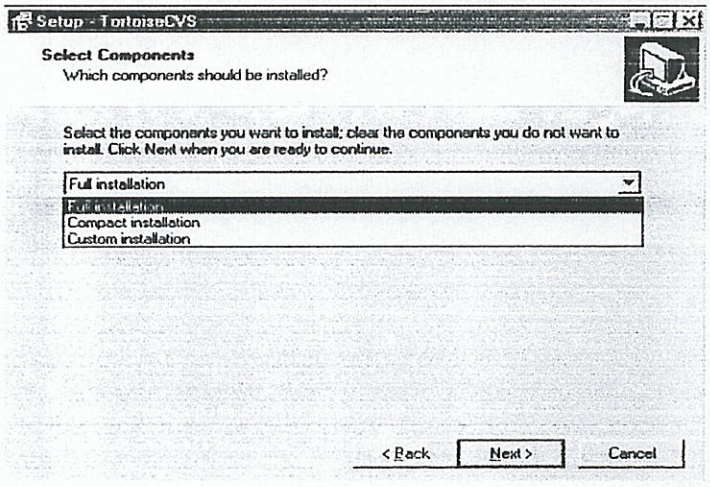
Gambar 3.14 Setup Wizard TortoiseCVS

3. Pilih lokasi di mana TortoiseCVS akan diinstall. Secara otomatis akan berada di "C:\Program Files\TortoiseCVS".



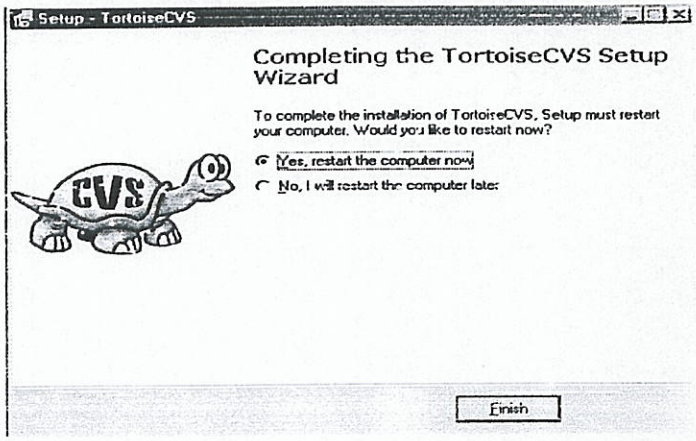
Gambar 3.15 Lokasi Instalasi TortoiseCVS

4. Pilih tipe installasi, apakah "Full", "Compact", stau "Custom installation".



Gambar 3.16 Tipe Instalasi TortoiseCVS

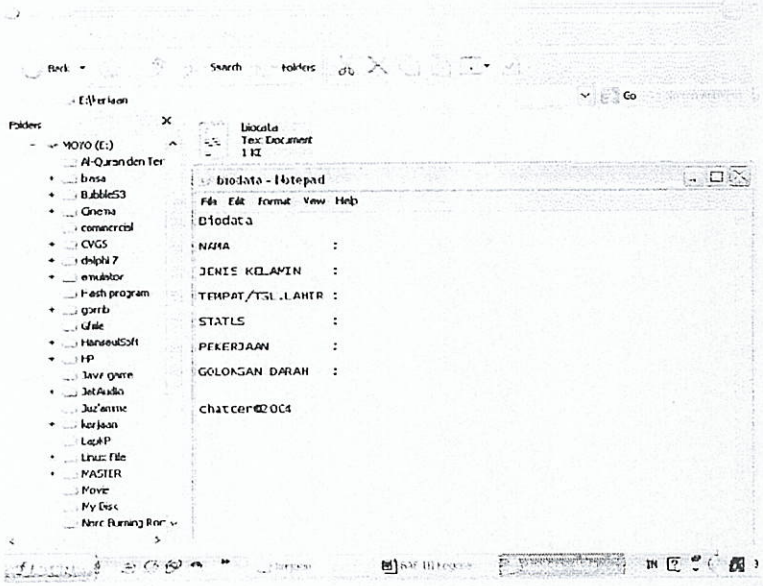
5. Setelah proses instalasi selesai kita harus merestart komputer agar instalasi sempurna



Gambar 3.17 Instalasi Selesai

B. Membuat Modul Baru

1. Buat suatu folder di drive manapun (penulis membuat nama folder 'kerjaan' di E:\MOYO), buat file biodata.txt di notepad yang isinya merupakan versi awal dari biodata, simpan di folder 'kerjaan'



Gambar 3.18 File Biodata.txt

2. Klik kanan di folder 'kerjaan' pilih **CVS > Make New Module..** kemudian muncul dialog *make new modul*, dari sini kita dapat masuk ke CVSROOT repository di dalam field **CVSROOT** , atau kita buat

CVSROOT dengan menggunakan field **Protocol**, **Server**, **Port**, **Repository name**, **User name** dan **Module**, isi field – field tersebut dengan isian :

CVSROOT : (tempat dimana CVSROOT diakses)

Protocol : Locally mounted folder (:local:)

Protocol parameters : kosongkan (karena kita memakai server localhost)

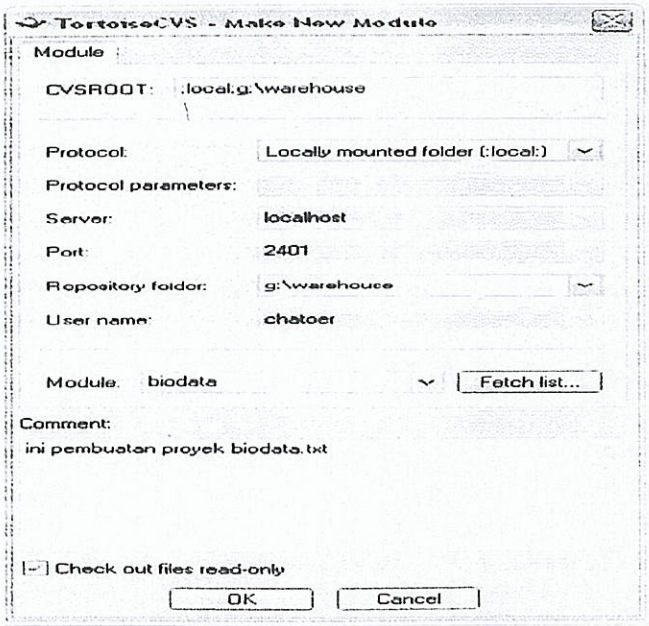
Server : localhost

Repository folder : g:\warehouse (lokasi folder repository)

User name : chatoer (merupakan *user name* dari admin)

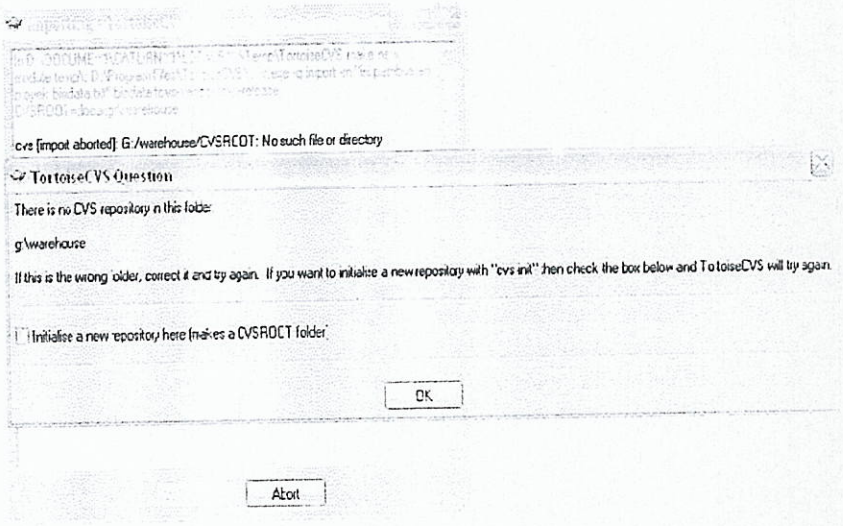
Module : biodata (nama modul, bila anda tidak mengetahui nama modul klik fetch list)

Comment : (komentar tentang modul) Setelah semua selesai diisi klik **OK**



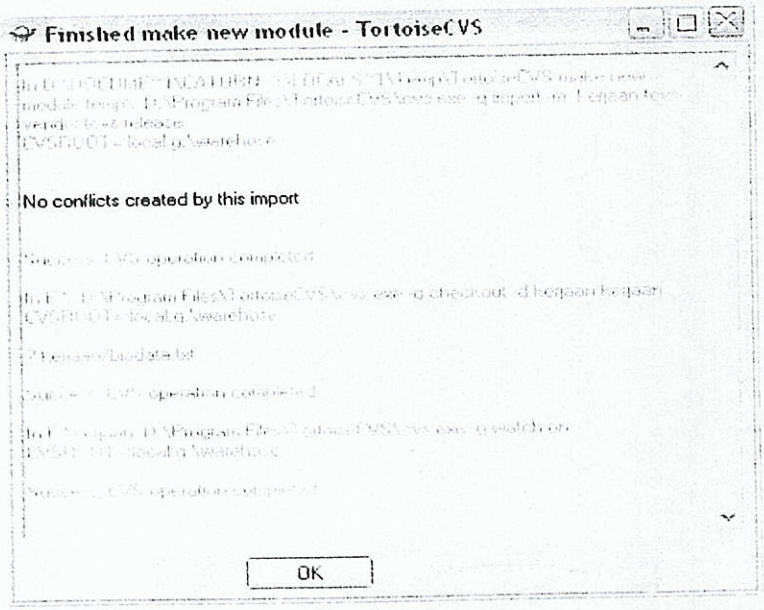
Gambar 3.19 Make New Module

3. Muncul peringatan bahwa folder Repository belum ada, untuk membuat folder Repository beri tanda cek di pernyataan *Initialize a new repository here (makes a CVSROOT folder)* lalu klik **OK**



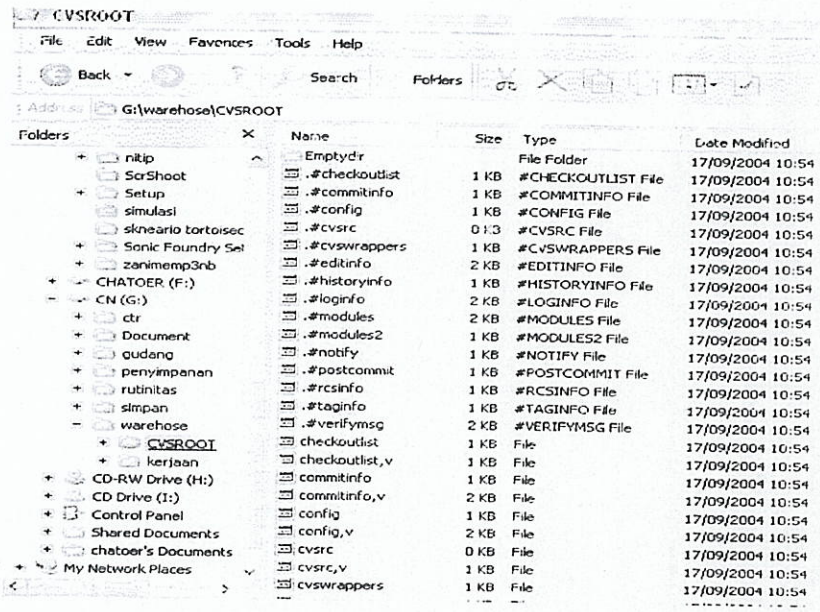
Gambar 3.20 Peringatan Repository

bila *progress dialog* menyatakan *Success* berarti folder repository telah berhasil dibuat dan klik **OK** di *progress dialog*



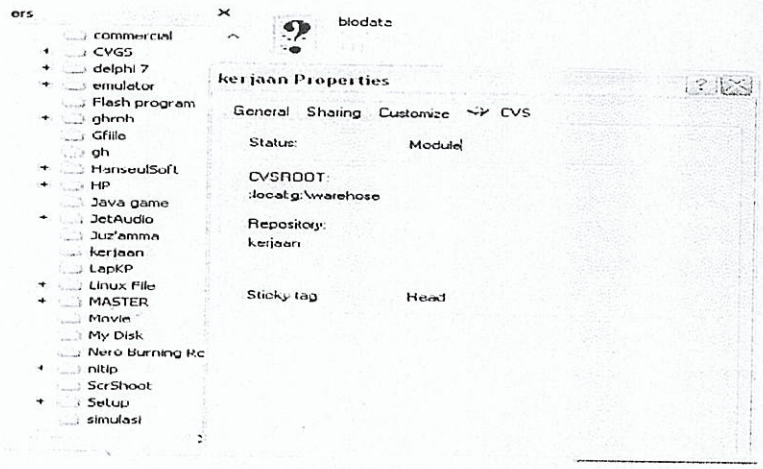
Gambar 3.21 Progress Dialog Finished Maka New Module

4. Secara otomatis folder repository akan terbentuk dan dapat dilihat di drive G:\warehouse. Repository berisi informasi – informasi penting dari modul yang kita kerjakan



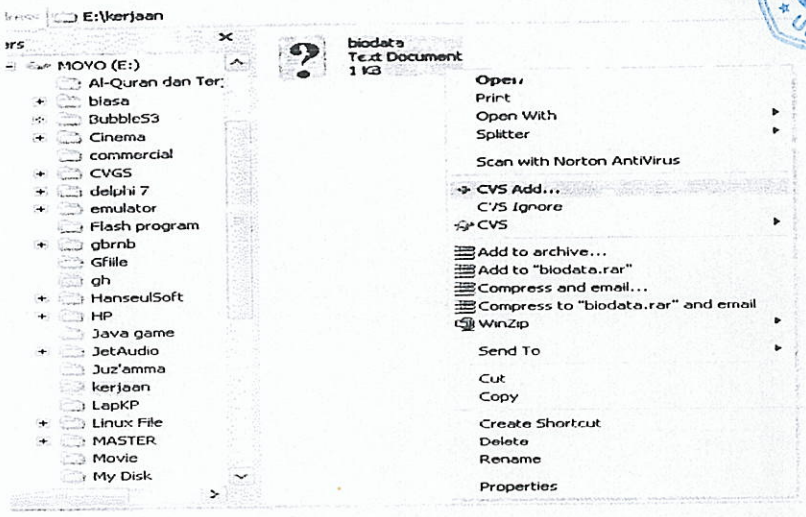
Gambar 3.22 Repository

5. Buka kembali Windows Explorer, disini terlihat direktori kerja (*sanbox*) folder 'kerjaan' ikonnya berubah, karena folder ini telah berada dalam kontrol CVS, sedangkan modul biodata.txt ikonnya juga berubah, yang artinya modul itu belum berada dibawah kontrol CVS



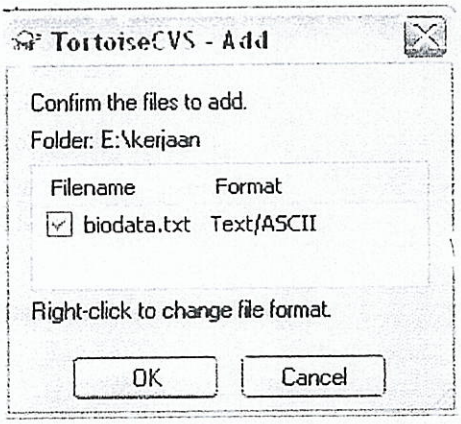
Gambar 3.23 File Tidak Dalam Kontrol CVS

6. Agar modul berada di bawah kontrol CVS, klik kanan pada file biodata.txt, pilih **CVS Add...**



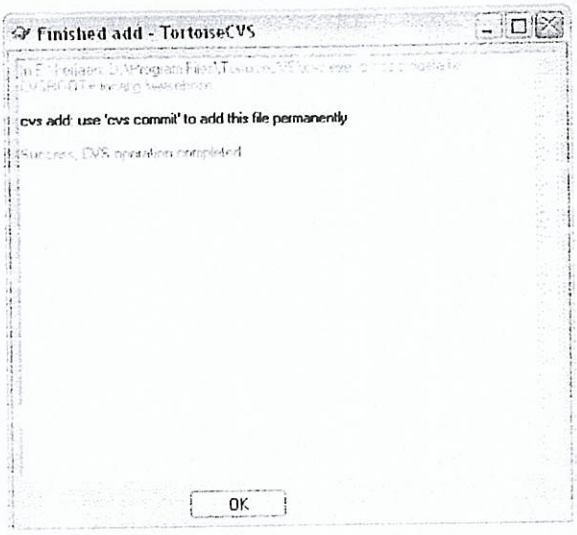
Gambar 3.24 CVS add

lalu akan muncul dialog *CVS add*, beri tanda cek pada file yang akan dimasukan lalu klik **OK**



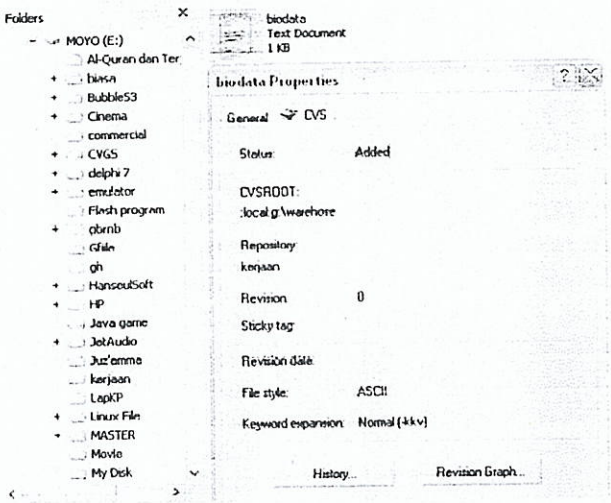
Gambar 3.25 Add Dialog

lalu muncul *progress dialog finished Add*, lalu klik **OK**



Gambar 3.26 Progress Dialog Finished Add

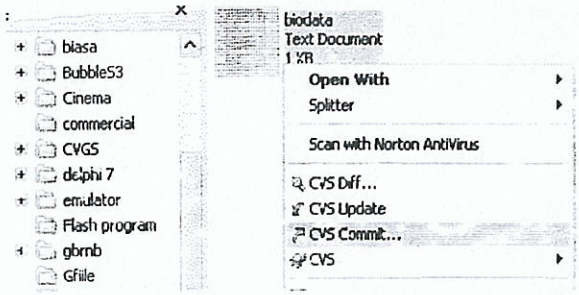
maka ikon file biodata.txt akan berubah, begitu pula propertiesnya



Gambar 3.27 File CVS

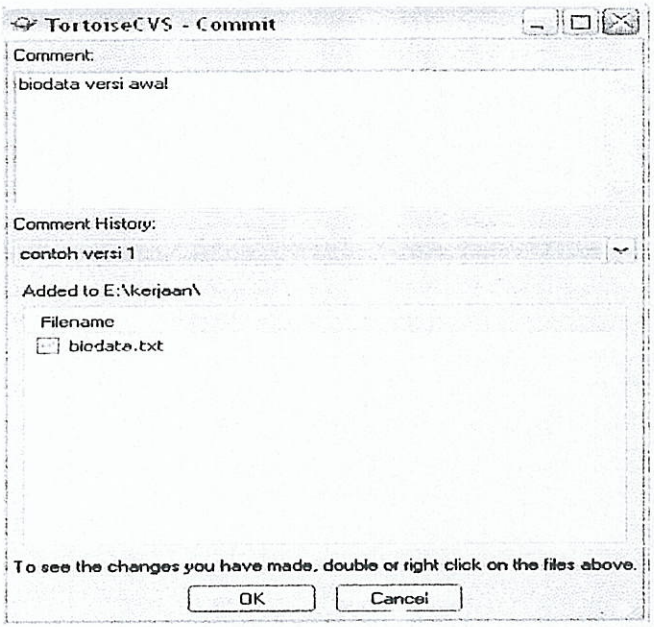
C. Commit Modul

1. Sebelum versi awal disimpan di repository, modul harus di *commit* terlebih dahulu dengan cara, klik kanan di file biodata.txt pilih **CVS Commit...**



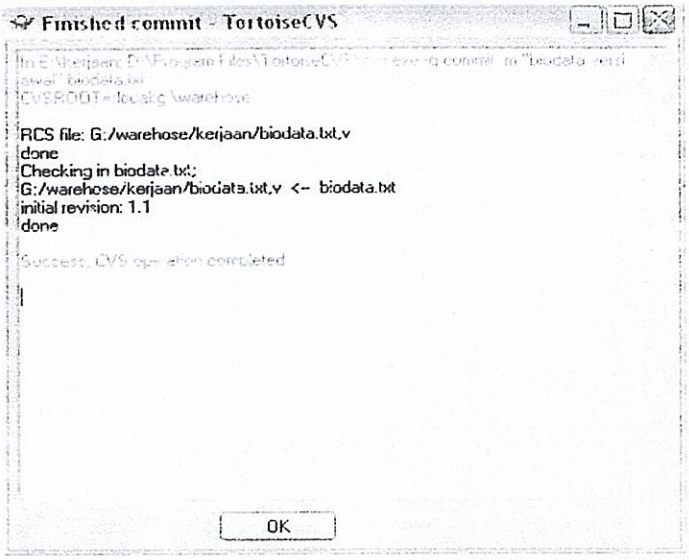
Gambar 3.28 CVS Commit

akan muncul *commit dialog*, masukan komentar dari versi ini di field *comment*, atau pilih komentar yang telah ada di field *comment history*, lalu klik **OK**



Gambar 3.29 CVS Commit Dialog

muncul *progress dialog untuk finished commit*, lalu klik **OK**



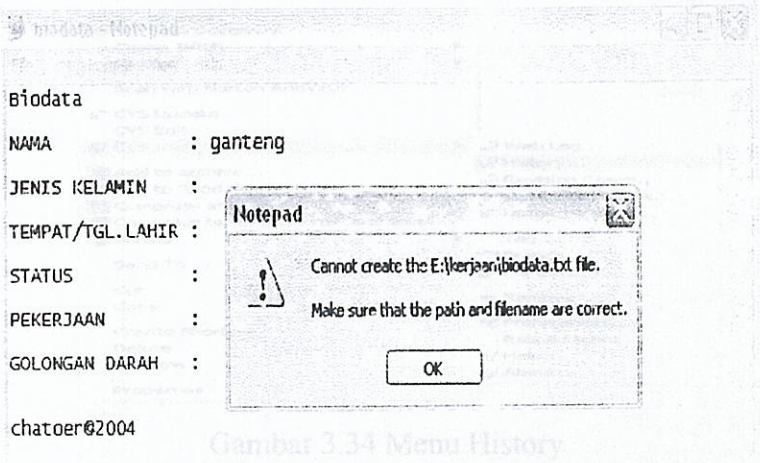
Gambar 3.30 Progress Dialog Finished Commit

2. Lihat kembali Windows Explorer, ikon file biodata.txt berubah, ini artinya file tersebut telah di-*commit* ke repository dan tidak bisa diubah,



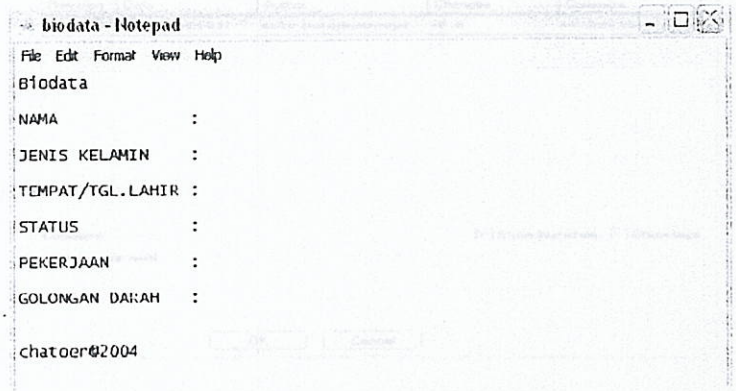
Gambar 3.31 File Biodata.txt

jika kita bersikeras untuk mengubah isi file biodata dan menyimpannya maka akan muncul peringatan, dan setelah kita mengklik **OK**, notepad akan meminta nama file baru, (seperti option Save As) dan akan menjadi file text biasa yang tidak berada dalam kontrol CVS



Gambar 3.32 Peringatan file Biodata,txt

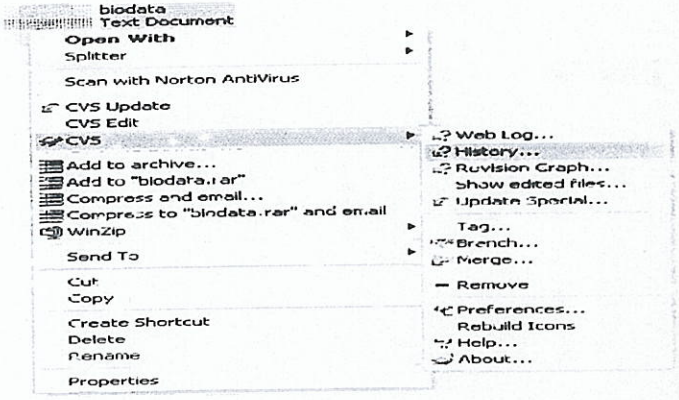
jadi file biodata.txt ini merupakan versi awal dari proyek biodata.



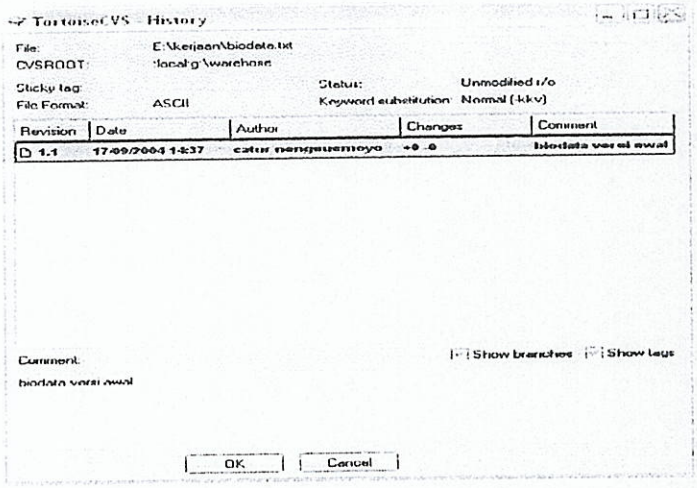
Gambar 3.33 Biodata,txt Versi Pertama

D. Melihat History File

1. Jika kita ingin melihat history dari file biodata.txt, klik kanan pada file biodata.txt, pilih **CVS > History**, lalu muncul *history dialog* yang memperlihatkan semua keterangan tentang versi awal file biodata.txt



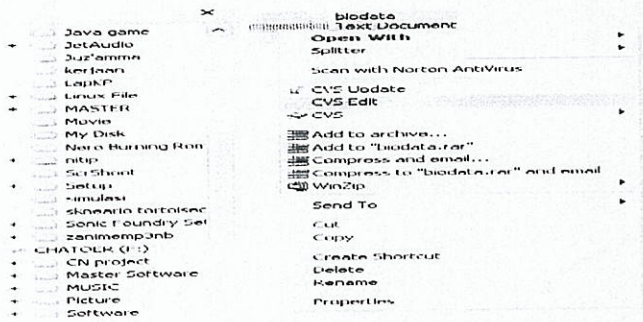
Gambar 3.34 Menu History



gambar 3.35 History Dialog

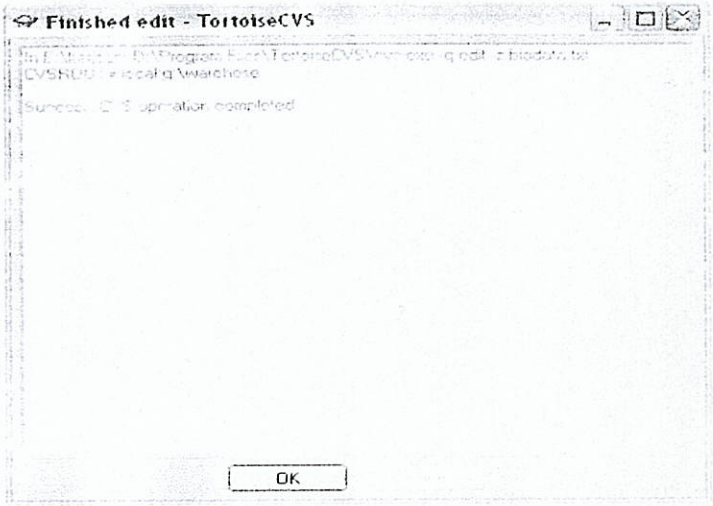
E. Membuat Versi Kedua

1. Jika kita ingin mengubah atau membuat versi kedua dari file biodata.txt, klik kanan pada file biodata.txt, pilih **CVS Edit..**



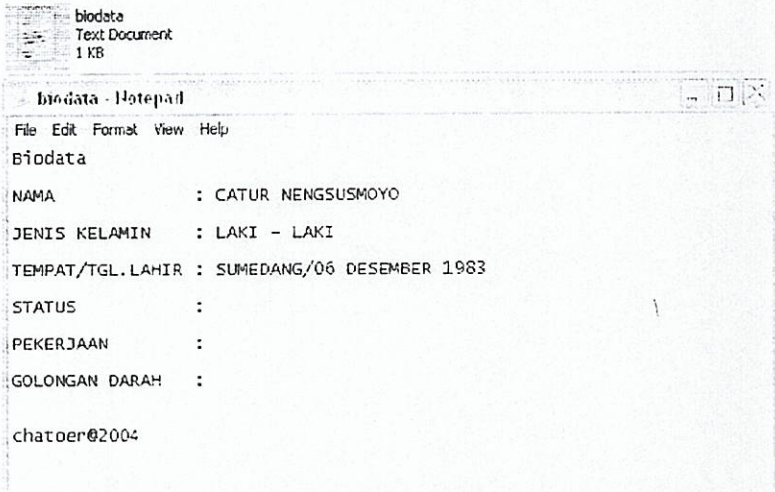
Gambar 3.36 Menu CVS Edit

muncul *progress dialog finished edit*



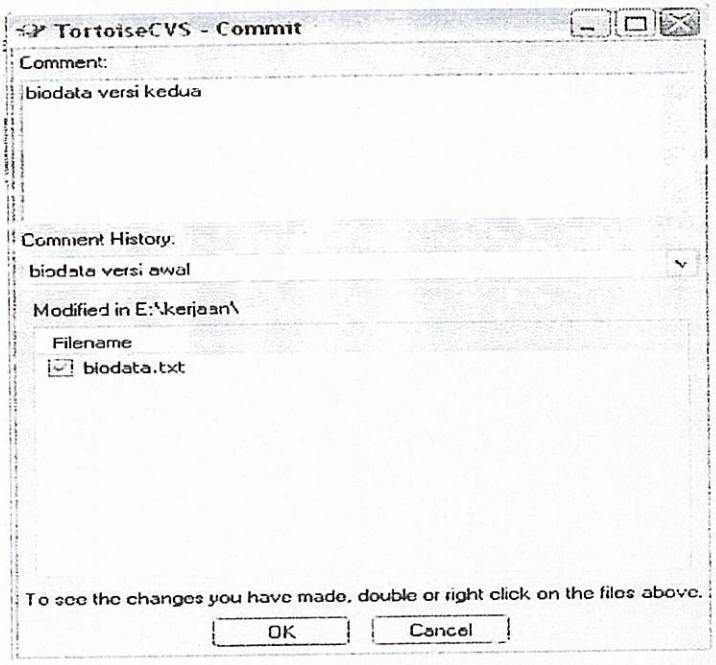
Gambar 3.37 Progress Dialog Finished Edit

maka kita dapat merubah file biodata.txt di notepad, setelah selesai simpan perubahan tersebut, file biodata.txt menjadi



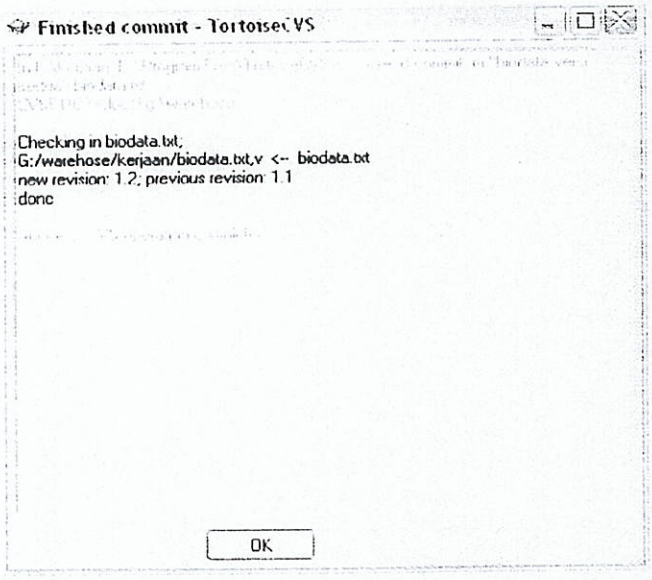
Gambar 3.38 Biodata.txt Versi Kedua

2. Lalu kita *commit* kembali file biodata.txt, caranya sama seperti di atas, kita isi komentar untuk versi kedua, lalu klik **OK**



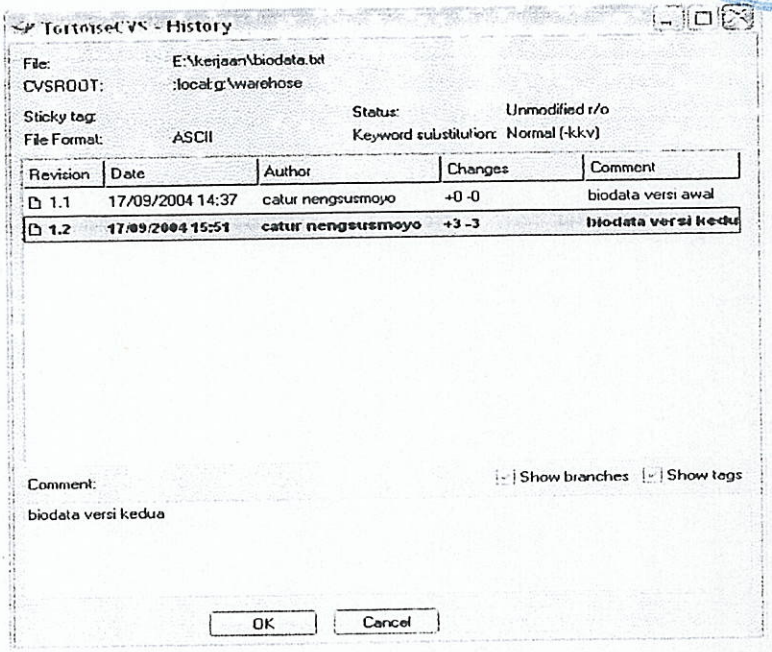
Gambar 3.39 CVS Commit

muncul *progress dialog finished commit* disini terlihat informasi bahwa ada revisi baru dari file biodata.txt, lalu klik **OK**



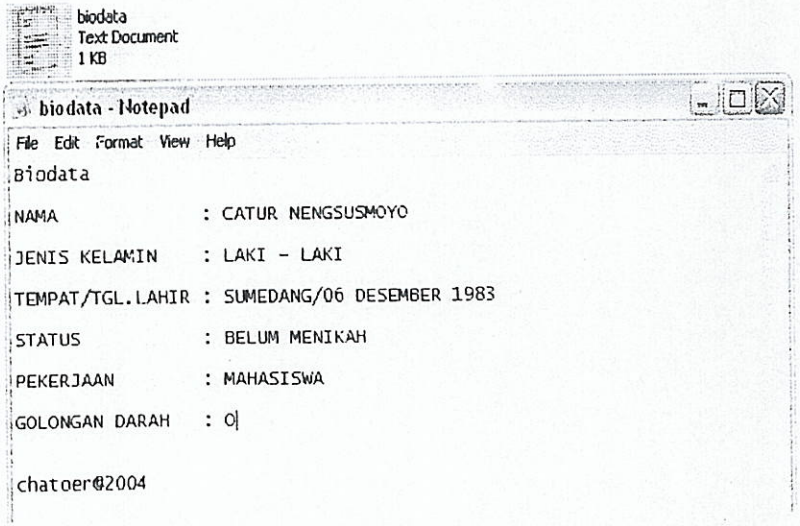
Gambar 3.40 Progress Dialog Finished Commit

3. Kita ingin melihat *history* yang baru dari file biodata.txt caranya sama seperti diatas. Terlihat bahwa kita sudah mempunyai dua revisi dari file biodata.txt



Gambar 3.41 History

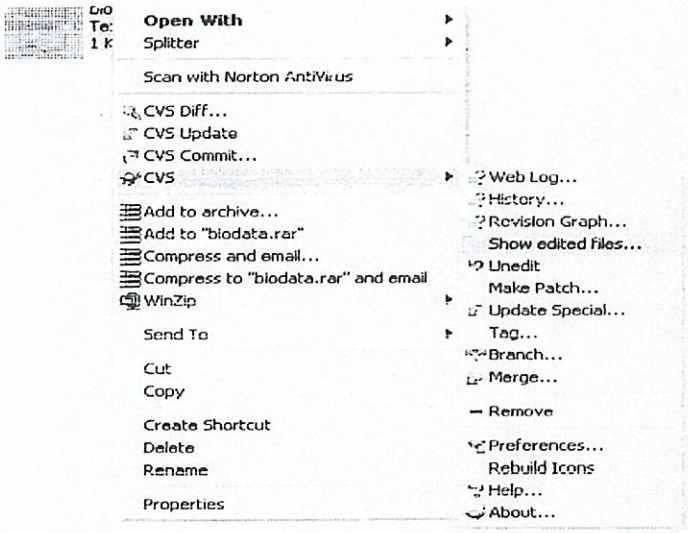
kita akan membuat revisi ketiga, caranya sama ketika kita akan membuat revisi kedua.



Gambar 3.42 Biodata.txt Versi Ketiga

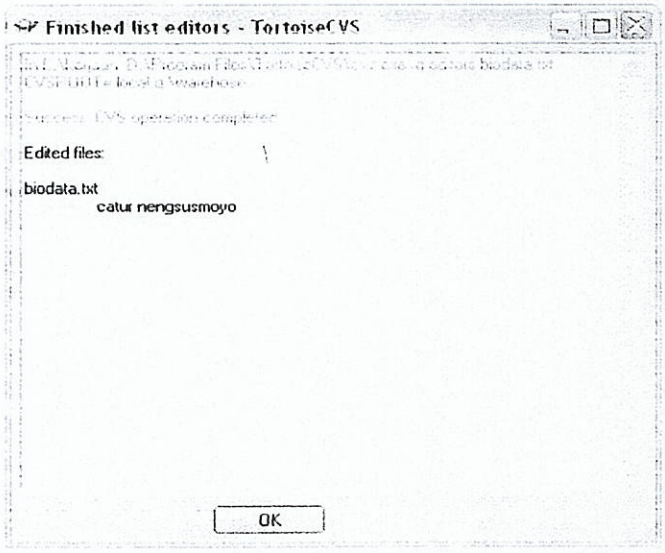
F. Melihat File Yang Diedit

1. Kita ingin melihat file apa saja yang telah diedit, maka sebelum kita meng-*commit* kembali, klik kanan file biodata.txt pilih **CVS > Show Edited Files..**



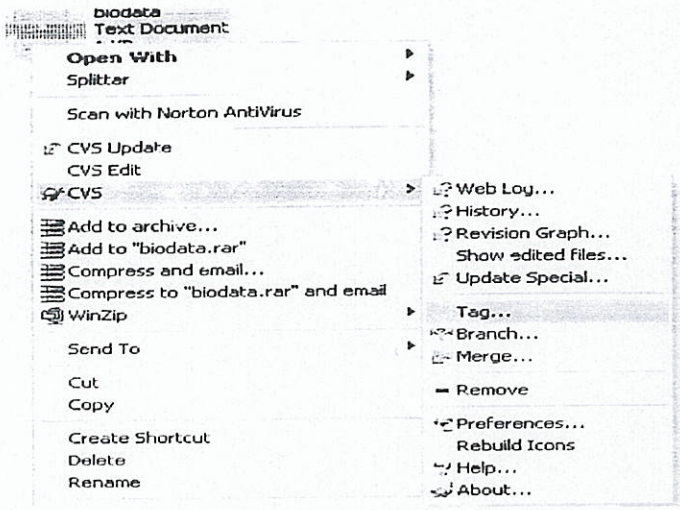
Gambar 3.43 Menu Edited File

muncul *progress dialog finished editor*



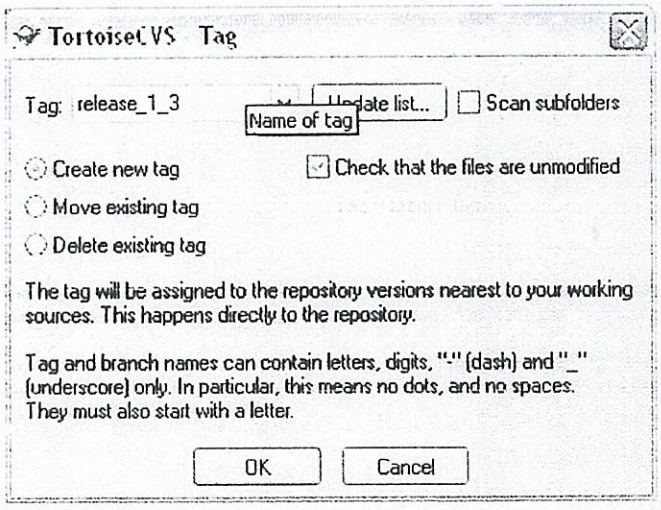
Gambar 3.44 Progress Dialog Finished List Editor

2. Kita ingin membuat *tag* (label) dari file biodata.txt, caranya klik kanan file biodata.txt pilih **CVS > Tag..**



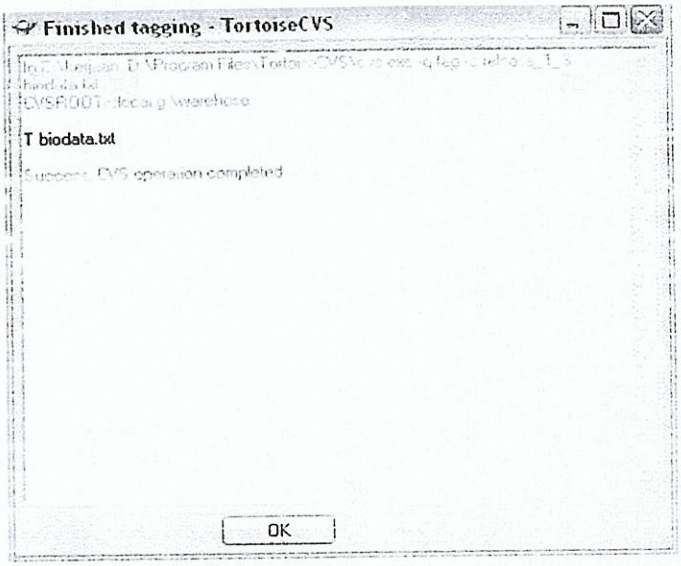
Gambar 3.45 Menu CVS Tag

muncul dialog tag, isi nama tag sesuai dengan yang kita inginkan lalu klik **OK**



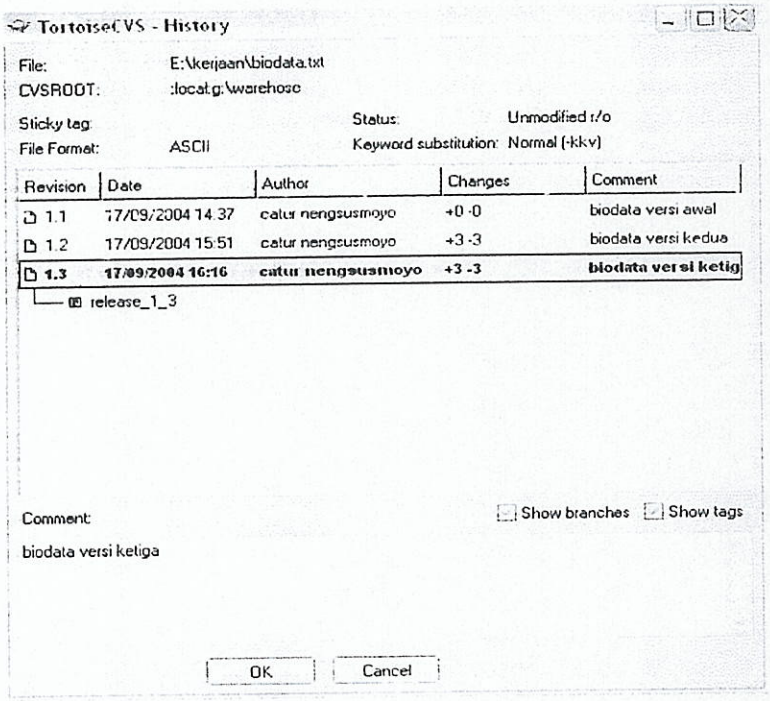
Gambar 3.46 CVS Tag Dialog

lalu *progress dialog finished tagging* akan muncul, klik **OK**



Gambar 3.47 Progress Dialog Finished Tagging

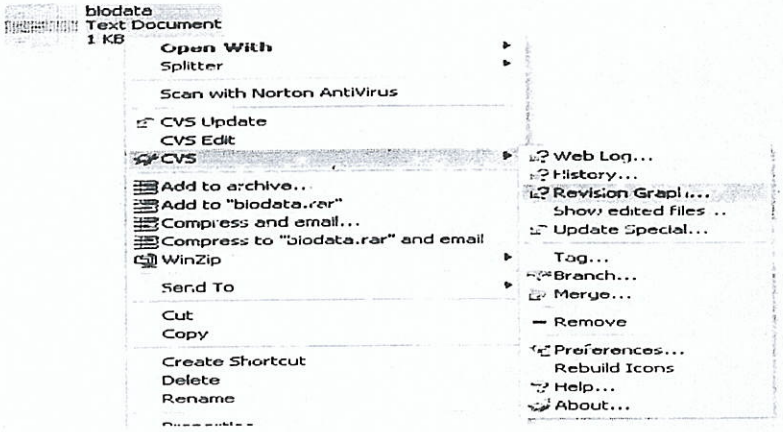
3. Untuk melihat tag kita buat, pilih **CVS > History..**



Gambar 3.48 History

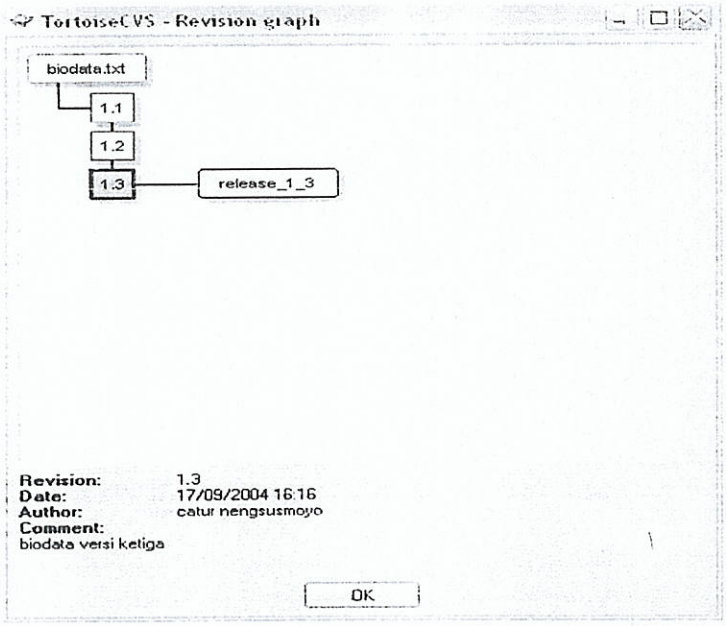
G. Untuk Melihat Grafik Semua Revisi File Yang Telah Dibuat

1. Klik kanan file biodata.txt, pilih **CVS > Revision Graph..**



Gambar 3.49 Menu Revision Graph

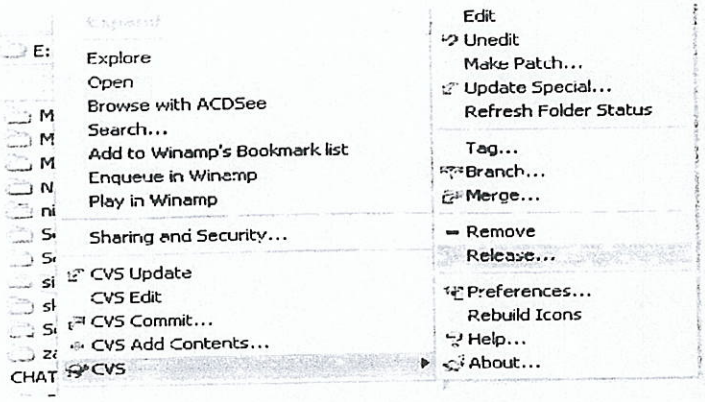
muncul *revision graph dialog*



Gambar 3.50 Revision Graph

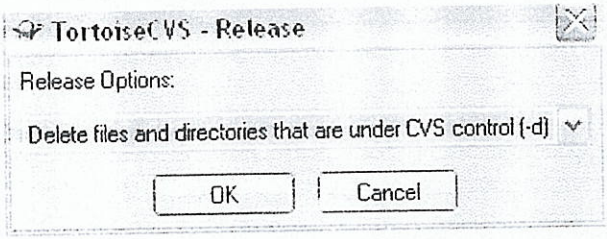
H. Mengeluarkan File Agar Tidak Dikontrol Oleh CVS

1. Klik kanan pada direktori kerja *(sanbox)* kita, pilih **CVS > Release..**



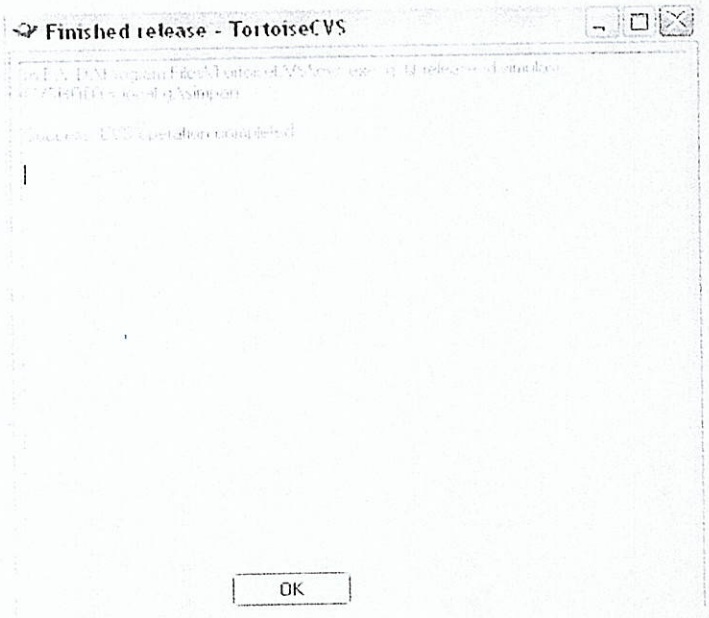
Gambar 3.51 Menu Release

muncul *dialog release*, klik **OK**



Gambar 3.52 Release Dialog

2. Setelah *progress dialog Finished release* muncul maka file tidak lagi di bawah kontrol CVS



Gambar 3.53 Progress Dialog Finished Release