

Version Control Menggunakan Subversion dan TortoiseSVN

Ifnu Bima Fatkhan

www.nagasakti.or.id/roller/ifnu

Versi 0.1

Januari 2007

Version Control

Version control dikenal dengan banyak istilah. Ada yang menyebutnya sebagai Configuration Management Tool, Source code management atau Source Control. Dalam modul ini, istilah yang digunakan adalah Version Control.

Kemampuan menggunakan version control merupakan hal yang masih langka ditemukan di kalangan programmer Indonesia. Tidak banyak buku, tutorial ataupun tempat pelatihan yang menyediakan bahan pembelajaran dalam bahasa Indonesia. Bahkan di banyak perusahaan software consultant, version control ini masih jarang digunakan.

Padahal kemampuan menggunakan version control adalah kemampuan wajib yang harus dimiliki oleh tim programmer. Di seluruh dunia, ribuan programmer terlibat dalam pengembangan proyek opensource, kolaborasi dalam skala raksasa seperti ini mustahil dilaksanakan tanpa adanya version control. Akan terjadi bencana dan kekacauan jika kode sumber hanya diletakkan di website, kemudian dibagi ke semua programmer. Jika dua orang programmer bekerja dalam file yang sama, mereka akan saling menimpa pekerjaan temannya, dan jika terjadi kesalahan maka tidak akan mungkin mengembalikan keadaan kode ke keadaan sebelumnya tanpa usaha yang cukup besar.

Version control dapat membantu sebuah 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. Version control mampu :

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.

Jumlah maksimal programmer yang dapat bekerja sama tanpa menggunakan version control adalah satu orang (Endy, 2006). Jika dalam sebuah tim pengembang software terdapat sebuah tim programmer yang lebih dari satu, version control adalah tools yang wajib digunakan.

Ada banyak aplikasi version control yang tersedia, beberapa aplikasi yang cukup terkenal antara lain :

1. Visual Source Safe, version control buatan microsoft. Terintegrasi dengan Visual Studio
2. CVS, version control dikembangkan oleh CollabNet yang dipimpin oleh Karl Fogel. CVS adalah version control yang dikembangkan dengan lisensi opensource dan sudah sangat banyak digunakan. CVS juga mempunyai software pendukung yang sangat besar, dan juga didukung oleh hampir semua IDE besar.
3. Subversion, dibuat dengan tujuan untuk menggantikan CVS yang mempunyai kelemahan-kelemahan sehingga tidak cocok lagi dengan paradigma pengembangan perangkat lunak yang sedang trend saat ini. Subversion ini dikembangkan oleh CollabNet dan dibuat oleh orang yang sama yang membuat CVS, Karl fogel.

Penggunaan Version Control

Di bagian sebelumnya sudah dijelaskan apa itu version control, untuk lebih jelasnya bagaimana skenario version control digunakan, simak cerita berikut.

Ifnu dan Dhiku terlibat proyek pengembangan sebuah aplikasi untuk menampilkan data berupa text dan gambar dari dalam database. Software ini dikembangkan atas permintaan sebuah stasiun televisi Oke Banged. Ifnu dan Dhiku berdomisili di tempat yang berbeda dan masing-masing mempunyai kesibukan yang tidak dapat ditinggalkan pada jam kerja. Sehingga mereka mempunyai waktu yang sangat terbatas untuk bertatap muka secara langsung.

Dhiku menginstal version control server dan membuat repository untuk program tersebut di rumahnya yang terhubung ke internet 24 jam sehari, 7 hari seminggu. Semua kode program yang dibutuhkan disimpan dalam repository dan dapat diakses oleh mereka berdua.

Pengembangan aplikasi ini hanya membutuhkan waktu seminggu, karena ukuran aplikasinya yang tidak terlalu besar dan sederhana. Selama waktu tersebut keduanya silih berganti menambahkan kode ke dalam repository. Mengerjakan bagiannya masing-masing dan melakukan test terhadap setiap fungsi yang ditambahkan.

Setelah waktu yang ditentukan habis, Dhiku menginstal aplikasi di stasiun televisi Oke Banged. Semua kode program terbaru diambil dari repository dan dimasukkan ke dalam laptop untuk selanjutnya

dibawa ke tempat presentasi. Setelah tiba di kantor stasiun televisi Oke Banged, kode program tersebut mengalami error serius ketika dikompilasi!! Dhiku tidak panik, dengan tenang segera melakukan koneksi ke internet menggunakan CDMA modem. Kemudian mengakses repository server dan melihat catatan perubahan kode program. Ternyata Ifnu menambahkan satu bonus fitur baru untuk melihat gambar dalam tampilan thumbnail yang dikerjakan semalam, pasti ini penyebabnya, karena hari sebelumnya Dhiku sudah memastikan bahwa semua fungsi program berjalan dengan baik dan siap untuk diimplementasikan.

Untungnya Ifnu tidak lupa untuk memberikan penanda (tag) terhadap perubahan yang dilakukannya semalam. Dengan cepat Dhiku mengambil kode program sesuai kondisi sebelum perubahan yang dibuat oleh Ifnu semalam (undo). Kali ini program berjalan dengan lancar tanpa ada satupun error. Berkat digunakannya Version Control dengan disiplin bencana yang ada di depan mata dapat dihindari.

Berdasarkan cerita diatas, jelas bahwa penggunaan Version Control dalam pengembangan software yang dilakukan oleh sebuah tim programmer sangat penting, bahkan sudah wajib hukumnya. Version control akan melakukan pencatatan setiap perubahan kode program. Dengan begitu, banyak sekali manfaat yang dapat diambil dan masalah yang dapat dihindari. Masih banyak lagi fitur-fitur version control yang akan kita bahas dalam bagian-bagian selanjutnya.

Kenapa Subversion?

Kalau ada yang gratis kenapa harus bayar? Kalau ada yang gratis dan legal, kenapa harus membajak?. Selain itu subversion sudah menjadi standard *de facto* version control di dunia opensource. Situs kolaborasi terbesar di dunia sourceforge.net sudah bermigrasi ke Subversion begitu juga situs dev.java.net sudah bermigrasi ke subversion. CollabNet membuat subversion sebagai pengganti langsung dari CVS yang sudah tidak dapat mendukung model pengembangan software dewasa ini. CVS mempunyai beberapa 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 didelete
5. Ijin akses tidak dapat diatur per folder

Subversion mengatasi semua kelemahan dari CVS dengan sangat elegan. Dengan desain seperti ini Subversion dapat diandalkan untuk mencatat semua perubahan sumber kode dengan sangat teliti. Terutama di era sekarang ini yang mengandalkan paradigma OOP dalam pengembangan software. OOP paradigma menggunakan metode “code by interface” yang terkadang memerlukan langkah refactoring dengan mengganti nama method, class (file) atau nama package(folder). CVS tidak dapat mencatat perubahan kode program karena ada penggantian nama dari file atau folder. Selain mengatasi kekurangan dari CVS, subversion juga mempunyai banyak keunggulan dibandingkan dengan CVS, antar lain :

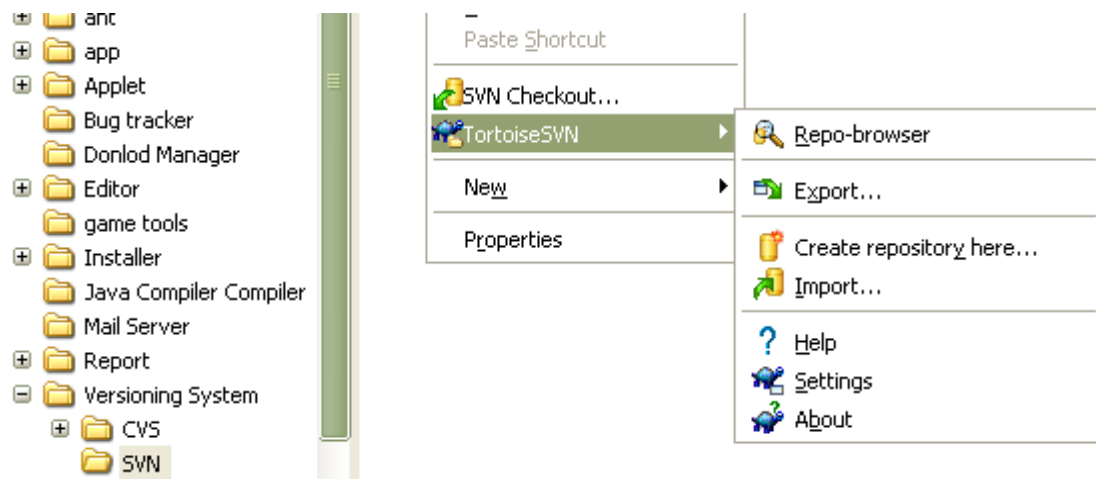
1. Dapat dijalankan dalam beberapa protokol, antar lain: HTTP, file, svn dan svn+ssh.
2. Dapat melakukan otentikasi user melalui protokol LDAP

Instalasi Subversion

Modul ini tidak akan membahas lebih lanjut tentang instalasi subversion atau konfigurasi lanjut subversion server. Untuk lebih lanjut silahkan download subversion dari website <http://subversion.tigris.org> kemudian bacalah dokumentasinya yang sudah sangat lengkap. Atau silahkan membeli buku “Konsep dan Penggunaan Subversion” oleh Endy Muhardin di toko buku, atau pesan langsung ke endy.muhardin@gmail.com. Atau baca tutorial sederhana tentang instalasi subversion yang saya buat <http://www.nagasakti.or.id/roller/Ifnu/entry/7>.

Instalasi TortoiseSVN

Dalam modul ini hanya akan dibahas aplikasi client dari subversion yaitu TortoiseSVN. Aplikasi TortoiseSVN dapat diunduh dari alamat <http://tortoisesvn.tigris.org>. Setelah diunduh langkah selanjutnya adalah melakukan instalasi TortoiseSVN, langkah-langkahnya sangat sederhana seperti proses instalasi program-program windows lain pada umumnya. Double klik file hasil downloadnya, dalam hal ini adalah TortoiseSVN-1.3.3.6219-svn-1.3.1.msi, kemudian ikuti instruksi selanjutnya. Setelah selesai proses instalasi, restart windows agar proses integrasi TortoiseSVN dengan windows explorer berjalan sukses.



Seperti kita lihat dalam gambar diatas, TortoiseSVN terintegrasi dengan context menu dari windows explorer. Pada bagian selanjutnya kita akan membahas penggunaan TortoiseSVN dalam sebuah project pengembangan perangkat lunak.

Menggunakan Subversion

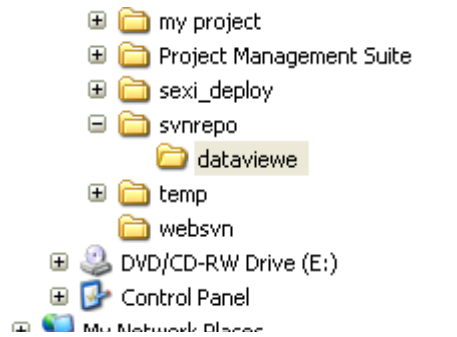
Membuat project baru

Jika kita bergabung ke dalam sebuah tim pengembangan perangkat lunak yang sudah berjalan, kemungkinan besar sudah tersedia repository dan server Subversion yang sudah terisi data dan kita tinggal menggunakannya. Namun jika kita baru memulai project pengembangan perangkat lunak, kita harus membuat repository baru yang masih kosong.

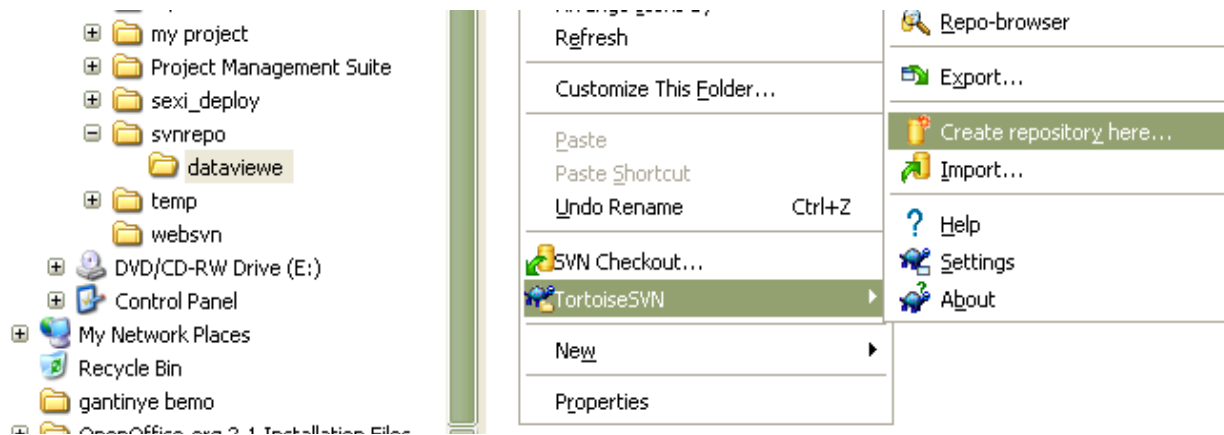
Pada bagian ini, kita akan melihat bagaimana memmulai project baru, membuat repository kosong dan beberapa pertimbangan yang perlu diperhatikan dalam mengatur folder repository.

Membuat repository baru

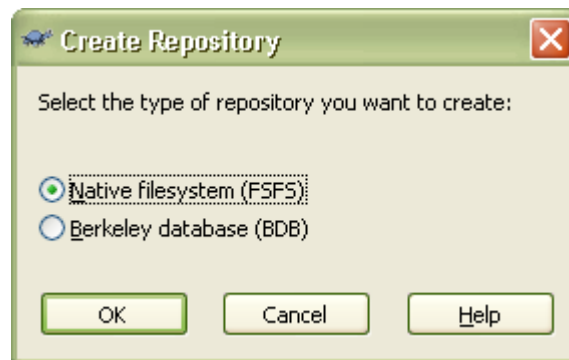
Cara membuat repository baru menggunakan TortoiseSVN sangatlah mudah, pertama-tama buatlah sebuah folder baru yang masih kosong, misalnya [d:\svnrepo](#), folder ini nantinya akan berisi repository-repository proyek pengembangan perangkat lunak. Kemudian buat lagi sebuah folder baru didalam [d:\svnrepo](#), misalnya : dataviewer.



Setelah folder untuk menyimpan repository dan folder untuk menyimpan proyek selesai dibuat, kita sudah siap untuk membuat repository baru, caranya sangat mudah. Buka windows explorer, klik kanan di jendela explorer sebelah kanan, pilih menu TortoiseSVN => Create repository here.

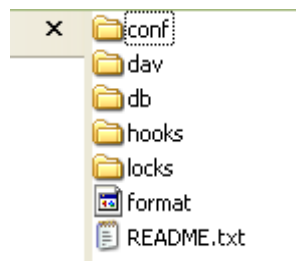


Kemudian akan muncul jendela yang akan menanyakan apakah tipe dari repository yang akan digunakan? Terdapat dua pilihan, yaitu: Native Filesystem(FSFS) atau Barkeley DB (BDB). Pilih FSFS, kemudian klik OK.



Setelah selesai, maka repository sukses dibuat dan kita akan melihat beberapa file dan folder dibuat secara otomatis oleh Subversion. File-file ini digunakan oleh Subversion untuk menyimpan semua data kode sumber, dan digunakan untuk menyimpan keterangan-keterangan yang diperlukan oleh subversion. Kita tidak akan pernah mengedit file-file ini secara langsung, kecuali file-file yang berada dalam folder hooks. File yang berada dalam folder hooks disebut dengan Hooks script yang merupakan file executable dan dijalankan oleh subversion pada event-event tertentu. Lebih lengkap mengenai hook

script, bagaimana membuatnya dan kapan subversion akan menjalankan hooks script dapat di baca di dokumentasi Subversion atau di buku “Konsep dan Penggunaan Subversion” yang ditulis oleh Endy Muhardin.



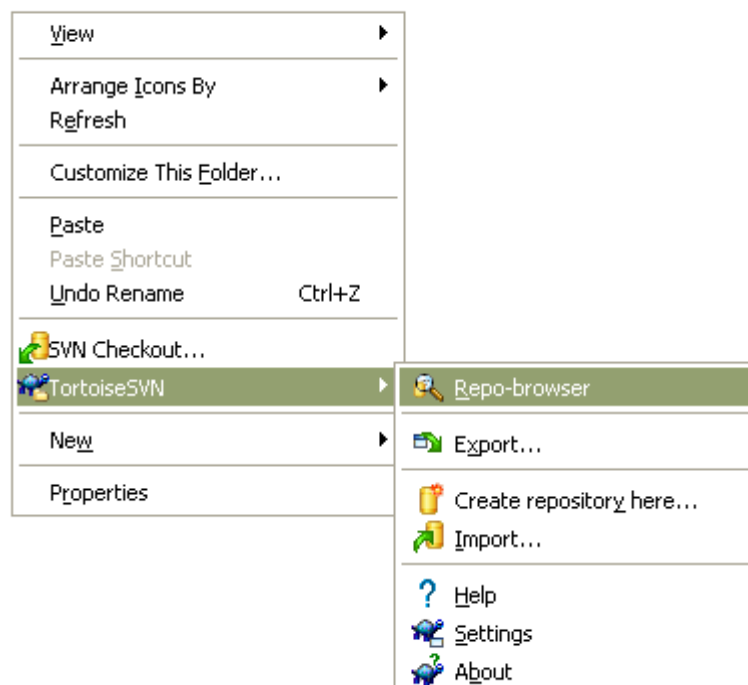
Kita tidak perlu membahas secara detail semua file yang dibuat secara otomatis oleh Subversion. Yang harus kita perhatikan adalah alamat dari folder yang telah kita buat tadi, yaitu :

d:\svnrepo\dataviewer

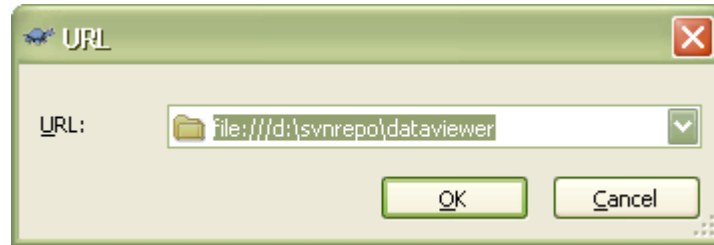
Alamat ini nantinya akan kita gunakan sebagai alamat untuk mengakses repository.

Untuk mengetahui apakah repository yang telah kita buat sukses, maka cobalah dengan mengakses repository tersebut menggunakan repo-browser dari TortoiseSVN. Berikut ini langkah-langkah menampilkan isi repository menggunakan repo-browser:

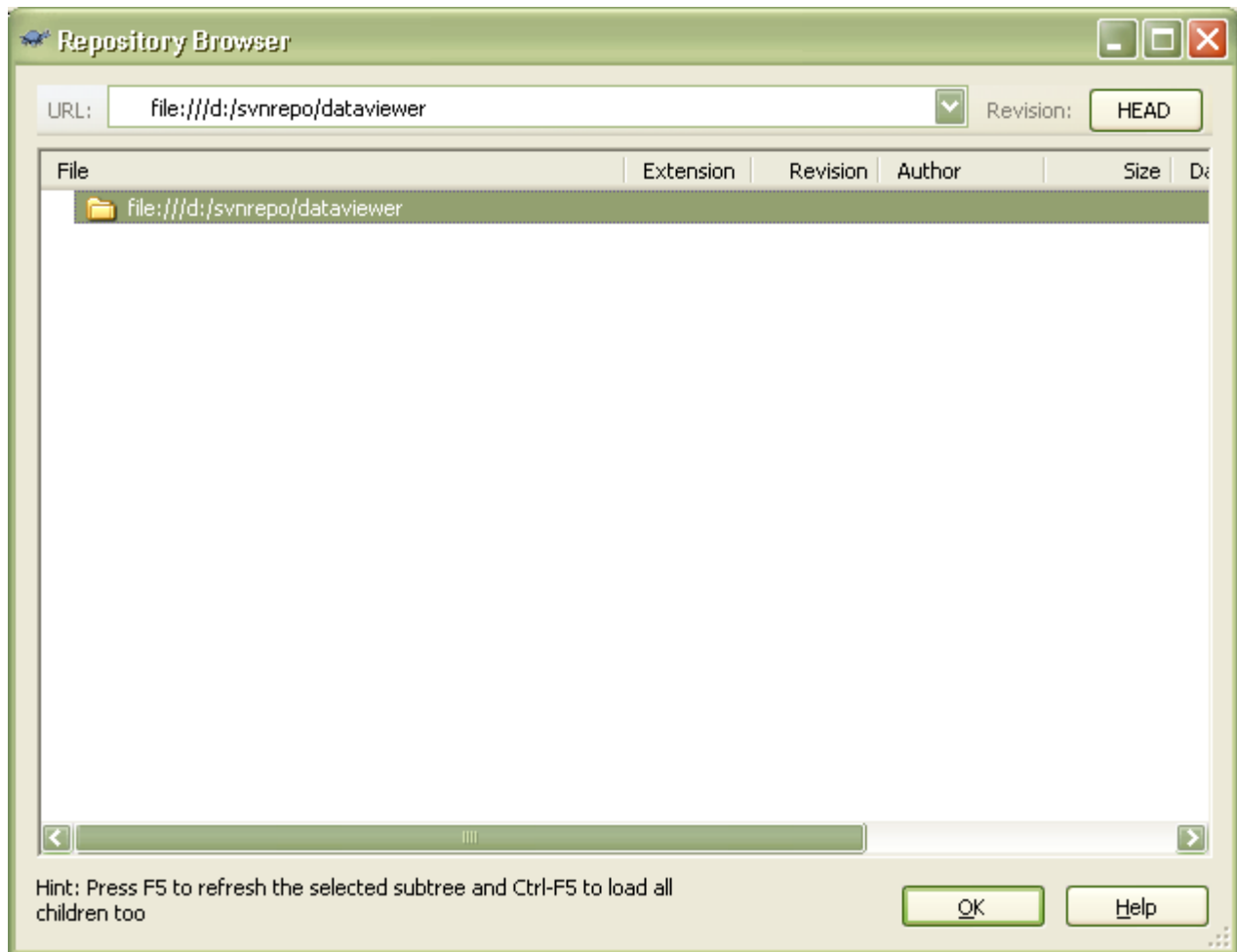
1. Munculkan windows explorer context (pop-up) menu dengan melakukan klik-kanan jendela bagian kanan dari windows explorer.
2. Pilih menu TortoiseSVN => repo-browser



- Setelah itu akan muncul jendela input yang meminta kita untuk memasukkan alamat repository yang akan dilihat. Masukkan alamat URL : <file:///d:/svnrepo/dataviewer> kemudian klik OK



- setelah semuanya dilakukan, akan muncul jendela repo-browser



Bisa kita lihat bahwa repository yang baru saja kita buat berhasil dan tentu saja isinya masih kosong.

Struktur folder repository

Setelah repository selesai dibuat, hal pertama yang harus dilakukan adalah menentukan struktur folder dari repository. Struktur folder ini sangat penting untuk diperhatikan, karena struktur folder yang rapi dan jelas dapat memudahkan pengguna mengerti isi repository hanya dengan melihat struktur dan nama

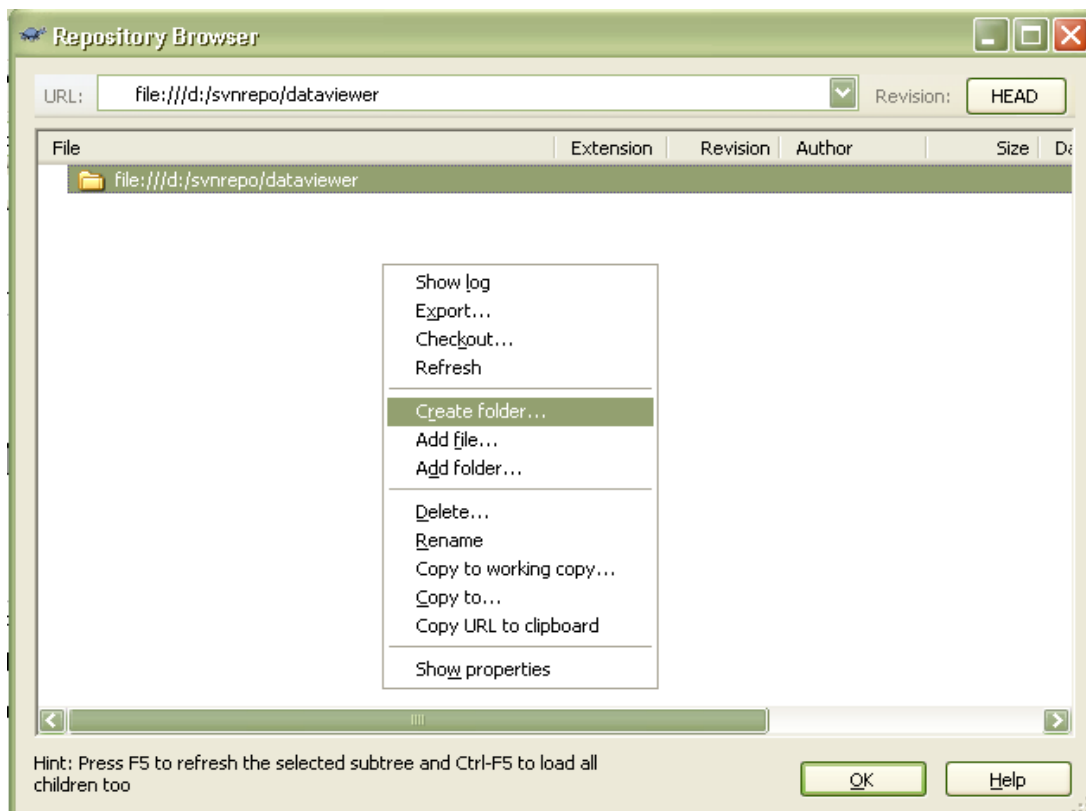
folder. Hal ini sangat bermanfaat untuk menghindari kesalahan yang mungkin dilakukan oleh user, misalnya commit ke folder yang salah.

Secara umum folder dalam repository dibagi menjadi tiga, yaitu:

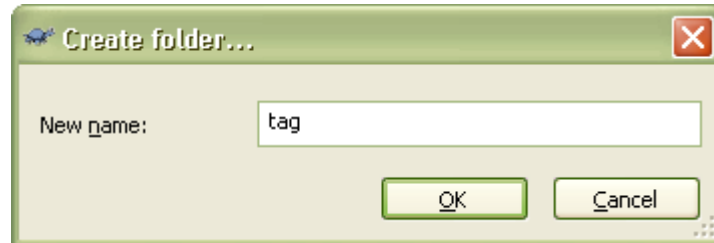
1. Trunk, dimana semua kode program yang masih dalam tahap pengembangan berada
2. Tag, pada beberapa event tertentu kita perlu menandai repository dan membuat titik yang jelas dimana kita bisa mengembalikan kondisi kode program dengan mudah. Misalnya rilis aplikasi ke QC, rilis aplikasi ke publik atau menginstal program ke lingkungan production.
3. Branch, jika kita ingin bekerja secara paralel, misalnya ada satu tim kecil pengembang yang melakukan bug fixing dan tim yang lain tetap menambahkan fitur baru ke folder trunk, kita perlu satu folder terpisah untuk bekerja, folder untuk memenuhi kebutuhan ini disebut sebagai branch.

Untuk membuat ketiga folder tersebut sangat mudah, buka kembali repo-browser dengan URL repository yang baru saja kita buat tadi. Lihat bagian sebelumnya untuk mengetahui langkah-langkah membuka repo-browser.

Setelah repo-browser terbuka, maka kita siap untuk membuat folder baru, ikuti langkah-langkah berikut ini:



1. Klik kanan didalam repo-browser untuk menampilkan context menu, pilih menu create folder. Setelah itu akan muncul jendela input untuk memasukkan nama dari folder yang baru saja dibuat. Masukkan tiga buah nama folder yang sudah kita definisikan sebelumnya: trunk, branch dan tag



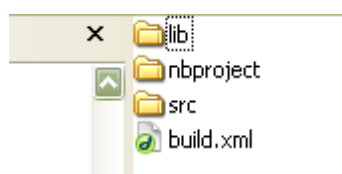
Setelah repository sukses dibuat dan struktur folder repository telah siap, hal berikutnya yang perlu kita lakukan adalah menentukan struktur folder untuk project kita. Kita melakukan ini dengan cara yang sedikit berbeda dengan ketika kita membuat struktur folder repository.

Ada dua cara yang bisa dilaksanakan untuk mengubah isi dan struktur folder repository. Cara pertama adalah dengan langsung mengedit isi repository dengan menggunakan repo-browser seperti sebelumnya sudah kita lakukan.

Cara kedua adalah dengan melakukan check-out repository ke folder lokal, lakukan perubahan terhadap folder lokal, baru kemudian gunakan perintah-perintah Subversion untuk mengupdate repository seperti commit, import dan add.

Mengedit langsung isi repository menggunakan repo-browser jarang dilaksanakan kecuali ketika mengkopy isi trunk ke dalam branch atau tag. Sedangkan operasi-operasi lainnya dilaksanakan dalam folder lokal kemudian memerintahkan subversion untuk mengupdate isi folder repository menggunakan perintah commit, import atau add.

Mari kita buat susunan folder project kita. Pada contoh ini kita akan menggunakan struktur standar aplikasi java yang dikembangkan menggunakan IDE Netbeans.



Dalam susunan folder diatas, kita menyediakan folder :

1. src : semua kode sumber aplikasi akan diletakkan di dalam folder ini
2. lib : berisi file-file library yang akan digunakan dalam aplikasi, misalnya file jar
3. nbproject : folder yang digunakan Netbeans untuk menyimpan konfigurasi dari aplikasi.

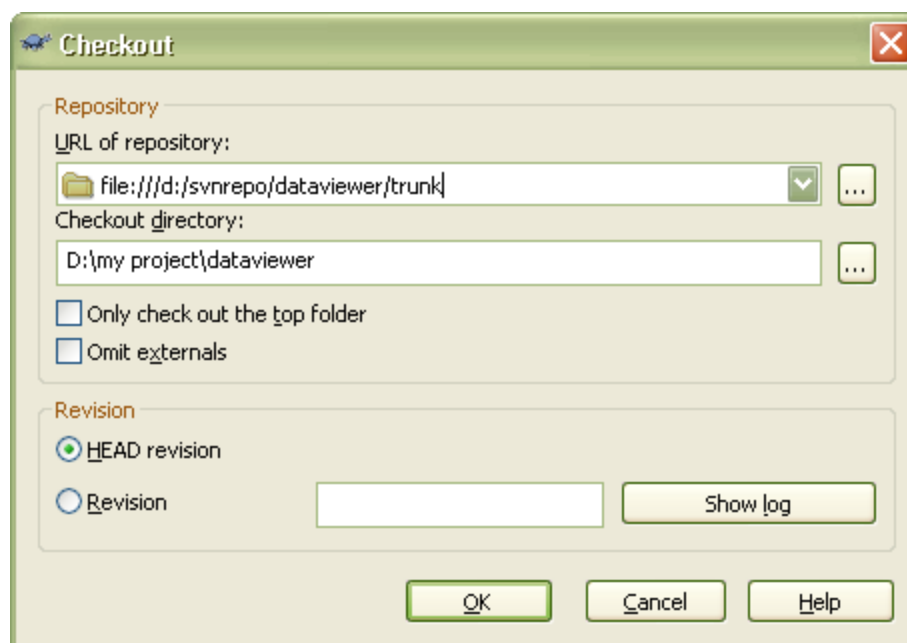
Ketiga folder tersebut diletakkan dalam sebuah folder yang nantinya akan kita gunakan sebagai folder kerja kita, misalnya di d:\my poject\dataviewer.

Import folder

Setelah kita menginstal TortoiseSVN, membuat repository, mengorganisasi folder repository dan menyiapkan folder lokal sebagai tempat kerja kita, maka semua kebutuhan untuk bekerja dengan Subversion sudah siap. Langkah berikutnya adalah menghubungkan antara repository dengan folder kerja kita, proses ini dikenal dengan Checkout.

Sesuai dengan aturan yang telah disepakati, semua perubahan dan penambahan fitur baru terhadap program dilaksanakan di folder trunk, sehingga checkout akan dilakukan terhadap folder ini.

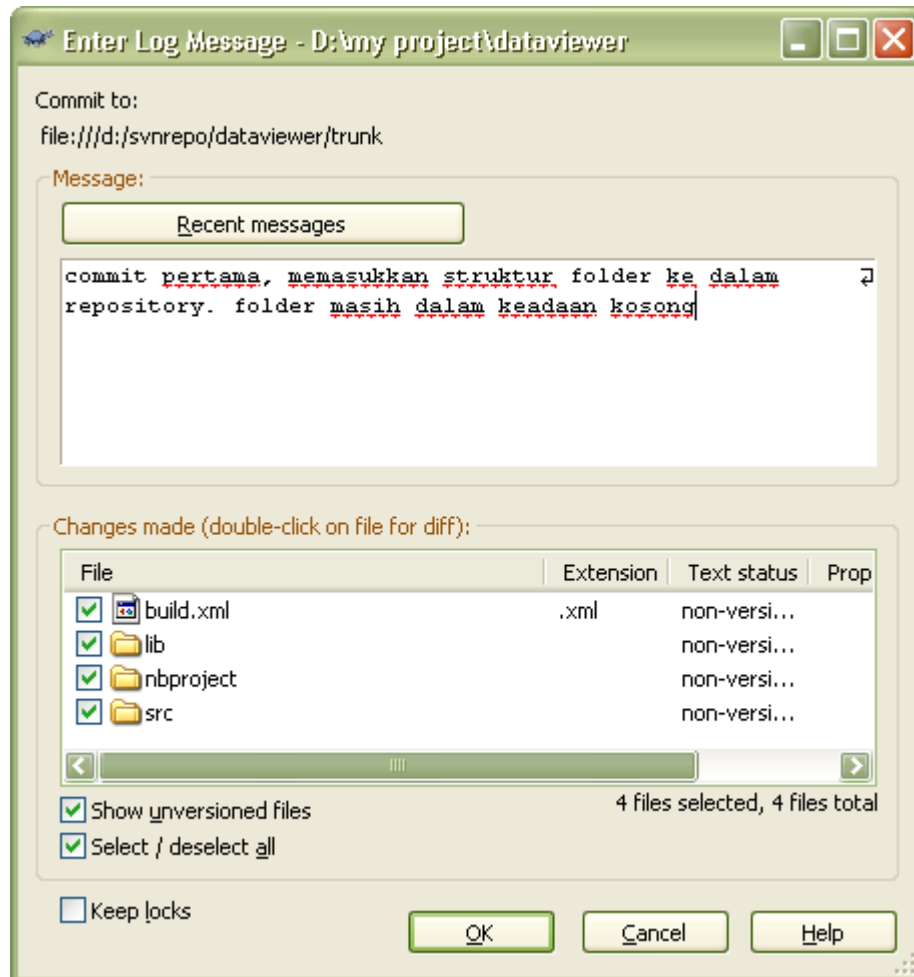
Untuk melakukan checkout, munculkan windows explorer context menu dengan melakukan klik kanan di jendela sebelah kanan dari windows explorer, kemudian pilih menu SVN checkout, selanjutnya akan muncul jendela SVN checkout seperti berikut ini:



Isikan URL <file:///d:/svnrepo/dataviewer/trunk> sebagai folder yang akan kita checkout. Setelah proses checkout selesai, maka folder d:\my poject\dataviewer sudah siap kita gunakan. Hal ini ditandai

dengan berubahnya isi context menu dari windows explorer, dimana sekarang muncul menu svn commit dan svn update menggantikan menu svn checkout, selain itu akan ada sebuah folder hidden baru bernama .svn yang berisi informasi keterhubungan antara folder lokal dan folder repository.

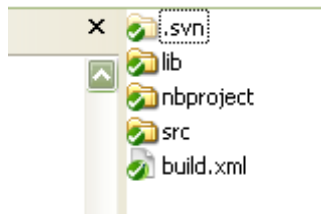
Langkah selanjutnya adalah mengimport folder-folder yang telah kita buat sebelumnya ke dalam repository dengan cara memilih menu svn commit.



Catatan Penting :

Pesan yang diisikan ketika kita melakukan commit disebut Log. Log yang baik menyatakan maksud dari perubahan, bukan isi perubahan. Isi perubahan dapat diketahui dengan mudah menggunakan perintah diff. Bagi rekan kita, yang lebih penting adalah mengapa kita melakukan perubahan tersebut bukan apa isi dari perubahan(Endy, 2006).

Setelah proses commit selesai dilaksanakan icon folder akan berubah seperti dibawah ini untuk menandakan bahwa folder tersebut adalah folder lokal yang terhubung dengan repository.



Beberapa istilah penting

1. Revision. Menunjukkan kondisi repository pada saat tertentu. Pada waktu kita membuat repository kosong, nilai revisinya adalah 0. ketika kita mengimport folder lokal ke folder repository maka nilai revision bertambah menjadi 1. setiap perubahan yang terjadi pada repository akan menaikkan nomer revision satu angka. Nomer revision berlaku global untuk satu repository bukan per file atau folder seperti pada CVS. Jadi jika kita merubah isi dari repository baik folder ataupun file dengan cara menambah, menghapus, mengganti nama ataupun mengubah isinya, maka akan merubah nilai revision untuk keseluruhan repository.
2. Folder Lokal. Mengacu pada file dan folder dalam komputer kita yang diperoleh dari proses checkout. Pada umumnya programmer akan melakukan modifikasi terhadap file dan folder ini pada rentang waktu tertentu, kemudian setelah dirasa perubahan sudah cukup dan tidak ada lagi error, programmer akan melakukan commit untuk mengirim perubahan ke server repository.
3. Folder Repository. Mengacu pada file dan folder yang berada dalam repository server. Modifikasi langsung terhadap file dan folder repository tidak lazim dilaksanakan, namun ada kalanya perubahan langsung terhadap folder dalam repository lebih praktis daripada harus melakukan serangkaian langkah checkout ke folder lokal, melakukan modifikasi dan mengcommit perubahan ke repository.
4. Checkout. Dilakukan hanya sekali ketika kita mulai bekerja, secara praktis perintah ini akan mengambil file dan folder dari repository ke folder lokal. Hasilnya adalah Folder Lokal.
5. Commit. Perintah untuk mengirimkan perubahan folder lokal ke repository.
6. Update. Perintah untuk mengambil perubahan terakhir dari repository untuk dimasukkan ke dalam folder lokal. Biasanya perintah ini lazim dilaksanakan sebelum kita melakukan commit, untuk menghindari kegagalan proses commit karena folder lokal mempunyai nilai revision tidak sama dengan nilai revision repository.

Cara kerja subversion

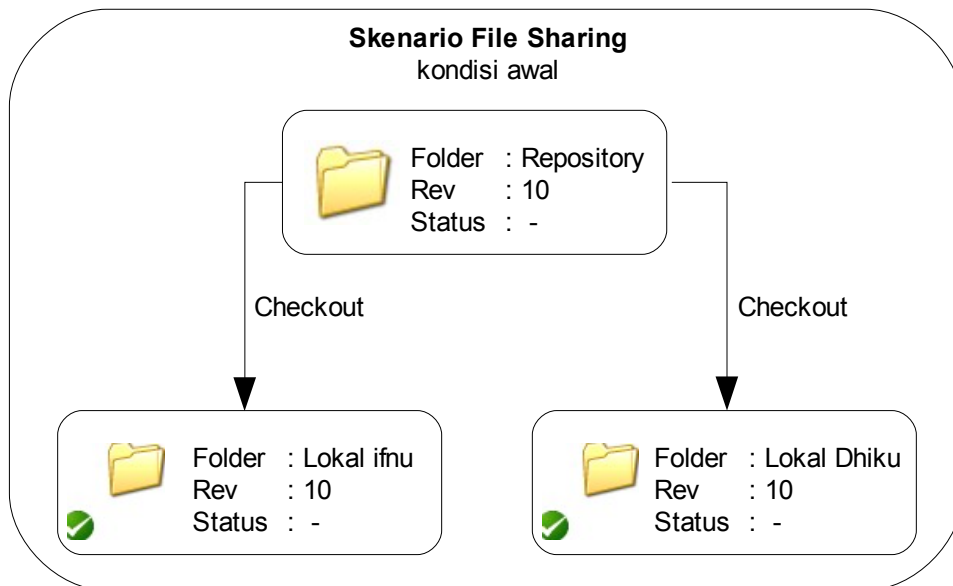
Sebelum memulai bekerja dengan subversion, ada beberapa konsep yang harus dipahami mengenai

version control. Diantaranya adalah versioning model dan mekanisme delta. Aplikasi version control yang berbeda dapat mengimplementasikan konsep ini secara berbeda pula.

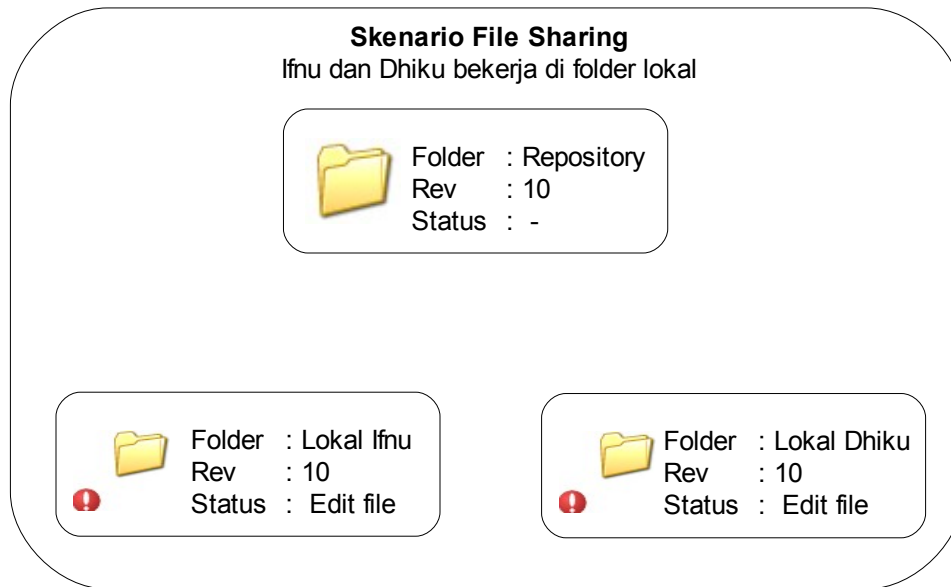
Permasalahan file sharing

Misalnya ada dua programmer Ifnu dan Dhiku, sedang bekerja di sebuah file yang sama. Mari kita lihat apa yang terjadi bila kita tidak menggunakan version control, tetapi mengandalkan mekanisme file sharing. Mekanisme file sharing ini menyimpan file dalam sebuah repository yang terletak di sebuah komputer yang salah satu foldernya dishare, atau di FTP server, atau yang paling primitif, source code diletakkan di dalam USB Flashdisk.

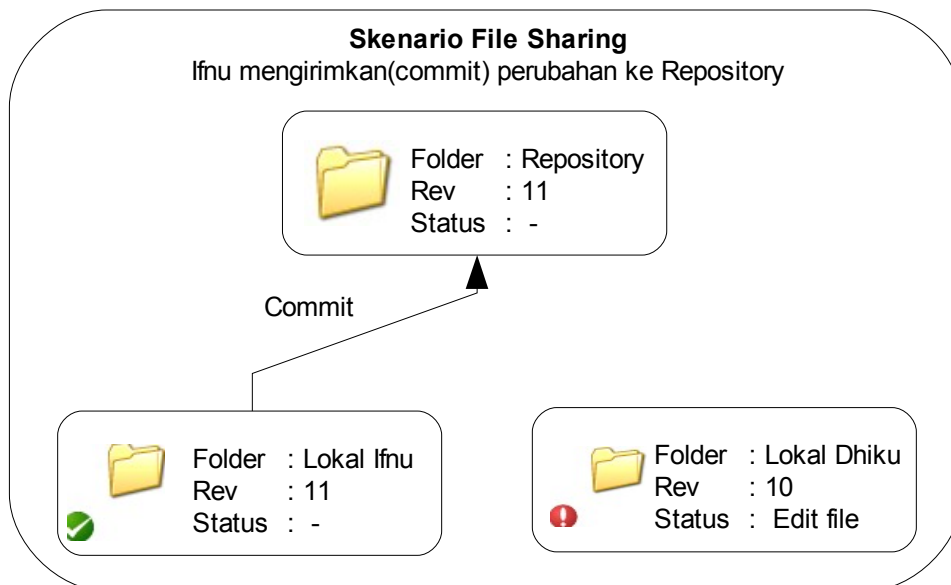
Ketika akan mulai bekerja, Ifnu dan Dhiku akan mengambil file yang sama dari dalam repository.



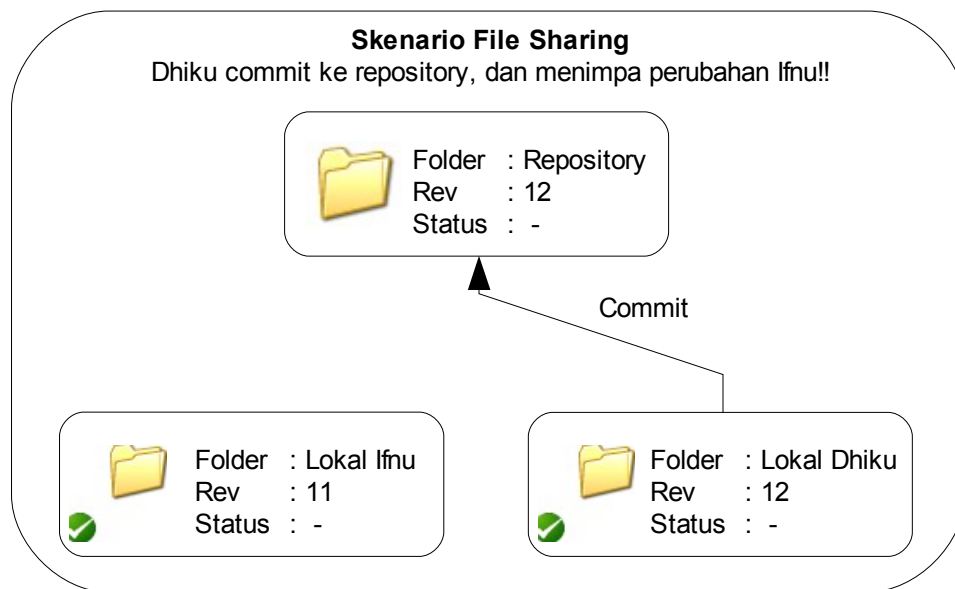
Setelah itu Ifnu dan Dhiku mulai bekerja di folder lokal masing-masing. Misalnya, dalam folder repository terdapat sebuah file MainPanel.java. Ifnu dan Diku sama-sama melakukan perubahan terhadap file MainPanel.java.



Kebetulan Ifnu selesai lebih dulu. Kemudian dia segera menyimpan perubahan di repository



Tidak lama kemudian Dhiku selesai bekerja dengan MainPanel.java, dan mengirimkan perubahan yang sudah dilakukanya ke dalam repository.

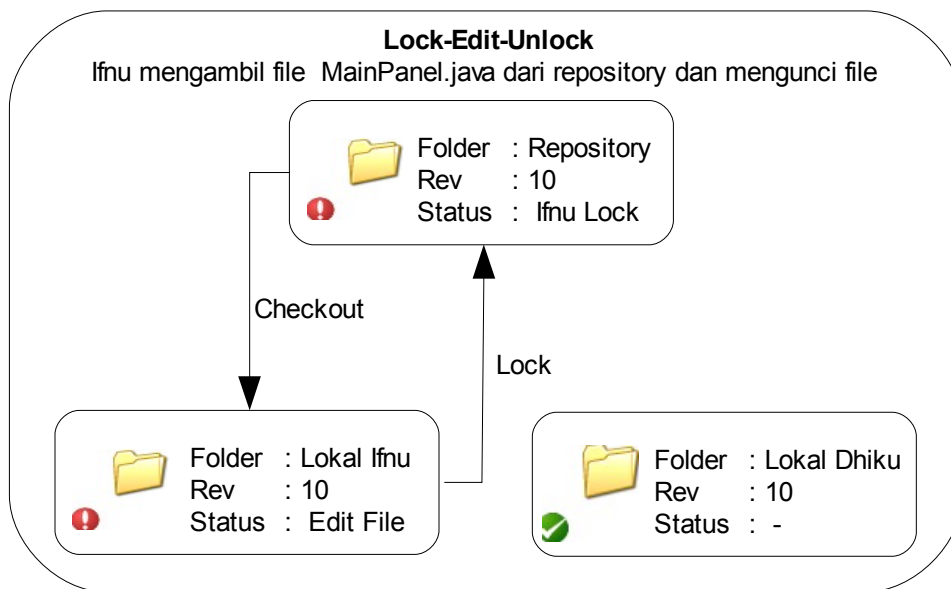


Tanpa adanya aplikasi Version Control yang baik, perubahan yang dibuat oleh Dhiku akan menimpa perubahan yang telah disimpan Ifnu sebelumnya. Hal ini bisa menyebabkan pertumpahan darah dalam tim. Bayangkan pertempuran yang mungkin terjadi jika jumlah programmer tak hanya dua, melainkan seratus empatpuluh sembilan orang!!. Permasalahan File sharing ini dapat diselesaikan dengan dua pendekatan yaitu model Lock-Edit-Unlock, dan model Checkout-Edit-Merge.

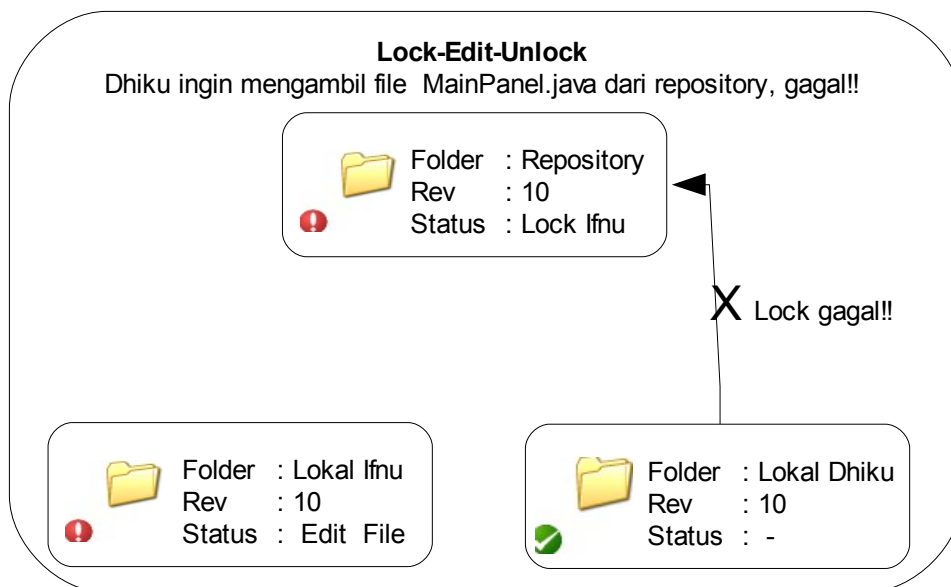
Model lock-edit-unlock

Model lock-edit-unlock adalah model yang dipilih oleh Microsoft untuk mengembangkan version controlnya yaitu Visual Source Safe. Dengan skenario yang sama, kita akan melihat bagaimana model ini mengatasi masalah file sharing.

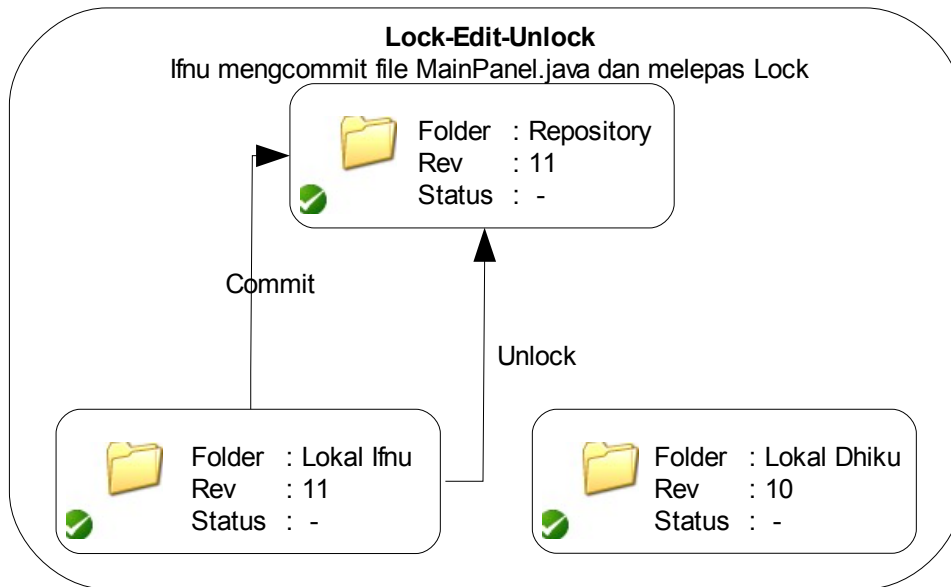
Ifnu lebih dahulu terkoneksi ke internet, dan ingin bekerja di file MainPanel.java. Dia langsung saja mengambil file MainPanel.java dari repository. Secara otomatis Visual Source Safe akan mengunci file MainPanel.java.



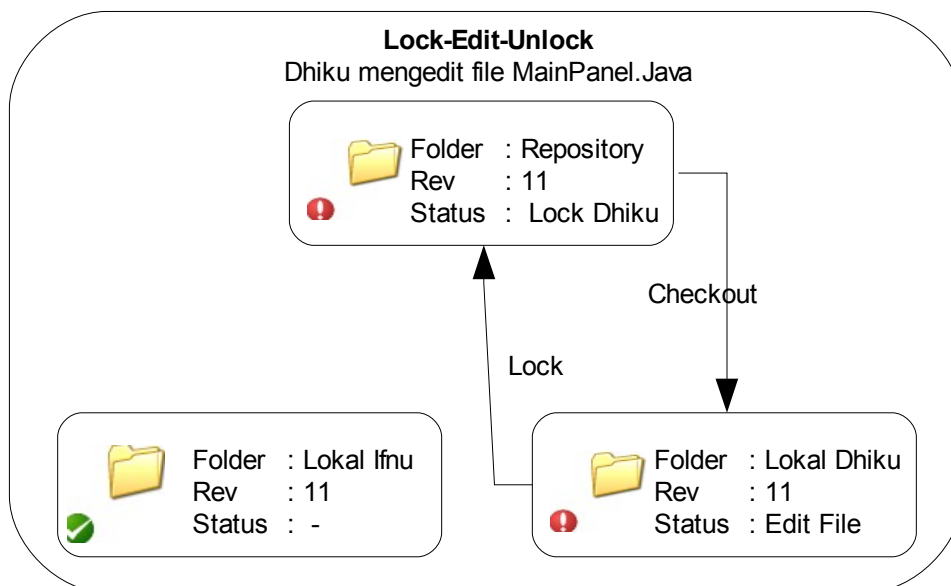
Tidak lama kemudian Dhiku konek ke internet, dan ingin bekerja dengan file MainPanel.java juga. Dia berusaha untuk mengambil file tersebut dari repository. Tentu saja, usaha Dhiku mengambil file MainPanel.java dari repository gagal karena Ifnu memegang lock terhadap file MainPanel.java.



Setelah Ifnu selesai bekerja dengan file MainPanel.java dia akan menyimpan perubahan yang telah dibuatnya ke dalam repository. Proses ini secara otomatis akan melepaskan lock terhadap file MainPanel.java sehingga semua orang kembali bisa bekerja dengan file ini.



Dhiku mencoba lagi untuk mengambil file MainPanel.java dari repository, kali ini usahanya berhasil karena Ifnu sudah melepaskan locknya.



Model ini sangat kaku dan melibatkan proses lock-unlock yang kompleks, sehingga akan muncul masalah-masalah dibawah ini.

1. Bagaimana kalau Ifnu membutuhkan waktu setengah hari untuk bekerja dengan file MainPanel.java? Apakah Dhiku akan duduk menganggur menunggu Ifnu selesai bekerja dengan file MainPanel.java?
2. Bagaimana kalau Ifnu bekerja dengan file MainPanel.java seharian penuh?, kemudian tanpa rasa bersalah pulang ke rumah bersiul-siul tanpa melepaskan locknya?
3. Bagaimana jika sebenarnya Ifnu dan Dhiku ingin bekerja dengan file MainPanel.java di bagian

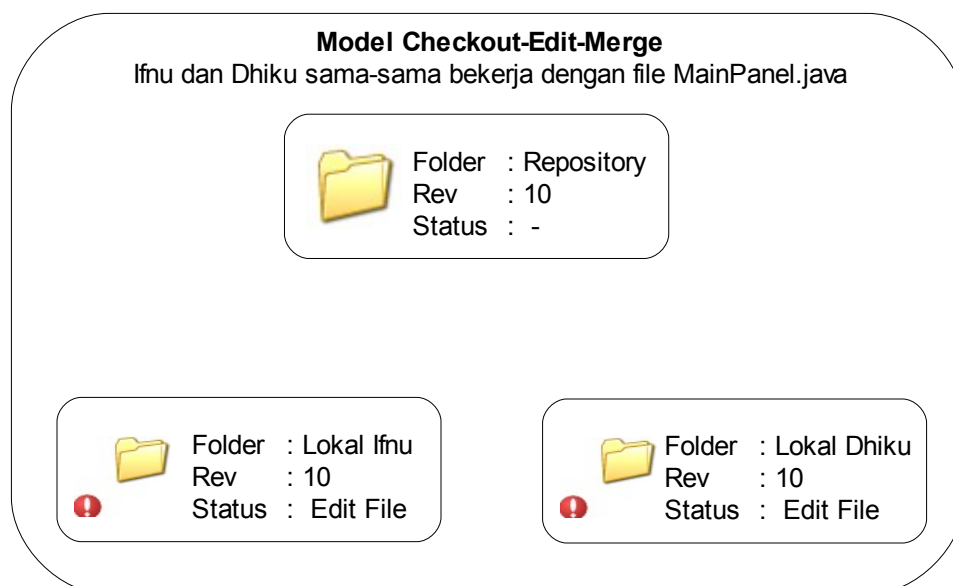
yang berbeda? Ifnu ingin menambahkan method showSplash() dan Dhiku hanya ingin memperbaiki code System.exit(0) menjadi System.exit(1) di bagian yang berbeda, apakah mereka berdua saling menunggu?

Ini memang kelemahan dari model lock-edit-unlock. Dengan model ini proses management kode sumber menjadi kaku dan tidak scalable. Model ini sulit mengakomodasikan jumlah programmer yang banyak dan mereka saling bekerja di bagian yang sama. Semakin banyak orang yang bekerja dalam satu tim, mekanisme lock-unlock akan sering terjadi. Semetara orang lain sedang bekerja dengan file yang kita butuhkan terpaksa kita mengajak rekan yang lainnya untuk segera membuka sesi baru game DotA atau login ke Friendster, lebih parah lagi mulai sibuk membuka-buka situs JobsDB.com atau karir.com.

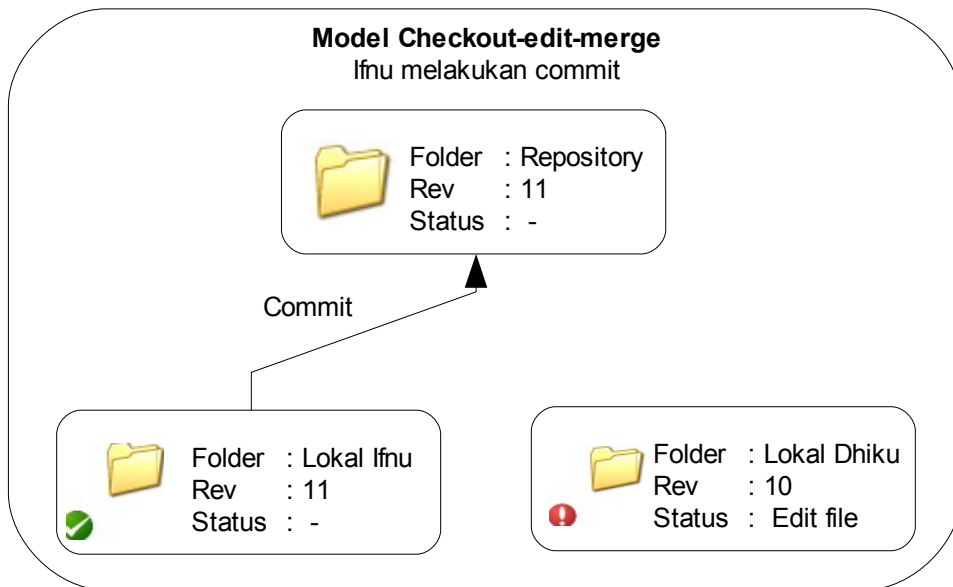
Model checkout-edit-merge

Model inilah yang digunakan oleh Subversion, model ini tidak menyaratkan adanya siklus lock-unlock (walaupun metode lock-unlock juga bisa dilaksanakan oleh Subversion), sehingga kasus rebutan file tidak lagi terjadi. Subversion dapat mengakomodasi tim dengan jumlah programmer sangat besar tanpa harus saling menunggu penggunaan resource.

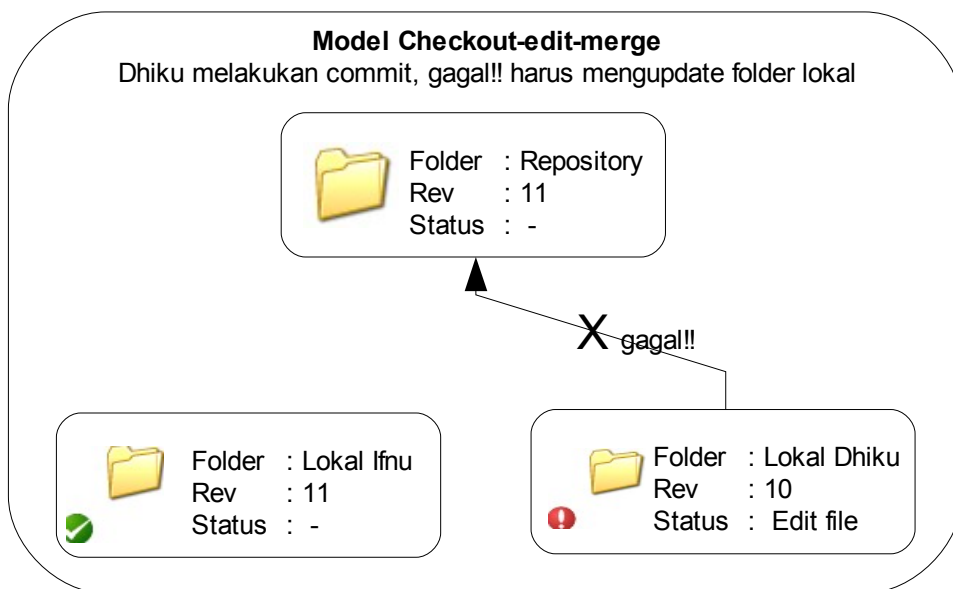
Mari sekali lagi kita saksikan aksi Ifnu dan Dhiku dalam bekerja sama mengedit file MainPanel.java. Kali ini tidak akan jadi masalah siapa yang akan melakukan proses checkout terlebih dahulu, karena tidak ada lagi mekanisme locking.



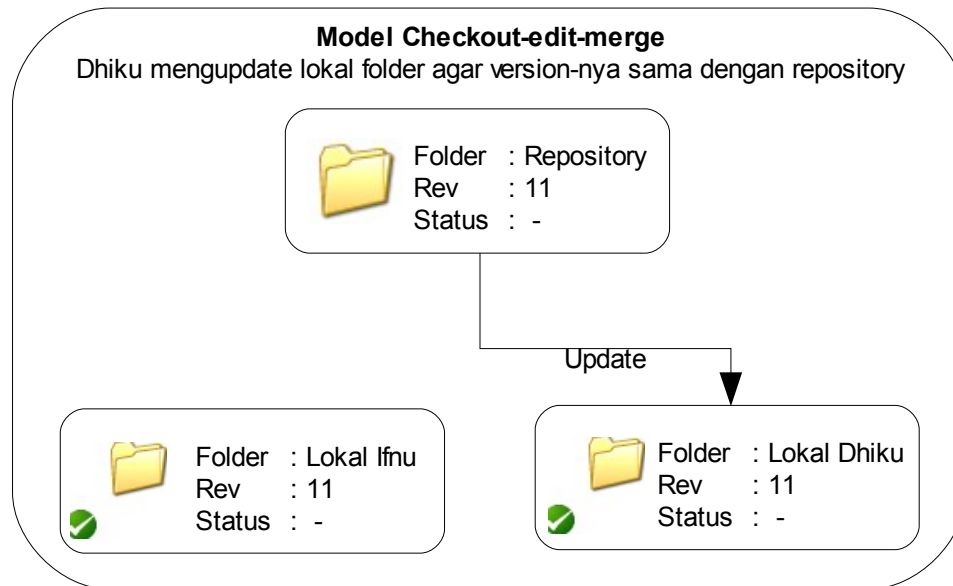
Ifnu selesai lebih dahulu bekerja dengan MainPanel.java dan segera melakukan commit untuk menyimpan perubahan dalam file MainPanel.java ke repository.



Tak lama kemudian Dhiku juga selesai dengan pekerjaannya dan akan berusaha untuk melakukan commit ke repository. Namun commit yang dilakukan Dhiku akan mengalami error “Out-of-date”. Error ini disebabkan nilai version repository lebih besar dari nilai version folder lokal Dhiku.



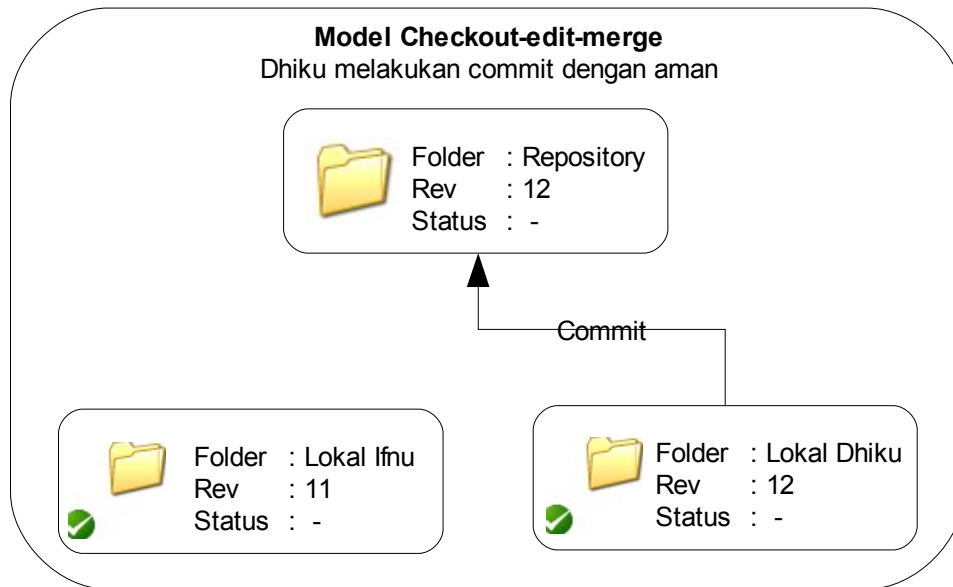
Untuk mengatasi hal ini Dhiku harus melakukan Update terhadap folder lokalnya agar nilai version folder lokal sama dengan nilai version repository.



Langkah ini juga akan memasukkan perubahan yang dilakukan oleh Ifnu dalam file MainPanel.java ke dalam folder lokal Dhiku. Ada dua kemungkinan yang terjadi:

1. Ifnu dan Dhiku bekerja di bagian (baris) yang berbeda dalam MainPanel.java. Misalnya Ifnu memperbaiki kode dalam method showSplash() sedangkan Dhiku memperbaiki kode System.exit(0) menjadi System.exit(1) di bagian yang berbeda. Skenario ini menunjukkan penggabungan (merge) perubahan yang aman. Subversion tidak akan memberikan peringatan apa-apa, dan Dhiku dapat melanjutkan dengan melakukan commit perubahan yang telah dilakukanya terhadap MainPanel.java.
2. Ifnu dan Dhiku bekerja di bagian (baris) yang sama dalam MainPanel.java. Misalnya keduanya mengedit baris ke 143 di dalam file MainPanel.java. Subversion akan memberikan peringatan bahwa telah terjadi **konflik**. Perlu suatu cara untuk melakukan resolusi terhadap konflik yang terjadi. Ifnu dan Dhiku harus duduk bersama untuk merundingkan perubahan mana yang akan dipakai, perubahan oleh Ifnu atau perubahan oleh Dhiku, atau kedua perubahan dipakai atau tidak satupun perubahan dipakai. Lebih lanjut tentang resolusi konflik akan kita bahas di bagian berikutnya.

Setelah Dhiku melakukan update, dan mengatasi konflik yang (mungkin) terjadi, Dhiku dapat melakukan commit dengan aman.



Delta dan Diff

Bagaimana sih sebenarnya Subversion menyimpan semua perubahan yang dimasukkan ke dalam repository? Apakah setiap ada perubahan dibuat sebuah file baru yang berisi versi sebelumnya? Bukankah teknik copy ini bisa membuat ukuran repository membengkak?. Subversion tidak menyimpan satu copy utuh kondisi file sebelumnya, melainkan hanya menyimpan perubahan yang ditambahkan user di setiap versionya. Sehingga file terbaru yang ada dalam repository adalah gabungan dari perubahan-perubahan ini.

Subversion untuk programmer

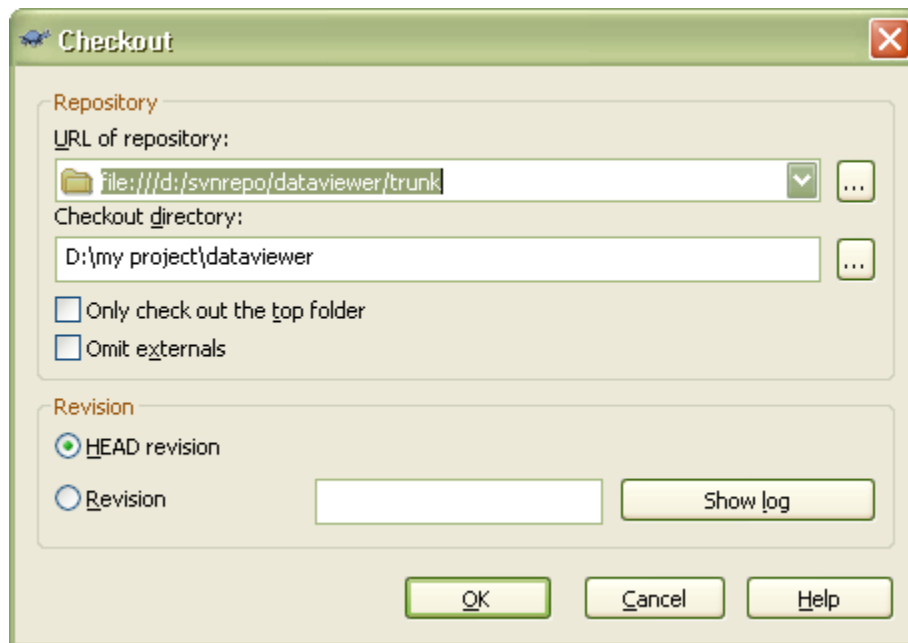
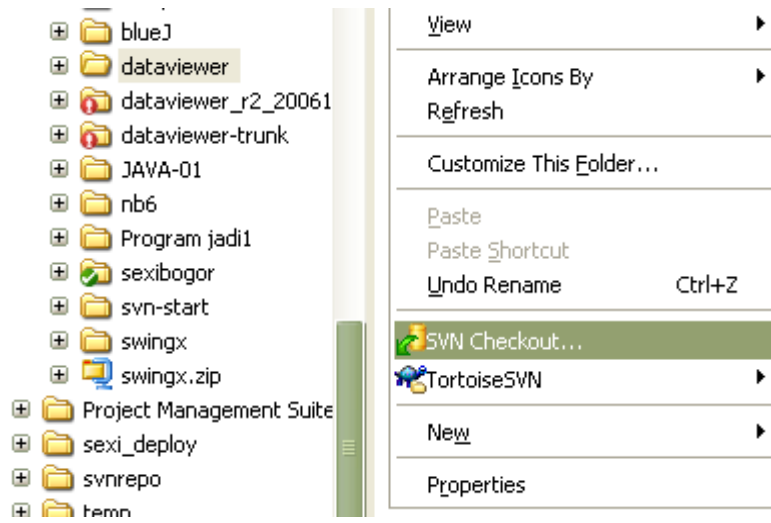
Penggunaan utama Version Control (Subversion) adalah untuk mengelola kode program. Pada sub-bab ini, kita akan menjadi programmer, dan melakukan kegiatan sehari hari menulis kode program, kemudian menyimpan kode program ke dalam repository subversion. Dalam sub-bab ini kita akan membahas lebih lanjut fitur-fitur yang dimiliki oleh subversion seperti tag, branch dan merge.

Siklus kerja dengan subversion

Pada hari pertama kita masuk kerja dan ditempatkan dalam sebuah tim pengembang perangkat lunak, biasanya kita akan diberikan sebuah komputer yang masih bersih dan hanya terinstal program-program umum yang akan kita gunakan.

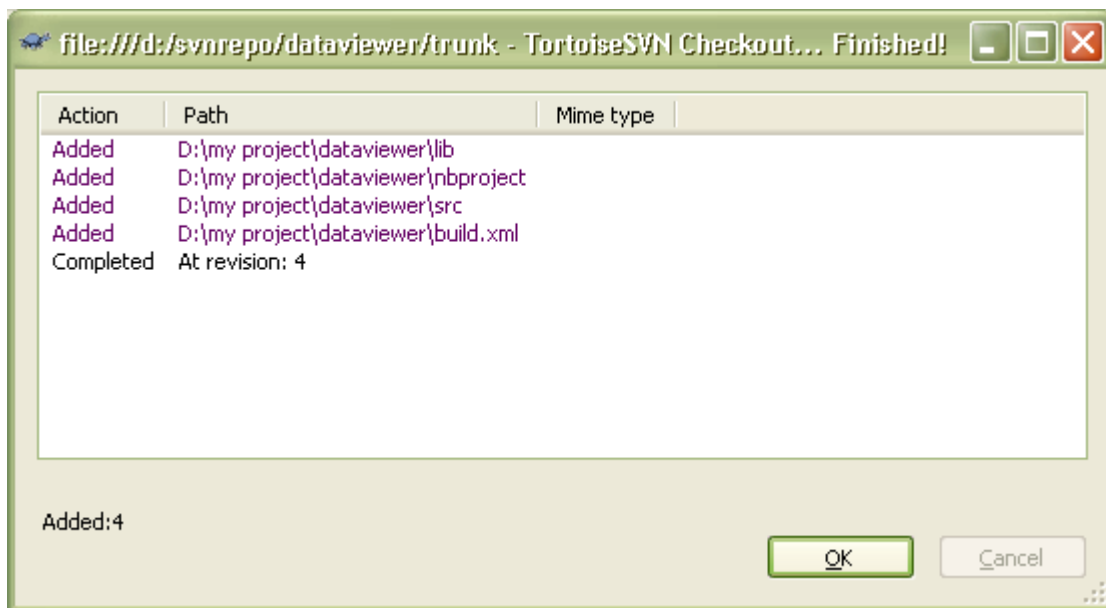
Hal pertama yang akan kita lakukan adalah mengambil kode program dari terbaru repository. Cara ini disebut dengan Checkout. Revision terbaru dalam repository dikenal dengan istilah revision HEAD. Langkah yang harus kita lakukan adalah membuat sebuah folder yang masih kosong, folder ini akan kita gunakan sebagai folder lokal tempat kita bekerja. Misalnya foldernya adalah d:\my

project\dataviewer. Masuk ke folder yang baru dibuat tersebut, klik kanan di jendela sebelah kanan dan pilih menu SVN Checkout.



Setelah menu SVN Checkout dipilih, akan muncul jendela checkout seperti di bawah ini. Isikan URL <file:///D:/svnrepo/dataviewer/trunk> di dalam kolom isian URL of repository. Pilih HEAD revision untuk mengambil revision terbaru dari repository.

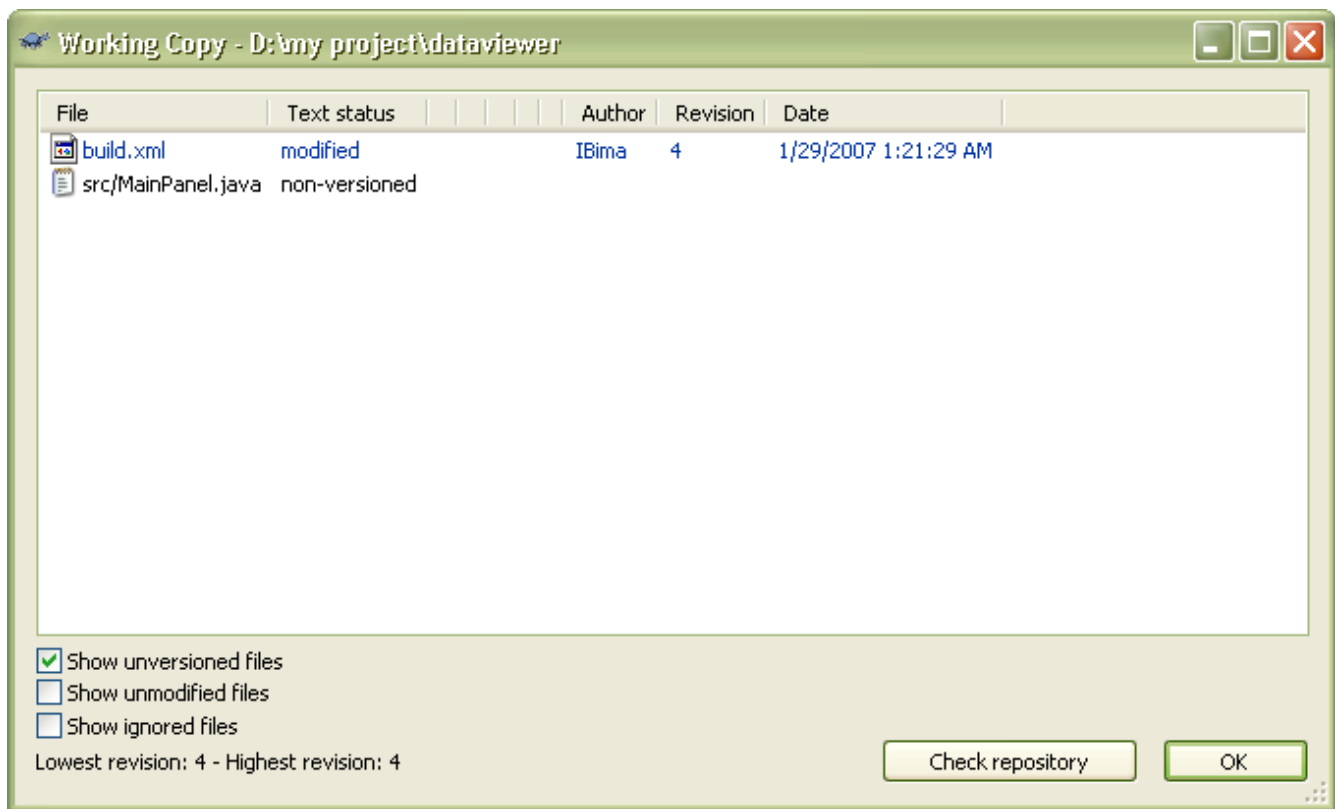
Dapat kita lihat dari gambar diatas, kita akan men-checkout folder trunk dari repository dataviewer, revision yang diambil adalah revision terbaru(HEAD).



Proses checkout berhasil mengambil 3 folder dan satu file. Revision terbaru adalah revision 4.

Perhatikan folder lokal yang berhasil dibuat, tampilan icon foldernya didekorasi dengan lingkaran berwarna biru dan tanda centang. Perubahan icon ini menandakan bahwa folder lokal kita terhubung dengan repository. Selain perubahan icon di folder lokal, isi context menu juga berubah.

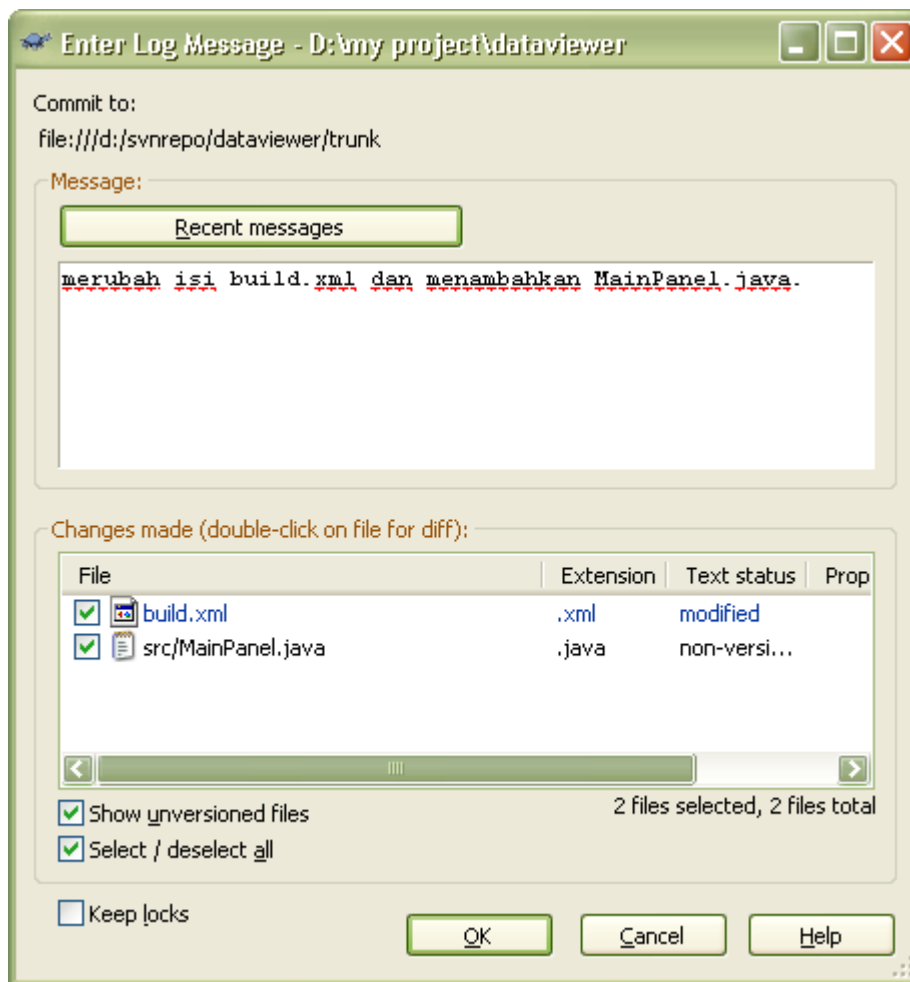
Segera setelah folder lokal siap, kita bisa mulai bekerja dengan file-file yang ada di dalamnya. Mari kita tambahkan sedikit perubahan pada file build.xml. Buka file build.xml dengan text editor, kemudian tambahkan sedikit perubahan, simpan perubahan, dengan segera icon file build.xml akan berubah dengan adanya lingkaran merah dengan tanda seru didalam lingkaran. Icon ini menandakan bahwa build.xml sudah dirubah dan perubahanya masih belum disimpan di repository. Kemudian buat sebuah file kosong dengan nama MainPanel.java didalam folder src.



Ada cara yang lebih baik untuk melihat daftar semua file yang berubah, yaitu dengan menggunakan menu "check for modification". Caranya: klik kanan di jendela windows explorer sebelah kanan, setelah context menu muncul, pilih menu: TortoiseSVN=>Check for modification.

Cara ini jauh lebih baik dibandingkan dengan hanya melihat tanda yang ada di icon file, karena perubahan yang terjadi bisa saja penambahan file, penghapusan file atau penamaan ulang file. Ketiga jenis perubahan ini tentu saja tidak dapat diketahui hanya dengan melihat tanda yang ada di icon file. Selain itu, cara ini dapat melihat perubahan yang terjadi di dalam folder yang lain.

Setelah perubahan selesai disimpan, lakukan commit ke repository dengan cara memilih menu svn commit di dalam context menu windows explorer. Pilih file build.xml dan MainPanel.java. Jangan lupa untuk memberikan Log yang jelas agar rekan kita dapat mengetahui maksud perubahan yang kita lakukan.



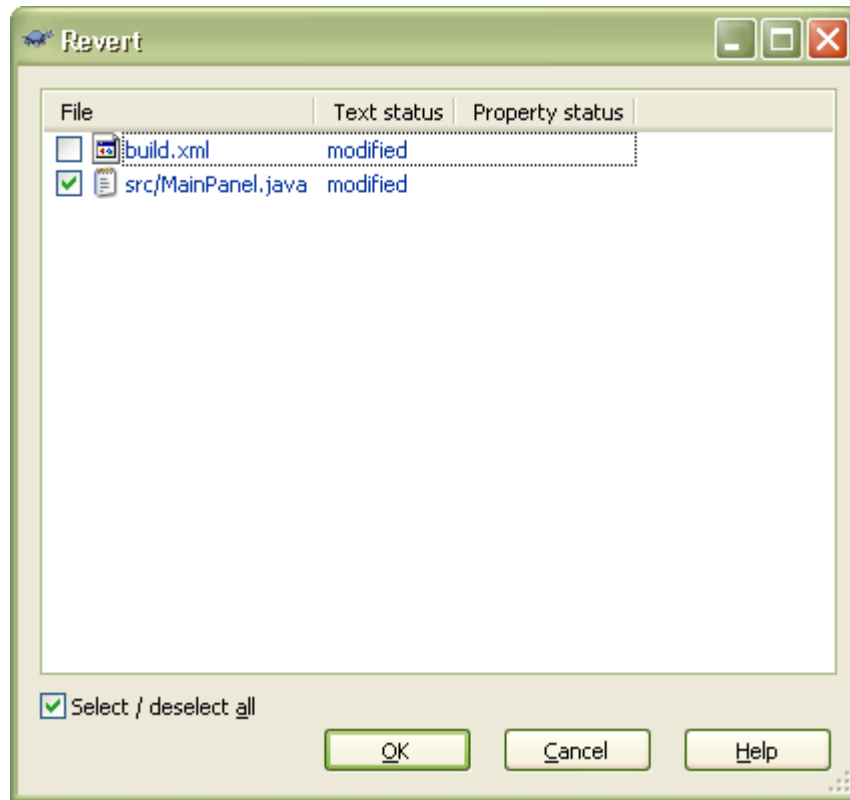
Bekerja dengan Tim

Subversion dibuat dengan tujuan utama memudahkan pengembangan software yang melibatkan sebuah tim. Untuk membuktikannya, kita akan mengundang Dhiku masuk dalam tim pengembang software dataviewer. Seperti yang sudah kita ketahui, Dhiku akan mulai bekerja dengan melakukan checkout kode program dari repository <file:///d:/svnrepo/dataviewer/trunk> ke dalam folder lokalnya. Tidak lama kemudian Dhiku mulai bekerja dengan file MainPanel.java, melakukan beberapa perubahan kemudian mengirimkan perubahan ke repository dengan menggunakan perintah commit.

Membatalkan perubahan (revert)

Ifnu terus bekerja di dalam folder lokalnya, melakukan perubahan di file MainPanel.java, setelah setengah jam berlalu, Ifnu hendak melakukan commit, namun tentu saja commit yang dilaksanakan Ifnu gagal, kenapa? Karena Dhiku sebelumnya sudah melakukan perubahan terhadap file MainPanel.java dan mengirimkan perubahan ke repository. Setelah berdiskusi, akhirnya diputuskan

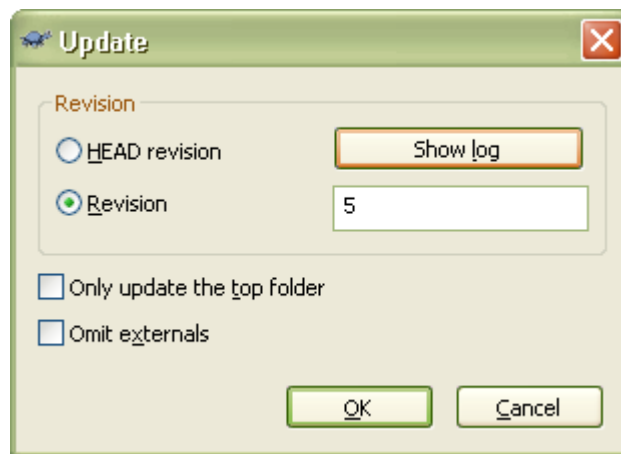
bahwa perubahan yang digunakan adalah perubahan yang dilakukan oleh Dhiku. Ifnu akan membatalkan perubahan yang dilakukanya dengan menggunakan perintah revert.



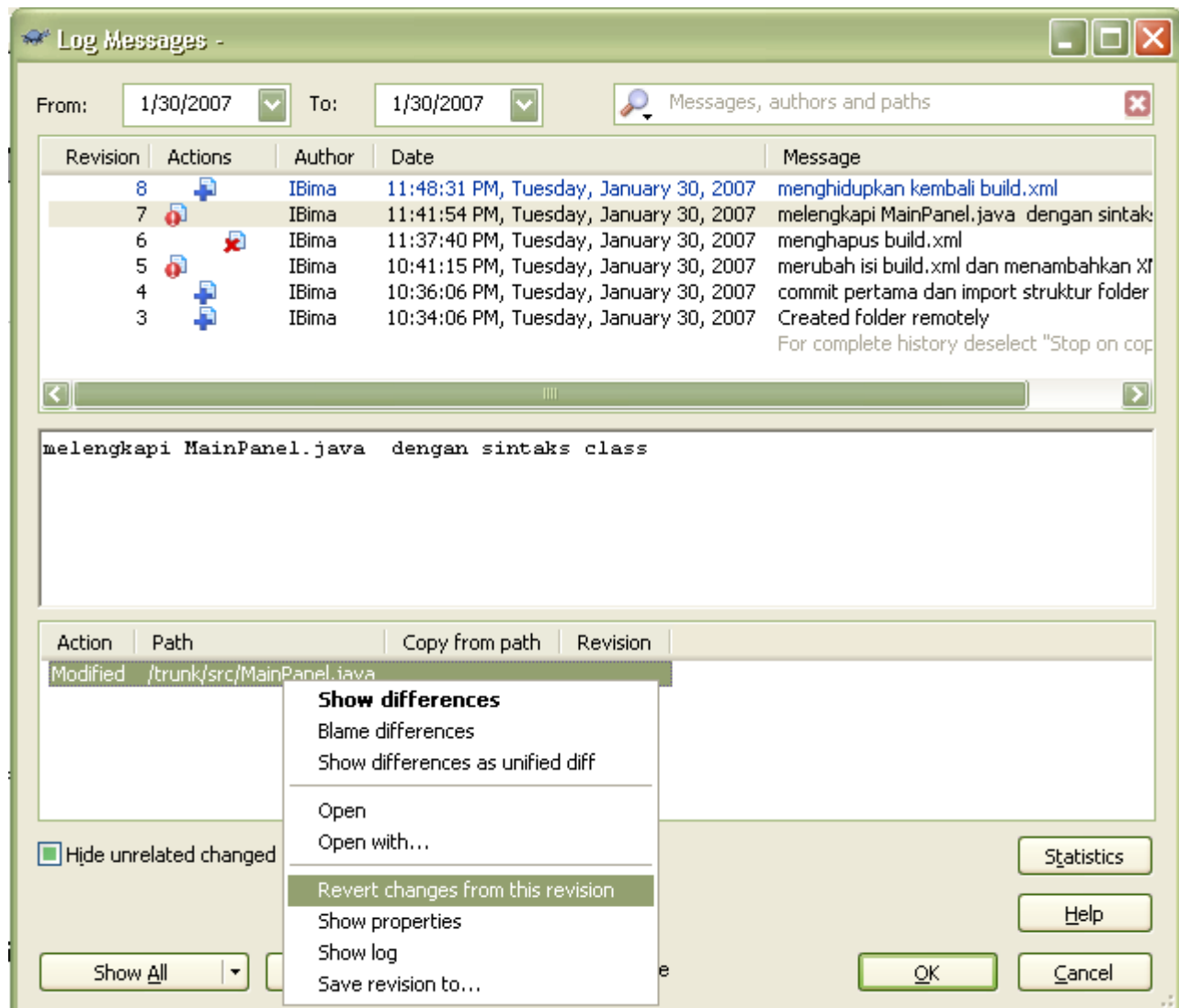
Gambar diatas memperlihatkan jendela revert. Terdapat dua file yang sudah dirubah oleh Ifnu yaitu build.xml dan MainPanel.java, Ifnu hanya perlu membatalkan perubahan di file MainPanel.java, sehingga file build.xml di-uncheck.

Mengembalikan Keadaan File ke Suatu Titik Tertentu

Kalau misalnya Ifnu sudah terlanjur mengirimkan perubahanya ke dalam repository, apakah bisa dibatalkan? Jawabanya adalah bisa. Caranya adalah dengan menggunakan perintah “Update to revision ...”. Namun anda perlu berhati-hati dengan perintah ini karena jika anda langsung memasukkan nilai revision ke dalam jendela seperti dibawah ini, yang terjadi adalah **seluruh** lokal repository akan dikembalikan ke keadaan sesuai dengan nilai revisionnya, dalam hal ini revision 5. Padahal yang kita inginkan hanya mengembalikan keadaan file MainPanel.java saja, bukan seluruh folder lokal.



Untuk mengembalikan MainPanel.java ke keadaan revision sebelumnya klik tombol Show log diatas, maka akan muncul jendela Log message seperti di bawah ini, kemudian pilih file yang akan di-revert di bagian bawah, klik kanan dan pilih menu: revert changes from this revision.



Jika kita lihat file MainPanel.java di dalam folder src, kita akan melihat bahwa icon MainPanel.java berubah dengan adanya lingkaran merah dan tanda seru di atasnya. Hal ini menandakan bahwa kita telah melakukan perubahan terhadap MainPanel.java, untuk membuat proses revert tadi menjadi permanen, seperti biasa, lakukan perintah svn commit.

Konflik

Misalnya Ifnu dan Dhiku bekerja dengan MainPanel.java secara bersamaan. Ifnu merubah MainPanel.java menjadi seperti berikut:

```
1 public class MainPanel{
2
3 public MainPanel(){
4     JFrame frame = new JFrame();
5 }
6
7 }
```

Ifnu menambahkan 4 baris program, dari baris kedua hingga baris ke enam dengan menambahkan sebuah default Constructor. Dhiku merubah MainPanel.java seperti dibawah ini:

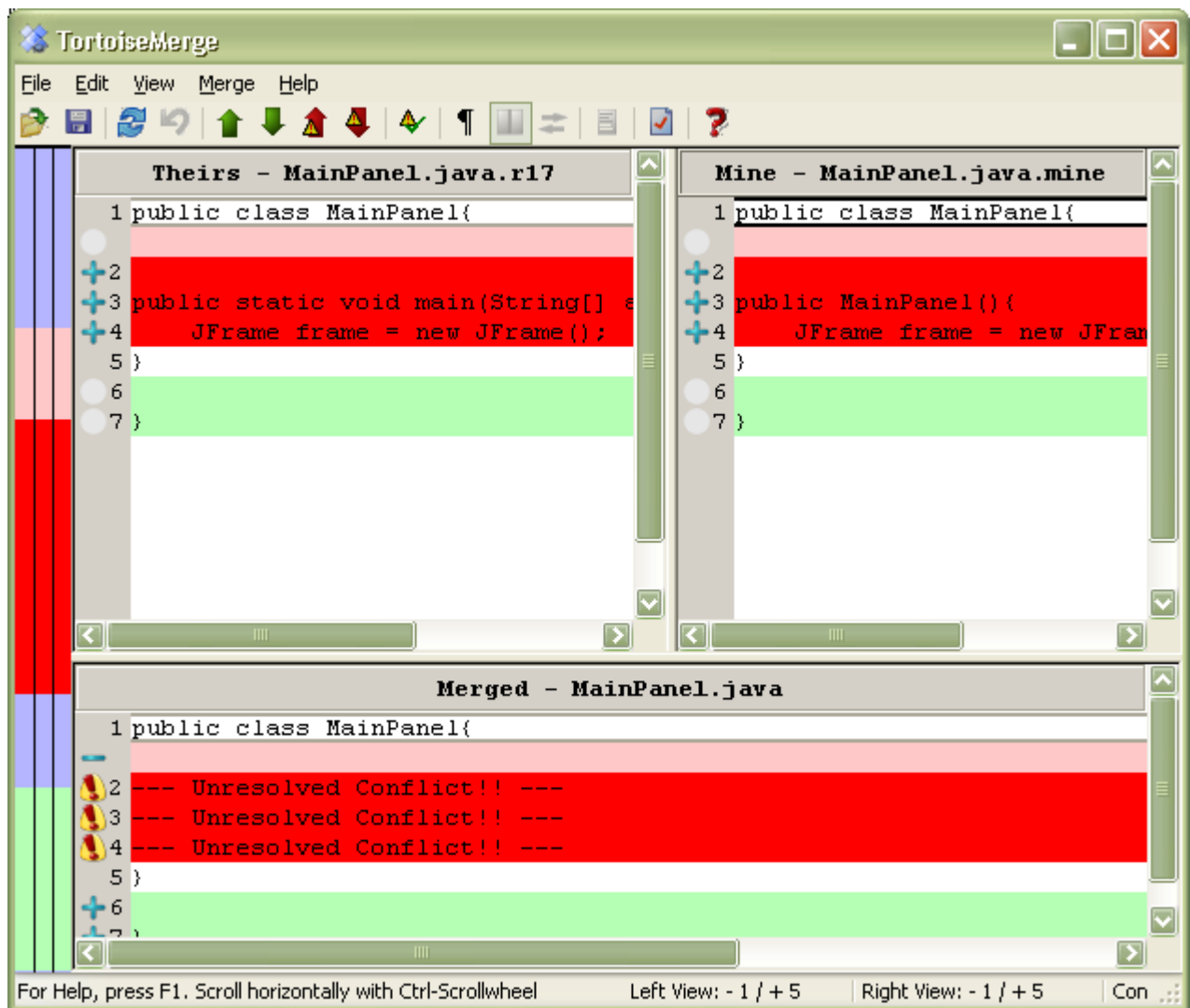
```
1 public class MainPanel{
2
3 public static void main(){
4     JFrame frame = new JFrame();
5 }
6
7 }
```

Ternyata Dhiku selesai bekerja dengan MainPanel.java dan melakukan commit terlebih dahulu, hal ini akan menaikkan nilai revision repository. Kemudian Ifnu selesai dan akan melakukan commit, ternyata Ifnu mengalami “out-of-date” error karena nilai revision folder lokal Ifnu lebih kecil dari nilai revision folder repository yang telah bertambah karena commit dari Dhiku. Untuk mengatasi hal ini Ifnu akan melaksanakan perintah SVN update agar nilai revision folder loka ifnu sama dengan nilai revision folder repository. Ifnu terkejut karena terjadi “konflik” di file MainPanel.java, dikarenakan keduanya melakukan perubahan pada bari yang sama yaitu baris 2 sampai 6.

Pemecahan Konflik

Ifnu memanggil Dhiku untuk duduk bersama mendiskusikan apa yang harus dilakukan terhadap konflik yang terjadi. Untuk melihat dimana konflik terjadi, klik kanan pada file MainPanel.java, kemudian pilih menu:

TortoiseSVN => edit conflict



Jendela TortoiseMerge diatas akan memberikan petunjuk dimana bagian yang mengalami konflik. Kita memerlukan sebuah text editor (dalam hal ini saya menggunakan Notepad++) untuk mengedit MainPanel.java.

```

1  public class MainPanel{
2  <<<<<<< .mine
3
4  public MainPanel(){
5      JFrame frame = new JFrame();
6  =====
7
8  public static void main(String[] args){
9      JFrame frame = new JFrame();
10 >>>>>>> .r17
11 }
12
13 }
14

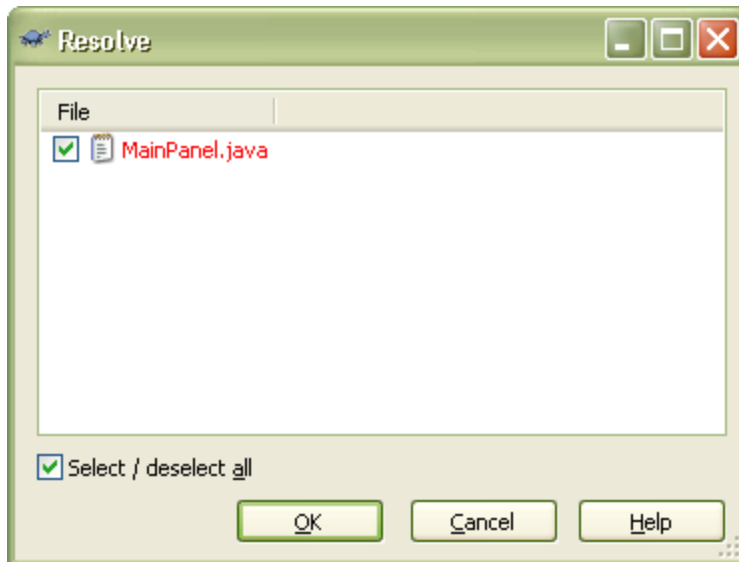
```

Perhatikan tanda <<<<<<< kemudian tanda ===== dan akhirnya tanda >>>>>>>. Tanda yang pertama (<<<<<<<) merupakan penanda awal terjadinya konflik, kode dibawahnya adalah kode program milik Ifnu. Tanda kedua (=====) adalah pemisah antara kode Ifnu dan kode Dhiku, sedangkan tanda terakhir (>>>>>>>) adalah penanda akhir daerah yang mengalami konflik.

Setelah berdiskusi Ifnu dan Dhiku sepakat untuk menggunakan kedua kode, sehingga hasil akhirnya adalah:

```
1 public class MainPanel{
2
3 public MainPanel(){
4     JFrame frame = new JFrame();
5
6 public static void main(String[] args){
7     JFrame frame = new JFrame();
8 }
9
10 }
11
```

Untuk memberitahu repository bahwa konflik sudah berhasil diatasi, gunakan perintah Resolved:



Setelah itu jangan lupa melakukan commit untuk mengirimkan perubahan ke repository.

Penutup

Version Control adalah tools yang wajib digunakan sebagai alat bantu bagi sebuah tim pengembangan software. Tanpa Version Control kode yang dihasilkan tidak dapat dikelola dengan baik dan beresiko tinggi terjangkit bug yang pada akhirnya menurunkan kualitas software yang dihasilkan.

Modul ini hanya menjelaskan sekelumit saja dari fitur yang dimiliki oleh subversion, masih banyak

fitur lain yang belum sempat kita bahas dalam modul ini, antara lain:

1. branch, digunakan untuk membuat beberapa varian aplikasi secara berbarengan. Misalnya aplikasi untuk client A sedikit berbeda dengan client B karena ada fitur tambahan.
2. tag, digunakan untuk menandai pengembangan software pada titik penting, misalnya release software ke QC, rilis software ke user untuk dilakukan UAT, rilis ke lingkungan produksi, naik versi, peninjauan arsitektur dan sebagainya.
3. Merge, menggabungkan perubahan yang dilakukan dalam branch ke dalam trunk. Hal ini dilakukan karena fitur tambahan yang diberikan kepada client B sangat berguna dan kita tertarik untuk menggabungkannya dengan kode utama.
4. hook script, subversion mempunyai banyak sekali event seperti commit, update, merge, mkdir dan lain-lain. Ada kalanya kita ingin melakukan beberapa kegiatan tambahan ketika event-event tersebut terjadi. Misalnya mengirimkan email kepada semua anggota tim karena ada salah satu dari mereka yang melakukan commit. Kegiatan tambahan ini bisa kita letakkan kedalam sebuah program yang disebut Hook Script.
5. subversion untuk non programmer, Subversion mampu menangani file berbentuk text maupun binary. Kemampuan subversion menangani file binary memberikan kemungkinan kepada anggota tim non-programmer untuk menggunakan subversion. Misalnya technical writer dapat menggunakan subversion untuk menyimpan file dokumentasi, atau Project Manager menggunakan subversion untuk menyimpan file jadwal pengembangan perangkat lunak.
6. instalasi dan kustomisasi subversion, Subversion mampu berjalan dalam beberapa protokol sekaligus, antara lain: file,svn,svn+ssh dan http. Proses instalasi dan kustomisasinya tidaklah segampang menginstal TortoiseSVN, perlu langkah-langkah yang cukup memusingkan dan pembahasan yang lebih mendalam.

Referensi

1. “konsep dan penggunaan subversion”, Endy Muhardin, Endy.muhardin@gmail.com
2. “Subversion Documentation”, <http://subversion.tigris.org>
3. “TortoiseSVN Documentation”, <http://tortoisesvn.togris.org>
4. “Version Control Menggunakan Subversion”, Endy Muhardin, <http://endy.artivisi.com>, Java Meet-Up 07.01