Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

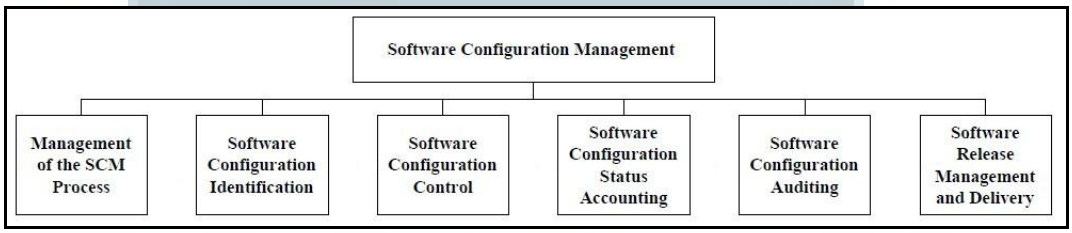
Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

LANDASAN TEORI

2.1 Software Configuration Management

Software Configuration Management (SCM) merupakan proses mengidentifikasi konfigurasi sistem pada suatu waktu tertentu dengan tujuan mengendalikan perubahan konfigurasi dan menjaga integritas suatu konfigurasi dalam sebuah siklus pengembangan sistem (Scott dan Nisse, 2001). *SCM* menawarkan solusi untuk menyelesaikan permasalahan perubahan yang tidak terstruktur dengan baik sehingga kode sumber yang dihasilkan menjadi berantakan dan tidak terkontrol. Dengan menggunakan *SCM*, diharapkan pengembang piranti lunak tidak perlu lagi menghabiskan banyak waktu untuk melacak perubahan yang terjadi pada kode sumbernya.



Gambar 2.1 Rincian *Software Configuration Management*.
(Scott dan Nisse, 2011)

Gambar 2.1 merupakan rincian dari proses SCM secara umum. SCM dibagi ke dalam enam tahapan yang dapat membantu pengembang untuk mengontrol perubahan yang terjadi.

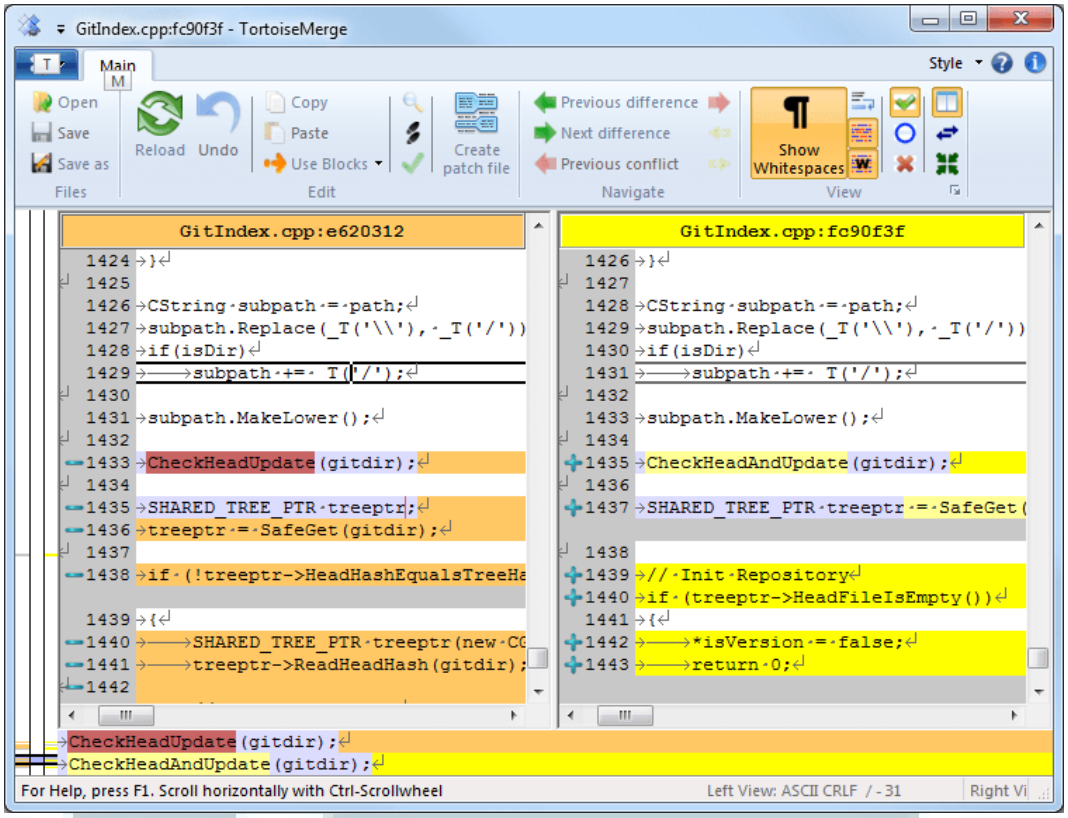
2.2 Data Comparison

Data Comparison program membuat daftar perbedaan di antara dua buah *file* (data). Perbedaan-perbedaan tersebut dapat dikemas dalam bentuk baris perbaris, dengan menunjukan baris mana yang harus ditambahkan, dihapus, atau dipindahkan untuk membuat *file* pertama berubah menjadi *file* kedua (Myres, Miller, 1985). *Data comparison* secara sederhana melihat dua (atau lebih) data yang diberikan, kemudian membandingkannya satu sama lain dan mengeluarkan hasil berupa perbedaan yang ada di antara data-data yang diberikan tadi. *Data comparison* merupakan bagian dari *SCM* dan berguna untuk melihat perubahan pada kode sumber yang terjadi secara kasat mata.

2.3 Diff Algorithm

```
121  125          userProfile.setUserEmail(et_email.getText().toString());
122  126          userProfile.setUserMobile(et_phone.getText().toString());
123  127
124       -       String jsonString = Config.getGson().toJson(userProfile);
     128  +       QGeotagInfo geotagInfo = new QGeotagInfo();
     129  +       geotagInfo.geoLat = "";
     130  +       userProfile.setGeotagId( geotagInfo );
     131  +
     132  +       String jsonString = Config.getGson(QGeotagInfo.class,new QGeotagInfoCreator())
     133  +               .toJson(userProfile);
     134  +       Log.i( "SignupActivity","json:" + jsonString );
125  135
126  136          // make the http response handler to handle the "after upload".
127  137          HttpResponseHandler httpResponseHandler = new HttpResponseHandler() {
```

Gambar 2.2 Contoh tampilan *diff* dari GitLab v7.6.2 (GitLab B.V, 2011)



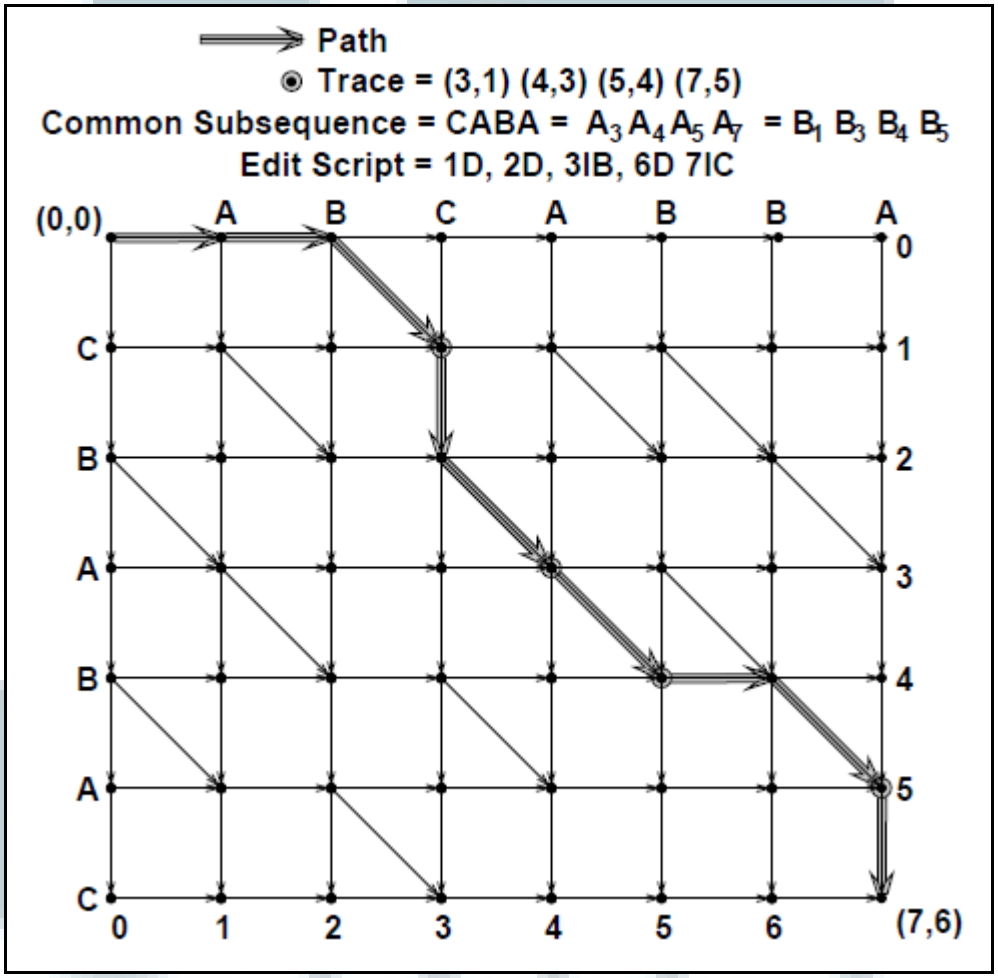
Gambar 2.3 Contoh tampilan *diff* dari TortoiseGit v1.8.11.0 (TortoiseGit Dev, Tanpa Tahun)

Secara umum, *diff algorithm* memiliki dasar penyelesaian dari *Longest Common Subsequence (LCS) problem*. *Diff algorithm* mencari baris yang tidak berubah diantara *file* yang diberikan. Sebuah program *diff* akan membuat daftar dari apa saja yang berubah dari suatu *file* dengan *file* lain dan juga sebaliknya. (Hunt, McIlroy, 1976)

Terdapat beberapa Algoritma *diff* yang lumayan terkenal, diantaranya adalah *Hunt McIlroy algorithm* oleh James W. Hunt dan M. Douglas McIlroy, *patience diff algorithm* oleh Bram Cohen, dan *Myers diff algorithm* oleh Eugene W. Myers.

2.4 Myers diff Algorithm

Myres diff algorithm merupakan sebuah algoritma yang menerapkan *diff algorithm* untuk mencari perbedaan dari dua buah data yang diberikan. Diterbitkan pada sebuah *paper* dengan judul "*An O(ND) Difference Algorithm and Its Variations*" pada tahun 1986 di U.S.A. Sama seperti *diff algorithm* lainnya, *Myers diff algorithm* juga berdasarkan *Longest Common Subsequence problem*, dengan mencari perbedaan terkecil yang terjadi antara *file* satu dengan *file* dua.



Gambar 2.4 Grafik Perubahan
(Myers, 1986)

Myers diff algorithm merepresentasikan perubahan yang terjadi antara dua buah data ke dalam sebuah grafik yang dapat dilihat pada gambar 2.4. Gambar 2.4 menunjukan perbandingan antara data pertama *abcabba* dengan data kedua *cbabac*. Hasil dari perbedaan kedua buah data dapat dilihat dari jalur yang terbentuk dari titik (0,0) sampai titik (7,6).

2.4.1 Greedy LCS/SES

```
Constant MAX ∈ [0,M+N]

Var V: Array [–MAX .. MAX] of Integer

V[1] ← 0
For D ← 0 to MAX Do
    For k ← –D to D in steps of 2 Do
        If k = –D or k ≠ D and V[k – 1] < V[k + 1] Then
            x ← V[k + 1]
        Else
            x ← V[k – 1]+1
        y ← x – k
        While x < N and y < M and a_{x+1} = b_{y+1} Do  (x,y) ← (x+1,y+1)
        V[k] ← x
        If x ≥ N and y ≥ M Then
            Length of an SES is D
            Stop
Length of an SES is greater than MAX
```

Gambar 2.5 *Pseudocode* untuk Algoritma *Greedy LCS/SES*
(Myers, 1986)

Gambar 2.5 merupakan *pseudocode* yang menunjukan algoritma Myers dengan menggunakan *Greedy LCS/SES*. LCS sendiri merupakan singkatan dari *Longest Common Subsequence*, sedangkan SES memiliki kepanjangan *Shortest*

Edit Script. Pada algoritma ini, dilakukan pengecekan dari depan ke belakang. Algoritma ini akan bergerak dari awal data menuju ke akhir data secara linear.

2.4.2 Linear Space Refinement

```
Δ ← N–M
For D ← 0 to ⌈(M+N)/2⌉ Do
    For k ← –D to D in steps of 2 Do
        Find the end of the furthest reaching forward D-path in diagonal k.
        If Δ is odd and k ∈ [Δ–(D–1), Δ+(D–1)] Then
            If the path overlaps the furthest reaching reverse (D–1)-path in diagonal k Then
                Length of an SES is 2D–1.
                The last snake of the forward path is the middle snake.
    For k ← –D to D in steps of 2 Do
        Find the end of the furthest reaching reverse D-path in diagonal k+Δ.
        If Δ is even and k+Δ ∈ [–D,D] Then
            If the path overlaps the furthest reaching forward D-path in diagonal k+Δ Then
                Length of an SES is 2D.
                The last snake of the reverse path is the middle snake.
```

Gambar 2.6 Algortima Myers Diff dengan *Linear Space Refinement* (Myers, 1986)

Gambar 2.6 merupakan *pseudocode* dari algoritma *Myers Diff* yang sudah ditambahkan dengan *Linear Space Refinement*. Algoritma ini juga memiliki tujuan untuk mencari jalur di grafik perubahan dari titik (0,0) ke titik paling akhir secara diagonal. Namun, berbeda dengan algoritma *Greedy LCS/SES* pada gambar 2.5, algoritma ini akan mengecek perbedaan dari dua buah data dari depan dan juga dari belakang secara pararel.

2.5 User Acceptance Test

Menurut Naik dan Tripathy (2008), *user acceptance test* adalah proses *testing* formal yang dilakukan untuk mencari tahu apakah sistem yang telah dibuat sudah memenuhi kriteria penerimaan (*acceptance criteria*) yang sudah ditentukan

oleh *user*. *User Acceptance Test* dapat membantu *user* untuk menentukan apakah *user* tersebut dapat menerima sistem dengan baik atau tidak.

Terdapat 19 kriteria penerimaan yang dijelaskan oleh Naik dan Tripathy. Kriteria tersebut adalah.

1. Functional Correctness and Completeness
2. Accuracy
3. Data Integrity
4. Data Conversion
5. Backup and Recovery
6. Competitive Edge
7. Usability
8. Performance
9. Start-up Time
10. Stress
11. Reliability and Availability
12. Maintainability and Serviceability
13. Robustness
14. Timeliness
15. Confidentiality and Availability
16. Compliance
17. Installability and Upgradability
18. Scalability
19. Documentation

2.6 Populasi

Menurut Sutrisno Hadi, populasi adalah seluruh penduduk yang akan dilakukan pengujian. Populasi didefinisikan sebagai sejumlah penduduk atau individu yang paling sedikit memiliki satu sifat yang sama.

Menurut M. Nazir, ada dua cara untuk mengumpulkan populasi. Kedua cara tersebut adalah.

1. Sensus (*complete enumeration*) digunakan jika semua anggota populasi dimasukan sebagai responden untuk diteliti. Dengan kata lain menanyakan semua anggota populasi satu persatu.
2. Sampel (*sample survey* atau *sample enumeration*) mengambil beberapa anggota populasi untuk dijadikan responden untuk diteliti.