HYPERBOLIC TREE MENGUNAKAN FISHEYE VIEW SEBAGAI ALTERNATIF SISTEM NAVIGASI FILE

Eddy Triswanto Setyoadi, ST., M.Kom.*
Anggya N.D. Soetarmono, S.Kom.*

ABSTRAK

Pada masa sekarang hampir semua aplikasi yang memiliki fitur navigasi menggunakan model *tree diagram* dengan *indented layout*, yaitu *tree* yang terekspansi ke kanan secara terus-menerus untuk menunjukkan hubungan antara file induk dengan file anak. Contoh yang paling mudah dan banyak dikenal orang adalah sistem *explore* folder pada Windows.

Kata Kunci : Tree Diagram, Indented Layout

1. PENDAHULUAN

1.1. Latar Belakang

Ketika sebuah folder (data induk) dipilih oleh user maka di bagian bawah dari folder tersebut akan muncul subfolder (data anak). Sistem ini hanya efektif ketika sebuah subfolder tidak memiliki subfolder lagi dalam jumlah yang banyak. Hal ini menimbulkan masalah yaitu, ketika sebuah subfolder memiliki subfolder lagi dan seterusnya, maka *user* akan mulai merasakan kebingungan dalam melakukan navigasi.

Dengan menerapkan *hyperbolic tree* masalah di atas dapat teratasi. Dalam *hyperbolic tree*, data yang menjadi data anak akan memiliki kedudukan yang sejajar dengan data induknya sehingga dapat dilihat asal-usulnya. Selain itu *hyperbolic tree* bisa menampilkan badan dari *tree* secara kesuluruhan. *Hyperbolic tree* menjadi lebih sempurna dengan menerapkan *fisheye view*. Dengan *fisheye view*, data yang dipilih oleh *user* akan diletakkan di pusat bidang sehingga *user* bisa fokus di satu bagian saja. Tampilan sistem navigasi yang menerapkan *hyperbolic tree* dengan *fisheye view* akan membantu user dalam memahami dan mengingat hubungan antar data.

1.2. Perumusan Masalah

1. Bagaimana membuat *hyperbolic tree*
2. Bagaimana menerapkan *fisheye view* dalam *hyperbolic tree*
3. Algoritma fisheye view yang dibahas adalah algoritma SHriMP

1.3. Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk meneliti cara mengembangkan sebuah *Human Computer Interface* dari sistem navigasi *file* berdasarkan *hyperbolic tree* dan *fisheye view*. Manfaat dari penelitian ini adalah sebagai acuan dalam merancang sebuah *Human Computer Interface* yang diharapkan dapat :

1. Memudahkan user dalam navigasi pada aplikasi yang memiliki banyak data
2. *User* menjadi lebih memahami hubungan antar data

* Staf Pengajar Program Studi S1-Sistem Informasi IKADO
* Staf Pengajar Program Studi S1-Sistem Informasi IKADO

3. Menjadi alternatif dari sistem yang konvensional dalam melakukan navigasi

2. LANDASAN TEORI

Pada masa-masa awal komputer ditemukan, tampilan yang ada masih tidak nyaman bagi *user*, terutama *user* yang tidak memiliki latar belakang pendidikan komputer[1]. Tampilan di layar masih berupa text dan pemberian perintah harus diketikkan secara manual oleh *user*. Hal yang sama juga berlaku untuk navigasi file, tidak ada *icon* atau gambar yang bisa membantu user dalam melakukan kegiatannya. *User* dipaksa untuk mengingat semua *syntax* supaya bisa melakukan aksi tertentu, bahkan aksi sederhana seperti *copy* atau *delete*. Oleh karena itu dicarilah sebuah cara supaya pemberian perintah kepada komputer bisa menjadi lebih praktis dan lebih mudah dioperasikan oleh orang awam.

2.1. Human Computer Interaction

Human Computer Interaction merupakan cabang dari ilmu komputer yang secara khusus mempelajari mengenai interaksi antara komputer dengan manusia atau *user*. Konsep awal dari *Human Computer Interaction* pertama kali dikemukakan oleh Douglas Englebart dan Alan Kay[2]. Douglas Englebart dikenal sebagai penemu dari mouse sedangkan Alan Kay merupakan seorang desainer yang mendesain tampilan awal dari tampilan window di layar komputer[3]. Kedua orang inilah yang meletakkan dasar dari tampilan sistem operasi dan cara berinteraksi dengan aplikasi komputer saat ini.Tujuan utama dari cabang ilmu ini adalah memudahkan manusia sebagai *user* dari komputer, dalam mengoperasikan komputer tersebut. Selain itu, komputer juga menjadi lebih responsif terhadap input – input yang diberikan oleh *user*.

Salah satu hasil dari usaha tersebut adalah sistem navigasi yang berfungsi membantu *user* dalam melihat dan mengatur data-data yang ada di dalam komputernya. Sistem navigasi dalam berbagai macam sistem operasi seperti Windows, Linux, maupun Macintosh pada umumnya menggunakan *tree diagram+window* untuk mempresentasikan hubungan antar folder atau data. Window yang terletak di sebelah kanan dari tree diagram akan menampilkan detil dari isi folder. Mouse digunakan untuk melakukan navigasi dalam *tree diagram* tersebut[4]. *User* akan memilih folder dari *tree diagram* yang ingin dilihatnya dan apabila bagian tersebut memiliki folder anak, maka dipresentasikan dalam bentuk ekspansi *subtree* ke arah kanan bawah dari folder induk. Dalam penggunaan umum atau sehari-hari, sistem navigasi *tree diagram* memang menjadi solusi yang terpopuler sampai saat ini. Namun sistem ini kurang efektif dalam memberikan gambaran hubungan antar data, sehingga *user* akan kesulitan dalam mengingat letak dari data – data tertentu, terutama apabila *user* tersebut memiliki data dalam jumlah yang sangat banyak. Untuk aplikasi yang memiliki ciri-ciri tersebut perlu digunakan cara pandang lain yang dapat membantu *user* dalam melakukan navigasi.

---

[1] *User* adalah orang-orang yang menggunakan atau berinteraksi dengan software di dalam komputer

[2] http://en.wikipedia.org/wiki/Douglas_Englebart
http://en.wikipedia.org/wiki/Alan_Kay

[3] Contoh dari tampilan *window* paling mudah dilihat pada sistem operasi Windows, dimana hampir setiap aksi *user* akan memunculkan *window* tertentu

[4] Sesuai namanya, *tree diagram* merupakan cara menampilkan data dalam bentuk *tree*. Keterangan lebih lanjut mengenai *tree diagram* bisa dilihat pada bab III

2.2. Cara Pandang Sistem Navigasi

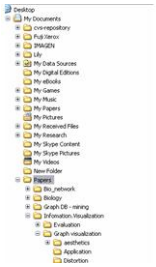
Seperti diuraikan di atas, sistem navigasi yang paling umum digunakan adalah *tree diagram+window***.** Sistem navigasi ini menerapkan cara pandang yang disebut *multiple view*. Ada banyak cara pandang dalam melakukan navigasi dan secara garis besar dibedakan dapat menjadi 3 cara, yaitu:

1. Pan + Zoom View

   Cara pandang ini menerapkan tampilan yang menyerupai peta. Data ditampilkan dalam satu *window* saja dan hubungan antar data digambarkan dalam bentuk garis-garis yang saling menghubungkan data tersebut. Apabila user ingin melihat detil tertentu dari suatu data, user bisa memperbesar tampilan yang ingin dilihatnya dengan menggerakkan *mouse*. Hal ini menjadi masalah karena daya pandang *user* hanya terbatas pada daerah yang diperbesar saja sehingga tidak bisa melihat semua data sekaligus.

2. Multiple view

   Data yang berjumlah banyak ditampilkan dalam bentuk *tree diagram* di *window* yang terpisah sehingga user bisa memilih salah satu data yang ingin ditelitinya lebih lanjut. Isi dari data tersebut akan ditampilkan secara lebih detil dalam *window* lain yang terletak di sebelahnya. Masalah yang sering terjadi dalam cara pandang ini adalah *user* menjadi kesulitan dalam mengingat dan memahami hubungan antar data yang ada. Hal ini disebabkan karena hanya sebagian saja dari *tree diagram* yang bisa ditampilkan ke layar dengan cara ini. *Tree diagram* menjadi boros tempat dan tidak efektif. Gambar 2.1. menunjukkan tampilan dari *tree diagram* dengan *indented layout*[5] pada sistem operasi Windows.

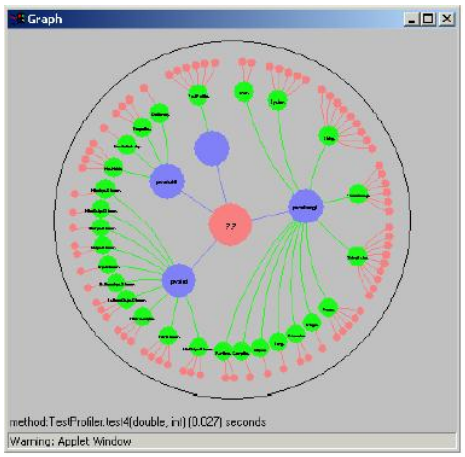


3. Context + Detail view

   Seperti cara pandang sebelumnya, data diletakkan dalam sebuah bidang dalam satu window saja. Data dipresentasikan dalam bidang tertentu dan data dilihat dengan menerapkan teknik distorsi, dimana data atau sekumpulan data yang menjadi fokus utama user ukurannya akan diperbesar. Hubungan antar data digambarkan dalam bentuk garis. *Hyperbolic tree* dengan *fisheye fiew* menerapkan cara pandang ini. *Tree diagram* yang diproyeksikan dalam bidang *hyperbolic* bisa dilihat secara keseluruhan. Menggunakan algoritma *fisheye view*, data yang tidak menjadi fokus utama akan secara otomatis bergeser ke samping dan terlihat seakan - akan menjauh dari pandangan. Dengan cara ini user bisa memahami hubungan antar data sekaligus melihat detil dari data yang menjadi fokus utama. Pada gambar dibawah terlihat bahwa data yang menjadi fokus utama diletakkan di tengah layar seakan-akan data

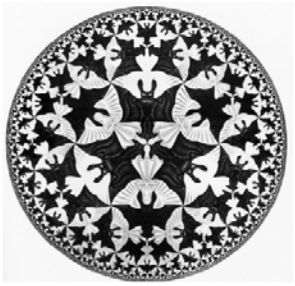
[5] *tree* yang terekspansi ke kanan secara terus-menerus untuk menunjukkan hubungan antara data induk dengan data anak

tersebut terletak paling dekat dengan *user* sedangkan data tetangganya terkesan semakin menjauh.

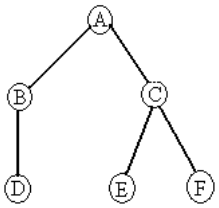


3. HYBERBOLIC TREE

Hyperbolic tree merupakan salah satu teknik visualisasi dari data. Ide awal mengenai *hyperbolic tree* ini terinspirasi sebuah karya seni berjudul ”Circle Limit IV” [6] oleh M.C. Escher yang bisa dilihat pada gambar dibawah ini.



Hal ini dijelaskan dalam karya tulis yang dibuat oleh J.Lamping, Ramana Rao, dan Peter Pirolli, ”A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Geometry”. Setelah melihat karya seni tersebut mereka mendapat ide untuk memproyeksikan *tree diagram* ke dalam bidang *hyperbolic*. *Tree diagram* merupakan cara umum untuk memvisualisasikan data, dimana data-data beserta kedudukannya digambarkan dalam bentuk pohon. Secara umum, data yang terletak di atas merupakan induk dari data yang terletak di bawahnya seperti bisa dillihat pada gambar 3.2.

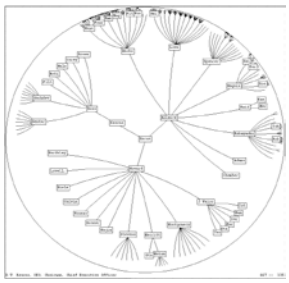


[6] Karya seni ini menggunakan Conformal Disc Model yang merupakan salah satu model dari bidang hyperbolic. Keterangan lebih lanjut mengenai hal ini bisa dilihat di www.wikipedia.org/hyperbolic_model

Gambar di atas menunjukkan bahwa titik A atau node[7] A mempunyai kedudukan yang paling tinggi. Dari sini bisa dikatakan bahwa node A merupakan *parent* dari node B dan C. Node B merupakan *child* dari node A dan *parent* dari node D dan demikian seterusnya. Sebenarnya ada berbagai macam jenis *tree diagram* dengan posisi node *parent* yang berbeda-beda. Pada *hyperbolic tree,* node *parent* diletakkan di bagian tengah dari bidang *hyperbolic*.

Tree diagram yang diproyeksikan ke bidang *hyperbolic* inilah yang kemudian dinamakan *hyperbolic tree* dan digunakan sebagai cara alternatif untuk memvisualisasikan struktur data. Dengan menggunakan bidang *hyperbolic, tree diagram* yang memiliki banyak cabang bisa ditampilkan secara keseluruhan pada satu layar monitor yang ukurannya terbatas. Dalam karya tulis mereka, J.Lamping, Ramana Rao, dan Peter Perolli menyatakan bahwa sebuah *tree diagram* standar dalam *window* - berukuran 600x600 pixel hanya bisa menampilkan 100 node saja. *Tree diagram yang* diproyeksikan dalam bidang *hyperbolic* bisa menampilkan sekitar 1000 node dalam window berukuran sama.

Gambar 3.3. menunjukkan *tree diagram* yang diproyeksikan ke dalam bidang *hyperbolic*. Node yang menjadi *parent* terlihat diletakkan di bagian tengah dari bidang.

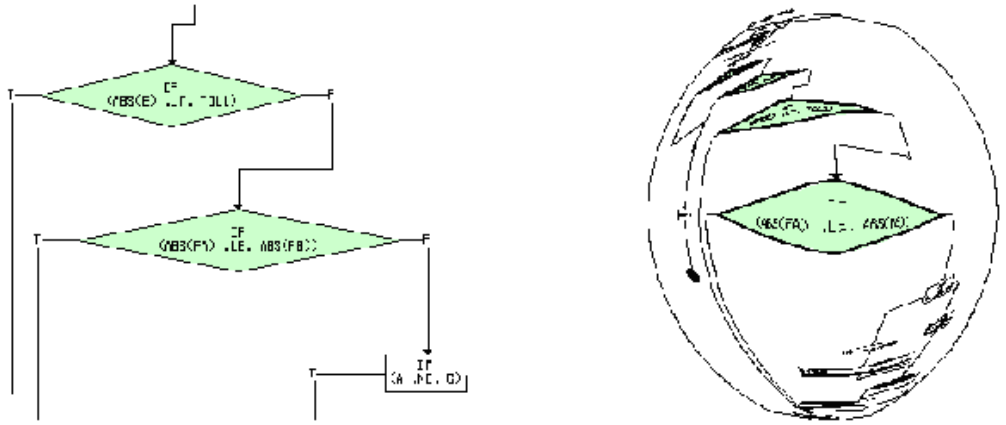


4. METODE PENELITIAN

4.1. Fish Eye View

Istilah *fisheye view* pertama kali diperkenalkan oleh George W. Furnas dalam karya tulisnya ”Generalized Fisheye Views”, dengan berdasarkan ciri khusus dari lensa mata ikan dimana objek yang terletak di titik fokus lensa akan terlihat lebih besar. Supaya data yang menjadi pusat perhatian dari *user* dapat dibedakan dari *file* anaknya, diperlukan *fisheye view*. Dengan menggunakan *fisheye view* data yang terletak ditengah bidang, dengan kata lain terletak di titik fokus, terlihat membesar. Data yang tidak menjadi fokus utama akan terlihat menjauhi titik fokus dan ukurannya mengecil seperti terlihat pada gambar dibawah ini.

[7] Node merupakan istilah yang merujuk pada tiap data dari *tree diagram*



linear view fish eye view

4.2. Algoritma SHriMP

Algoritma *fisheye view* yang akan dibahas adalah SHriMP (*Simple Hierarchical Multi Perspective*). SHriMP merupakan algoritma yang fleksibel sehingga dapat diubah sesuai kebutuhan untuk berbagai macam jenis tampilan. Dengan algoritma ini skala dari data yang menjadi fokus utama akan diperbesar. Sedangkan data yang terletak bersebelahan dengannya akan mengecil dan bergeser ke samping.

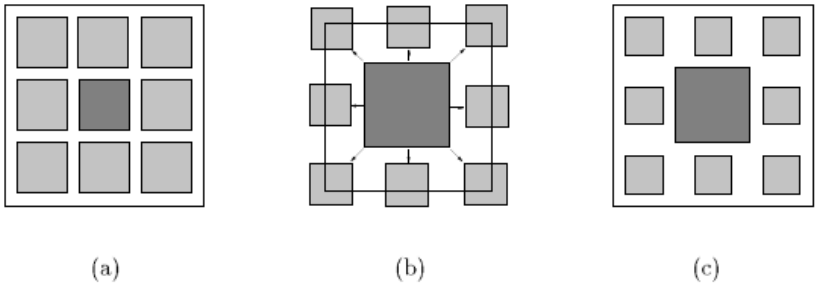
Algoritma ShriMP dijelaskan oleh Margaret-Anne D. Storey, F.David Fracchia, dan Hausi A. Müller dalam paper mereka[8]. Sebelum data diproyeksikan ke bidang *hyperbolic*, data-data tersebut perlu dikenai algoritma ShriMP supaya bisa memiliki tampilan *fisheye view*. Dalam penjelasan berikut akan digunakan istilah node sebagai perwakilan dari data[9]. Perlu diketahui bahwa tampilan pada gambar diatas(b) tidak bisa dilihat oleh *user* sehingga seakan-akan node pusat membesar dan node samping bergeser memberi tempat.

Gambar diatas menunjukkan cara kerja algoritma ShriMP dimana sebuah node ukurannya membesar dan menggeser node-node lain ke arah samping. Gambar 4.2.(a) menunjukkan keadaan sebelum node pusat diperbesar, ada kotak pembatas yang mengelilingi semua node tersebut[10]. Gambar diatas (b) menunjukkan bahwa node-node tersebut dikenai algoritma ShriMP dimana node tengah ukurannya membesar dan mendorong node-node samping ke arah luar, kotak pembatas seakan-akan tidak ada. Gambar diatas (c) merupakan hasil akhir dimana skala dari semua node disesuaikan dengan kotak pembatas, sehingga bisa diletakkan dalam kotak pembatas seperti semula.

[8] Paper mereka berjudul "Customizing a Fisheye View Algorithm to Preserve the Mental Map"

[9] Node berbentuk kotak digunakan supaya pembaca lebih mudah memahami proses.

[10] Kotak pembatas tersebut merupakan kotak imajiner yang hanya ada untuk keperluan penghitungan. Kotak pembatas ini sama dengan *containment circle* pada bab III

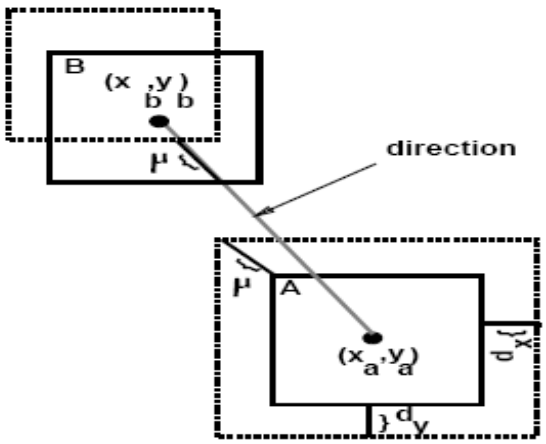


Ilustrasi Pembesaran Tampilan dari Node yang Dipilih

4.3. Penghitungan Vektor Translasi (T)

Ketika sebuah node pusat dikelilingi oleh banyak node samping, perlu diusahakan supaya posisi akhir dari kumpulan node tersebut memiliki kemiripan dengan posisi sebelum dikenai *fisheye view*. Supaya hal tersebut bisa terjadi perlu ditetapkan sebuah vektor translasi (T). Vektor translasi ini akan digunakan untuk menentukan seberapa jauh pergeseran dari node samping setelah node pusat ukurannya diperbesar. Untuk melakukan hal itu perlu ditarik sebuah garis lurus dari titik tengah node pusat ke titik tengah semua node samping. Vektor translasi (T) diterapkan pada koordinat yang dilalui oleh garis lurus tersebut.

Pada gambar dibawah node A ukurannya membesar dan mendorong node B ke arah luar. Koordinat dari titik pusat node A adalah (x_a,y_a) dan koordinat dari titik pusat node B adalah (x_b,y_b). Dx dan dy merupakan nilai selisih dari jarak garis samping awal dengan garis samping dari ukuran node A yang telah diperbesar.



Ilustrasi Perpindahan Posisi Node B Ketika Node A Diperbesar

Rumus yang diterapkan untuk mengukur vektor translasi (T) adalah sebagai berikut :

$$\mu = \sqrt{d_x^2 + d_y^2}$$

$$\mathbf{T}_x = \mu \frac{x_b - x_a}{\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}}$$

$$\mathbf{T}_y = \mu \frac{y_b - y_a}{\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}}$$

Contoh penerapan :
Misalkan (d_x ,d_y)= (3,3); (x_a ,y_a)= (0,0); (x_b,y_b)= (10,10)

$$\mu = \sqrt{3^2 + 3^2}$$

$$\mathbf{T_x} = 4{,}24 * \frac{10 - 0}{\sqrt{(10-0)^2 + (10-0)^2}} = 2{,}968$$

$$\mathbf{T_y} = 4{,}24 * \frac{10 - 0}{\sqrt{(10-0)^2 + (10-0)^2}} = 2{,}968$$

4.4. Perubahan posisi node menggunakan Algoritma SHriMP

Skala dari perbesaran ukuran node pusat bisa ditentukan dengan bebas oleh *programmer*. Hal inilah yang menyebabkan algoritma SHriMP bersifat fleksibel. Perubahan ukuran dari node pusat akan menyebabkan posisi dari node samping bergeser ke arah luar menembus kotak pembatas. Pergeseran ini terjadi dengan menerapkan rumus berikut :

$$x' = x_p + s\,(x + \mathbf{T}_x - x_p)$$

$$y' = y_p + s\,(y + \mathbf{T}_y - y_p)$$

Contoh penerapan :
Misalkan: (x_p,y_p) = (0,0); (x,y) = (10,10); s = 2

x’ = 0 + 2 (10 + 2,968 – 0) = 25,936
y’ = 0+ 2 (10 + 2,968 – 0) = 25,936

Untuk mengembalikan node ke posisi semula, digunakan rumus berikut :

$$x = (x' - x_p)/s + x_p - \mathbf{T}_x$$

$$y = (y' - y_p)/s + y_p - \mathbf{T}_y$$

Contoh penerapan :

x = (25,936 – 0) / 2 + 0 – 2,968 = 10
y = (25,936 – 0) / 2 + 0 – 2,968 = 10

- Koordinat awal dari node samping → (X,Y)
- Koordinat baru dari node samping → (X’,Y ’)
- Koordinat titik pusat → (Xp,Yp)
- Tx dan Ty merupakan vektor translasi
- s adalah hasil bagi dari rasio ukuran kotak pembatas semua node sesudah pemberian vektor translasi T dengan sebelum pemberian vektor translasi T

5. Penutup

5.1. Kesimpulan

- Sistem navigasi yang menerapkan *hyperbolic tree* dengan *fisheye view* sangat cocok digunakan dalam aplikasi yang menganggap hubungan antar data merupakan faktor penting yang perlu diperhatikan.

- Jumlah data yang mampu ditampilkan secara bersamaan dalam satu layar oleh *hyperbolic tree* dengan *fisheye view* berjumlah jauh lebih banyak dari *tree diagram* biasa.
- Sebagian besar *user* mengalami kesulitan dalam melakukan navigasi dengan *hyperbolic tree* karena telah terbiasa menggunakan *tree diagram* dengan *indented layout* pada sistem operasi Windows.
- Kelemahan utama dari *hyperbolic tree* dengan *fisheye view* adalah detil menjadi tidak jelas ketika data yang ditampilkan berjumlah sangat banyak (<1000 node).

6. Daftar Pustaka

Eklund, Peter, Nataliya Roberts, dan Steve Green. **Ontorama, Browsing RDF Ontologies using a Hyperbolic-Style Browser.** Australia : The University of Queensland, St Lucia QLD 4072.

Lamping, J. dan Ramana Rao. (1995). **The Hyperbolic Browser : A Focus+Context Technique for Visualizing Large Hierarchies**. Palo Alto : Xerox Palo Alto Reseach Center.

Pavlo, Andy (2006). **Interactive, Tree Based Graph Visualization.** E-book.

Storey, Margaret Anne D., F.David Fracchia, dan Hausi A. Müller. (1999). **Customizing a Fisheye View Algorithm to Preserve the Mental Map**. E-book.

Storey, Margaret Anne D., F.David Fracchia, Hausi A. Müller, dan Kenny Wong. (1999). **On Integrating Vizualization Technique for Effective Software Exploration.** E-book.