**PENGEMBANGAN KAKAS BERBASIS WEB**

**VISUALISASI PROGRAM DENGAN DETEKSI ALGORITMA**

**PROPOSAL TESIS**

**Karya tulis sebagai salah satu syarat**

**untuk memperoleh gelar Magister dari**

**Institut Teknologi Bandung**

**Oleh**

**HABIBIE ED DIEN**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

****

**INSTITUT TEKNOLOGI BANDUNG**

**September 2017**

**METODOLOGI PENGEMBANGAN KAKAS**

**VISUALISASI PROGRAM UNTUK GRAF BERBASIS WEB**

**DENGAN MEMANFAATKAN PUSTAKA *D3JS***

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

Tanggal September 2017

|  |  |
| --- | --- |
| Pembimbing Pertama  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Dr. Ir. M.M. Inggriani) | Pembimbing Kedua  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Yudistira Dwi W. Asnar, S.T., Ph.D.) |

# DAFTAR ISI

HALAMAN PENGESAHAN i

[DAFTAR ISI ii](#_Toc492461380)

[DAFTAR GAMBAR DAN ILUSTRASI iii](#_Toc492461381)

[DAFTAR TABEL iv](#_Toc492461382)

[Bab I Pendahuluan 1](#_Toc492461383)

[I.1 Latar Belakang 1](#_Toc492461384)

[I.2 Rumusan Masalah 3](#_Toc492461385)

[I.3 Tujuan 3](#_Toc492461386)

[I.4 Batasan Masalah 3](#_Toc492461387)

[I.5 Metodologi Penelitian 4](#_Toc492461388)

[I.6 Sistematika Penulisan 4](#_Toc492461389)

[I.7 Rencana Waktu Penyelesain Tesis 5](#_Toc492461390)

[Bab II Tinjauan Pustaka 6](#_Toc492461391)

[II.1 Struktur Data Graf 8](#_Toc492461392)

[II.2 Revolusi Teknologi Web dan *Computational Thinking* 8](#_Toc492461393)

[II.3 Terminologi Visualisasi Perangkat Lunak 6](#_Toc492461394)

[II.4 Peran Penting Eksekusi Kode dalam Pembelajaran Pemrograman **Error! Bookmark not defined.**](#_Toc492461395)

[II.5 Hasil Eksplorasi Kakas Visualisasi Program 12](#_Toc492461396)

[II.6 Eksplorasi Kakas *Online Python Tutor* 13](#_Toc492461397)

[II.6.1 Arsitektur Kakas OPT 15](#_Toc492461398)

[II.6.2 Komponen Kakas OPT 16](#_Toc492461399)

[II.6.3 *Capturing Execution Trace* 16](#_Toc492461400)

[II.6.4 *Execution Trace Format* 17](#_Toc492461401)

[II.6.5 Fitur *Data-Driven Documents* (D3) *Framework* 19](#_Toc492461402)

[II.7 *Matrix Framework* 19](#_Toc492461403)

[II.7.1 Visualisasi 19](#_Toc492461404)

[II.7.2 Struktur 21](#_Toc492461405)

[II.7.3 Konstruksi Visual 22](#_Toc492461406)

[II.7.4 Animasi 23](#_Toc492461407)

[II.7.5 Simulasi 23](#_Toc492461408)

[II.7.6 Tampilan Antarmuka Pengguna 24](#_Toc492461409)

[II.8 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi 24](#_Toc492461410)

[Bab III Analisis Masalah Awal 25](#_Toc492461411)

[III.1 Analisis dan Peluang Pemecahan Masalah 25](#_Toc492461412)

[III.2 Penunjang Pengembangan Kakas 26](#_Toc492461413)

[III.2.1 Kebutuhan Perangkat Lunak 26](#_Toc492461414)

[III.2.2 Kebutuhan Perangkat Keras 27](#_Toc492461415)

[DAFTAR PUSTAKA 28](#_Toc492461416)

# DAFTAR GAMBAR DAN ILUSTRASI

[Gambar II.1 (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969) **Error! Bookmark not defined.**](#_Toc492467560)

[Gambar II.2 Model visual graf secara umum: (a) Pohon dan (b) Graf 11](#_Toc492467561)

[Gambar II.3 Tampilan Antarmuka OPT untuk Bahasa Pemrograman C 13](#_Toc492467562)

[Gambar II.4 Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman 14](#_Toc492467563)

[Gambar II.5 Tampilan Simulasi dan Visualisasi Kode C Pointer 14](#_Toc492467564)

[Gambar II.6 Arsitektur OPT untuk visualisasi C dan C++ 15](#_Toc492467565)

[Gambar II.7 Ilustrasi Sederhana Proses Kakas OPT 16](#_Toc492467566)

[Gambar II.8 Ilustrasi Format *Trace* Eksekusi Program 18](#_Toc492467567)

[Gambar II.9 Peran Utama D3 sebagai *Framework* Visualisasi 19](#_Toc492467568)

[Gambar II.10 Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004) 20](#_Toc492467569)

[Gambar II.11 Turunan hirarki *concept interfaces* (Korhonen dkk., 2004) 21](#_Toc492467570)

[Gambar II.12 Proses konstruksi representasi visual (Korhonen dkk., 2004) 22](#_Toc492467571)

# DAFTAR TABEL

[Tabel I.1 Jadwal rencana kegiatan penelitian tesis 5](#_Toc492461313)

[Tabel II.1 Daftar nama algoritma graf dan model visualnya (Valiente, 2002) 8](#_Toc492461314)

[Tabel II.2 Daftar nama algoritma pohon dan model visualnya (Valiente, 2002) 10](#_Toc492461315)

[Tabel II.3 Perkembangan kakas VP untuk bahasa pemrograman C/C++ 12](#_Toc492461316)

[Tabel III.1 Hasil Analisis dan Peluang Pemecahan Masalah 25](#_Toc492461317)

[Tabel III.2 Kakas dan *library* pendukung pengembangan OPT 26](#_Toc492461318)

# Bab I Pendahuluan

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian, batasan implementasi, metodologi penelitian, sistematika penulisan laporan tesis dan rencana waktu penyelesaian tesis.

## I.1 Latar Belakang

Berbagai riset dalam dekade tahun terakhir para peneliti mencari cara untuk mengembangkan kakas guna memahami alur kerja sebuah program. Visualisasi menjadi salah satu teknik yang telah dilakukan, sehingga saat ini muncul istilah *software visualization* (visualisasi perangkat lunak). Visualisasi adalah teknik interaksi terpenting dalam penyampaian informasi (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013; Gračanin dkk., 2005; Guo, 2013). Melalui indera visual, manusia lebih cenderung menangkap lebih banyak informasi yang diterima dibandingkan melalui indera lainnya (Ware, 2004).

Istilah visualisasi dalam perangkat lunak (Cetin dan Andrews-Larson, 2016) didefinisikan sebagai sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-komputer. Visualisasi perangkat lunak (*software visualization*/ SV) dibagi menjadi dua bagian besar (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013), yaitu visualisasi algoritma (*algorithm visualization*/ VA) dan visualisasi program (*program visualization*/ VP). VA berkaitan dengan abstraksi algoritma, sedangkan VP berkaitan dengan eksekusi kode program dan proses struktur data.

Saat ini teknologi internet dan web memberikan kemudahan akses untuk berbagi informasi (Bonk, 2009). Aplikasi web terus berkembang dari sekadar penyampaian informasi kontekstual statis. Kemajuan teknologi internet dan web tersebut berdampak terhadap kemutakhiran penyajian visualisasi informasi secara dinamis. Teknologi seperti Java3D, VRML, X3D, dan SVG memiliki kemampuan *rendering* yang *powerful*, tetapi sulit untuk berinteraksi dengan sumber data mentah (Holmberg dkk., 2006). Selain itu, web *browser* standar masih memiliki beberapa isu terhadap aplikasi visualisasi interaktif. Isu pertama, visualisasi hanya dapat direpresentasikan dengan teknik *computer graphics* modern yang hal tersebut tidak mungkin dapat diterapkan dalam lingkungan web (contohnya animasi 2D/3D). Isu kedua adalah interaksi. Web *browser* standar hanya menyediakan *form* kontrol dan interaksi yang terbatas. Beberapa interaksi telah dapat dilakukan pada *client-side* dengan integrasi teknologi seperti *JavaScript*, *Java Applets*, dan berbagai pendukung *plug-in*, tetapi jika hal tersebut menjadi lebih efektif untuk interaksi visualisasi. Isu ketiga adalah teknologi jaringan internet. Ini berkaitan dengan performa komunikasi data antar *server-client*, baik itu *throughput* maupun latensi, yang akan berdampak terhadap keefektifan kinerja aplikasi. Isu terakhir adalah aksesibilitas terhadap teknologi, yaitu *browser* standar pengguna dapat menggunakan aplikasi secara mandiri tanpa bergantung terhadap *plug-in* lain.

Perkembangan kakas VP berbasis web masih sangat langka (Sorva, 2012; Sorva dkk., 2013). Umumnya pengembangan kakas dibangun dengan *Java Virtual Machine* (JVM) (Helminen dan Malmi, 2010) yang pada dasarnya kurang efektif, karena akan membutuhkan waktu dan usaha lebih untuk mengkonfigurasi pada *browser*. Salah satu kakas VP berbasis web yang masih aktif adalah *Online Python Tutor* (OPT), yang dapat langsung diakses melalui *browser* dengan alamat *www.pythontutor.com*. OPT memanfaatkan pustaka D3JS untuk melakukan visualisasi. D3JS adalah sebuah pustaka *JavaScript* bersumber terbuka sebagai “*kernel*” visualisasi yang diciptakan untuk mengurangi ketergantungan pada *browser* (Bostock dkk., 2011). Namun saat tesis ini ditulis, OPT hanya dapat memvisualkan data primitif berbentuk tabel dan *stack*. Untuk data yang lebih kompleks seperti visual graf, OPT belum dapat merepresentasikannya sesuai persepsi visual algoritma graf. Permasalahannya adalah setiap kode program memiliki algoritma tersendiri, sehingga membutuhkan *trigger* sebagai penentu terhadap visual mana yang sesuai untuk dilakukan model abstraksi. Pada penelitian ini, pengembangan kakas berdasar kode sumber dari OPT dengan memanfaatkan pustaka D3JS. Kakas OPT ini bersifat bebas (*free*) dan bersumber terbuka (*open source*) (Guo, 2013), sehingga tidak ada batasan dalam pengembangannya.

## I.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan tersebut, maka diperoleh rumusan masalah sebagai berikut.

1. Apa strategi pengembangan kakas untuk mendeteksi algoritma graf dalam sebuah kode program sehingga dapat direpresentasikan secara abstrak ?
2. Apa yang harus dilakukan dalam visualisasi untuk meningkatkan persepsi manusia dalam representasi visual graf dari sebuah kode program ?

## I.3 Tujuan Penelitian

Sebagai solusi dari masalah yang telah diuraikan, maka tujuan dari penelitian tesis ini adalah untuk menghasilkan modifikasi kakas OPT yang dapat melakukan visualisasi eksekusi program dengan mendeteksi adanya algoritma graf dalam sebuah kode program.

## I.4 Batasan Implementasi

Untuk memfokuskan kegiatan penelitian tesis ini diperlukan batasan, sehingga cakupan pembahasan dan penyelesaian tidak meluas. Adapun batasan implementasi pada tesis ini adalah :

1. Implementasi kakas VP berdasar kode sumber dari OPT dengan memanfaatkan pustaka D3JS;
2. Bahasa pemrograman yang dapat divisualisasikan adalah C dan C++;
3. Algoritma yang dapat dideteksi dalam kode program adalah BFS dan DFS;
4. Kode program yang dapat divisualisasi minimal memiliki data fisik berupa *array* atau *pointer*;
5. Kakas tidak menerima *standard input* (*stdin*) dan *include multiple class object files*.

## I.5 Metodologi Penelitian

Uraian metode penelitian dan rencana pengembangan serta pengujian dijelaskan pada bagian ini. Penelitian tesis ini akan dilaksanakan melalui beberapa tahapan sebagai berikut.

1. Tinjauan Pustaka dan Eksplorasi

Tinjauan pustaka menghasilkan kesimpulan awal yang dapat dijadikan landasan dalam perumusan masalah dan pengembangan kakas. Kemudian dilakukan eksplorasi terhadap kakas VP yang telah dikembangkan.

1. Analisis Masalah dan Perumusan Skema Graf

Tahap ini dilakukan kajian terhadap masalah yang akan diteliti kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Selanjutnya, dirumuskan suatu skema untuk mengonversi kode menjadi model animasi dalam sebuah tampilan grafis diagram pohon dan graf.

1. Pengembangan Kakas

Pengembangan kakas dilakukan dengan menggunakan metode *Rational Unified Process* (TP026B, n.d.).

1. Pengujian dan Evaluasi Kakas

Hasil pengembangan kakas perlu diuji dengan memasukkan berbagai jenis kode program dan tipe data, kemudian menganalisis hasil dari visualisasi tersebut. Selain itu, kakas perlu dievaluasi secara empiris dengan bantuan responden untuk menilai efektivitas kakas dalam representasi visual graf.

## I.6 Sistematika Penulisan

Penulisan laporan hasil penelitian tesis ini akan dibagi menjadi enam bab, yaitu:

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, metodologi penelitian, dan sistematika penulisan laporan.

1. Bab II Tinjauan Pustaka dan Eksplorasi

Bab ini berisi uraian tentang terminologi visualisasi perangkat lunak, perkembangan kakas visualisasi dari hasil penelitian sebelumnya yang berkaitan dengan masalah yang dikaji, sehingga memberikan gambaran perkembangan terhadap masalah yang akan diteliti.

1. Bab III Analisis Masalah dan Perumusan Skema Graf

Bab ini berisi analisis terhadap masalah yang akan diteliti, kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan, sehingga untuk mendukung hipotesis tersebut, dibangun pula rancangan dari sistem yang akan dikembangkan.

1. Bab IV Pengembangan Kakas

Bab ini menjelaskan proses pengembangan dan implementasi kakas VP berdasarkan metode *Rational Unified Process* (TP026B, n.d.).

1. Bab V Pengujian dan Evaluasi Kakas

Bab ini menguraikan proses dan hasil pengujian serta evaluasi dari kakas yang dikembangkan. Pengujian dilakukan untuk membuktikan bahwa kakas berhasil dikembangkan lebih baik dibandingkan sebelumnya. Evaluasi kakas dilakukan untuk menilai efektivitas dan kualitas representasi visualisasi.

1. Bab VI Penutup

Bab ini berisi kesimpulan dan saran untuk pengembangan kakas lebih lanjut.

## I.7 Rencana Waktu Penyelesain Tesis

Penelitian tesis ini akan dilaksanakan sekurang-kurangnya dalam waktu enam bulan dan paling lama dua belas bulan, dengan rincian kegiatan yang tercantum pada Tabel I.1 berikut ini.

Tabel I.1 Jadwal rencana kegiatan penelitian tesis

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **Bulan ke-** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Studi literatur dan eksplorasi |  |  |  |  |  |  |
| 2 | Analisis masalah dan perumusan skema graf |  |  |  |  |  |  |
| 3 | Pengembangan kakas |  |  |  |  |  |  |
| 4 | Pengujian kakas |  |  |  |  |  |  |
| 5 | Evaluasi kakas dan penulisan laporan |  |  |  |  |  |  |

# Bab II Tinjauan Pustaka dan Eksplorasi

Pada bab ini dijelaskan mengenai terminologi visualisasi, perkembangan teknologi web dan pustaka untuk visualisasi, graf dan model visualnya, perkembangan kakas VP, hasil eksplorasi kakas OPT, dan kesimpulan awal berdasarkan studi literatur dan eksplorasi.

## II.1 Terminologi Visualisasi Perangkat Lunak

Visualisasi perangkat lunak (*Software Visualization*) merupakan bidang aktif dalam riset dan pengembangan sistem. Berbagai sistem visualisasi perangkat lunak bermunculan untuk digunakan dengan tujuan tertentu dan terus berkembang setiap tahunnya (Sorva, 2012). Gračanin dkk., (2005) mendefinisikan visualisasi perangkat lunak (*Software Visualization*) sebagai suatu bidang untuk menginvestigasi dengan pendekatan dan teknik tertentu yang bertujuan dalam merepresentasikan grafis algoritma secara statis dan dinamis, program (*code*), dan data yang diproses. Visualisasi perangkat lunak memiliki tujuan utama untuk menganalisis program dan pengembangan; untuk meningkatkan pemahaman terhadap konsep yang tak nampak dan cara kerja perangkat lunak. Tantangan utamanya adalah mencari langkah efektif dalam pemetaan berbagai aspek perangkat lunak untuk direpresentasikan secara grafis menggunakan metafora visual.

Istilah “visualisasi perangkat lunak” telah lama berkembang dan didefinisikan sebagai sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-komputer (Cetin dan Andrews-Larson, 2016). Visualisasi perangkat lunak dibagi menjadi dua, yaitu visualisasi algoritma (VA) dan visualisasi program (VP) (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013). VA berkaitan dengan abstraksi dari konsep langkah kerja perangkat lunak, sedangkan VP terkait pada cara kerja eksekusi kode program dan proses struktur datanya. Melalui kakas VP dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari materi ajar (Sorva, 2012).

Visualisasi secara intuitif tampaknya telah menjadi bagian dari sarana pembelajaran yang menarik (Sorva, 2012). Berbagai literatur mendukung pernyataan tersebut (Cetin dan Andrews-Larson, 2016; Guo, 2013; Pathania dan Singh, 2014; Sorva dkk., 2013; Sorva dan Sirkiä, 2010). Penelitian terkait banyak yang mengusulkan visualisasi digunakan dalam lingkungan belajar. Hampir setiap teori pembelajaran merekomendasikan visualisasi sebagai sarana untuk memberikan pemahaman lebih baik kepada pembacanya (Clark dan Mayer, 2011; Mayer, 2014).

Penggunaan sarana visual sebagai alat bantu belajar telah menjadi budaya di seluruh dunia, baik itu dalam pembelajaran komputer atau lainnya. Berdasarkan salah satu survei bahwa penggunaan visualisasi dalam perkuliahan informatika atau ilmu komputer hampir dilakukan setiap hari (Sorva, 2012).

Banyak kakas VP dibuat karena termotivasi dengan munculnya kesulitan belajar untuk memahami konsep algoritma dan pemrograman. Beberapa kesulitan yang ada diantaranya sebagai berikut (Sorva dkk., 2013).

1. **Persepsi statis terhadap pemrograman**. Umumnya para pelajar memahami konsep program tertentu (seperti objek, rekursi) hanya merupakan bagian dari kode program, bukan memahaminya sebagai suatu komponen aktif yang utuh saat eksekusi kode program.
2. **Kesulitan memahami cara kerja mesin komputer**. Hambatan utama bagi pemula adalah memahami notasi mesin (Boulay, 1986), yaitu sebuah abstraksi dari komputer dalam eksekusi kode program dengan bahasa pemrograman tertentu. Bahasa pemrograman dan paradigma yang berbeda dapat mengasosiasikan dengan notasi mesin yang berbeda pula.
3. **Salah paham terhadap konsep dasar pemrograman**. Berbagai literatur pendidikan komputasi terdapat banyak laporan tentang kesalahpahaman pelajar yang dilakukan saat mempelajari konsep pemrograman (Clancy dkk., 2003; Sorva, 2012). Permasalahannya menyangkut eksekusi kode program yang tersembunyi dalam notasi mesin dan konsep algoritma yang tidak jelas dalam kode program, seperti referensi variabel dan *pointer*, objek, rekursi, variabel kontrol perulangan, dan sebagainya.
4. **Kesulitan penelusuran dan langkah program**. Kesulitan yang dialami pelajar dengan menelusuri program langkah demi langkah telah banyak dilaporkan di berbagai studi multinasional dan literatur (Sorva dkk., 2013). Laporan tersebut menyatakan bahwa pelajar gagal memahami pernyataan program secara sekuensial dan belum mampu memahami proses pertukaran nilai variabel dalam tiga baris sederhana.

## II.2 Revolusi Teknologi Web dan Pustaka untuk Visualisasi

## II.3 Graf dan Model Visualnya

Graf memiliki berbagai macam algoritma dan model visualnya, sehingga perlu dijabarkan lebih rinci agar mengetahui konsep dasar terhadap proses kerja dari kakas VP yang akan dibangun. Daftar nama algoritma dan model visual graf disajikan pada Tabel II.1 berikut.

1. Daftar nama algoritma graf dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Graf** | **Model Visual** |
| --- | --- | --- |
| 1 | *Breadth First Search* (BFS) |  |
| 2 | *Depth First Search* (DFS) |  |
| 3 | Prim |  |
| 4 | Dijkstra |  |
| 5 | Bellman-Ford | http://2.bp.blogspot.com/-GG_lKtfmonw/UYi73BodmvI/AAAAAAAAAjw/UGlBfBpEKI4/s400/nodenode.png |
| 6 | Floyd-Warshall |  |
| 7 | Kruskal |  |

Pohon (Rinaldi Munir, 2010) merupakan bagian dari graf terhubung yang tidak mengandung sirkuit. Teori pohon (*tree*) merupakan konsep paling penting, karena memiliki banyak terapan di berbagai bidang ilmu komputer maupun bidang lainnya. Konsep ini telah banyak dikaji secara intensif sebagai objek matematika dan terapan dalam kehidupan manusia sehari-hari. Banyak orang menggunakan pohon untuk membuat hirarki seperti silsilah keluarga, struktur organisasi, organisasi pertandingan olahraga, dan lain-lain. Pohon parsing (*parse tree*) digunakan oleh para ahli bahasa untuk menguraikan kalimat.

Sejak tahun 1857, istilah pohon telah lama digunakan oleh matematikawan asal Inggris bernama Arthur Cayley. Ia memanfaatkan pohon untuk menghitung jumlah senyawa kimia. Pohon sebagai struktur data rekursif yang merupakan bagian penting dari perkuliahan Struktur Data dan Algoritma Pemrograman. Daftar nama-nama algoritma dan model visual pohon yang ditampilkan pada Tabel II.2 berikut ini merupakan yang paling sering digunakan dalam perkuliahan Struktur Data dan Algoritma Pemrograman.

1. Daftar nama algoritma pohon dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Pohon** | **Model Visual** |
| --- | --- | --- |
| 1 | *Binary Search Tree* (BST) |  |
| 2 | *Digital Search Tree* |  |
| 3 | *Radix Tree* | https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/An_example_of_how_to_find_a_string_in_a_Patricia_trie.png/220px-An_example_of_how_to_find_a_string_in_a_Patricia_trie.png |
| 4 | AVL (Adelson-Velskii dan Landis) *Tree*; terdiri dari *Single Rotation* dan *Double Rotation* |  |
| 5 | *Red-Black Tree* | Diagram of binary tree. The black root node has two red children and four black grandchildren. The child nodes of the grandchildren are black nil pointers or red nodes with black nil pointers. |
| 6 | *Search Tree Traversal*; terdiri dari *pre-order*, *in-order*, dan *post-order* | Hasil gambar untuk tree traversal algorithm |
| 7 | *Splay Tree* |  |

Dari daftar nama berbagai macam algoritma dan model-model visual graf tersebut, dapat ditarik kesimpulan secara umum representasi visualisasi yang dapat dilakukan hanya dua bagian besar yaitu model visual pohon dapat dilihat pada Gambar II.1 (a) dan model visual graf pada Gambar II.2 (b).

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

1. Model visual graf secara umum: (a) Pohon dan (b) Graf

## II.4 Hasil Eksplorasi Kakas Visualisasi Program

Tinjauan pustaka terhadap penelitian mengenai pengembangan kakas VP dilakukan untuk memperoleh gambaran mengenai konsep VP dalam representasi visual serta mempelajari arsitektur kakas yang telah dikembangkan. Kemudian dilakukan eksplorasi yang bertujuan untuk menghindari duplikasi riset yang dilakukan oleh peneliti lain, sehingga dapat diperoleh sebuah ide baru yang belum pernah dikembangkan sebelumnya.

Pada Tabel II.3 menunjukkan perkembangan kakas VP khusus untuk bahasa pemrograman C dan C++ yang masih aktif pada terakhir dekade tahun ini. Sorva (2012) menjelaskan secara rinci perkembangan 40 kakas VP pada Bab 11 disertasinya. Dari semua perkembangan kakas yang masih aktif itu, dipilih hanya kakas VP yang mendukung bahasa pemrograman C dan C++ untuk ditampilkan pada Tabel II.3.

1. Perkembangan kakas VP untuk bahasa pemrograman C/C++

| **No.** | **Tahun** | **Nama** | **Dukungan Bahasa** | **Visual Graf** | **Berbasis Web** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1996 | GRASP / jGRASP | Ada, Java, C, C++, Objective-C, VHDL | Binary Tree | - | Aktif |
| 2 | 2000 | The Teaching Machine | C++, Java | **-** | **√** | Aktif |
| 3 | 2002 | PlanAni | Pascal, Java, C, Python | - | - | Aktif ? |
| 4 | 2003 | Jeliot 2000 / Jeliot 3 | Java, C, Python | - | - | Aktif |
| 5 | 2004 | OGRE | C++ | - | - | Non-aktif? |
| 6 | 2005 | VIP | C++ | - | - | Aktif |
| 7 | 2005 | ViLLE | Java, C++, Python, PHP, JavaScript, pseudocode | - | - | Aktif |
| 8 | 2010 | Online Python Tutor | Python, Java, C, C++, Ruby, JavaScript | **-** | **√** | Aktif |
| 9 | 2011 | CSmart | C | - | - | Aktif |

Hasil eksplorasi perkembangan kakas VP sejak tahun 1996 hingga terakhir tahun 2011 terlihat pada Tabel II.3. Terdapat tujuh kakas yang masih aktif dan dua kakas yang statusnya tidak jelas. Hanya dua kakas yang mendukung teknologi web. Kakas *The Teaching Machine* (*www.theteachingmachine.org*) memang masih aktif dalam pengembangan. Kekurangannya, kakas ini membutuhkan verifikasi *Plugin Java* yang terpasang di *browser* pengguna. Hal ini sangat berbeda dengan kakas OPT yang dapat langsung dikunjungi (*pythontutor.com*) dan digunakan secara langsung dari *browser* pengguna.

## II.5 Eksplorasi Kakas *Online Python Tutor*

*Online Python Tutor* atau disingkat OPT (Guo, 2013) merupakan kakas VP bersifat bebas dan bersumber terbuka. Kakas ini telah memiliki banyak fitur VP yang mendukung berbagai macam bahasa pemrograman, seperti Python, Java, C, C++, Ruby, JavaScript, TypeScript dan masih terus dikembangkan[[1]](#footnote-1). Ketika penelitian tesis ini dibuat, OPT telah berkembang pada versi 5 yang dirilis pada tanggal 27 Juli 2016. Penelitian aktif dilakukan pada versi 5 dengan menggunakan bahasa pemrograman *TypeScript* sebagai dasar pembangunan kakas OPT.



1. Tampilan Antarmuka OPT untuk Bahasa Pemrograman C

Pada Gambar II.2 ditunjukkan tampilan antarmuka kakas OPT untuk bahasa pemrograman C. Fungsi nomor 1 adalah untuk berbagi antarmuka bagi pengguna lain yang ingin belajar pemrograman secara berkolaborasi. Ketika tombol tersebut diklik, tampilan tampak seperti pada Gambar II.3. Fungsi ini menggunakan kode pustaka dari *TogetherJS[[2]](#footnote-2)* yang dikembangkan oleh *Mozilla Labs* secara bebas dan terbuka. Pengguna lain dapat ikut bergabung dengan mengunjungi pranala yang dibagikan oleh pengguna pertama yang mengaktifkan fitur ini.



1. Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman

Fungsi nomor 2 pada Gambar II.2 adalah untuk memasukkan kode program. Pelajar dapat mulai belajar bahasa pemrograman dengan mengetikkan kode di *form editor* yang tersedia. Selanjutnya pelajar dapat menekan tombol pada nomor 3 untuk melihat simulasi dan visualisasi dari kode program yang telah dibuatnya, seperti pada Gambar II.4.



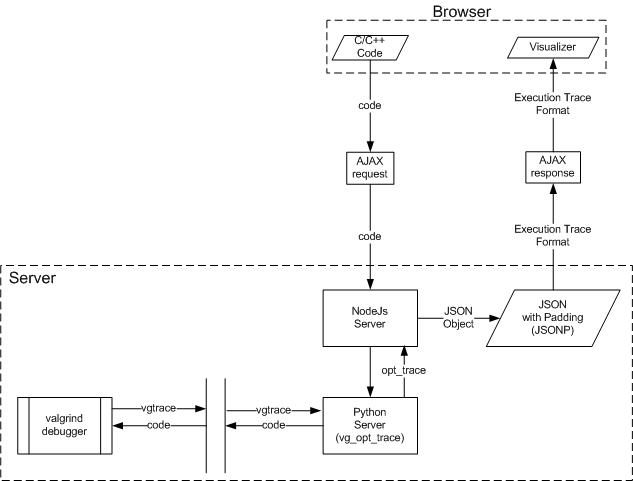
1. Tampilan Simulasi dan Visualisasi Kode C Pointer

Pada beberapa subbab berikut dijelaskan susunan arsitektur dasar pembangunan kakas OPT yang mendukung proses VP khusus dalam bahasa pemrograman C dan C++.

### II.5.1 Arsitektur Kakas OPT

Penjelasan arsitektur ini berguna untuk memberikan gambaran umum tentang cara kerja atau alur sistem didalamnya, sehingga dapat menentukan metode yang optimal. Selain itu, dapat dijadikan dasar dalam mengembangkan sistem dan menunjang modularisasi pada jangka waktu yang panjang.

Pada Gambar II.5 ditampilkan diagram alur komunikasi antara *browser* milik pengguna dengan *server*. Pertukaran informasi kode program dilakukan dengan fitur permintaan *AJAX* (*Asynchronous JavaScript And XML*).



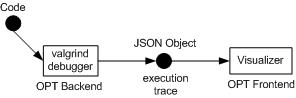
1. Arsitektur OPT untuk visualisasi C dan C++

Untuk lebih jelas tahapan proses pada Gambar II.5 di atas, berikut langkah-langkah yang terjadi ketika pengguna melakukan “Eksekusi Visualisasi” pada kode programnya.

1. Pada antarmuka *browser*, sebenarnya kode tersebut dikirim sebagai tipe data *string* ke *server* dengan metode *AJAX request*.
2. Kemudian *server* mengeksekusi kode tersebut dengan *NodeJS* yang dikombinasikan dengan *server* *Python* untuk menghasilkan *trace* v*algrind*.
3. *Server Python* mengubah format *trace* eksekusi sebagai *JSON* (*JavaScript Object Notation*) sebelum dikembalikan ke *server* *NodeJS*.
4. *JSONP* digunakan untuk memperoleh data dengan permintaan *AJAX* yang berbeda domain atau alamat *IP* (*Internet Protocol*).
5. Respon *AJAX* diterima oleh *visualizer* kemudian dibaca sesuai format yang ditentukan. Maka diperoleh visualisasi kode program yang tampil di *browser* pelajar.
6. Ketika pelajar menekan tombol “*Forward*” atau “*Back*” (lihat Gambar II.4) proses visualisasi akan membaca poin indeks yang berada pada *trace* eksekusi sesuai dengan banyak langkah yang telah terbentuk.

### II.5.2 Komponen Kakas OPT

Kakas OPT dapat disederhanakan menjadi beberapa inti komponen dengan representasi abstrak seperti tampak pada Gambar II.6 berikut ini.



1. Ilustrasi Sederhana Proses Kakas OPT

*OPT backend* berfungsi untuk respon terhadap terbentuknya *trace* eksekusi program. Kemudian *trace* eksekusi diubah menjadi standar format tertentu yang dipaketkan berbentuk objek *JSON*. *OPT frontend* (antarmuka *browser*) menerjemahkan objek *JSON* menjadi visualisasi yang dikombinasikan dengan pustaka *D3 framework*. Maka visualisasi dapat tampil kepada pelajar seperti pada Gambar II.4.

### II.5.3 *Capturing Execution Trace*

Ketika kode program dikirim ke *OPT backend*, kode C/C++ dieksekusi dengan *valgrind framework*. Kemudian dari *valgrind* terbentuk *trace* yang sudah dimodifikasi oleh pengembangnya. Versi *valgrind* yang digunakan adalah 3.11.0 dengan modifikasi (versi asli bisa unduh di *www.valgrind.org*) beberapa baris kode agar dapat menghasilkan *trace* eksekusi program. Beberapa berkas yang dimodifikasi adalah sebagai berikut:

|  |  |
| --- | --- |
| 1 | valgrind-3.11.0\memcheck\**mc\_main.c**  (Baris ke 54, 55, 3789, 5693, 5715-5718, 7394-7406, 7646-7647, 7761-7763) |
| 2 | valgrind-3.11.0\include\**pub\_tool\_debuginfo.h**  (Baris ke 36-37, 124-132, 191, 198, 214, 285-408) |
| 3 | valgrind-3.11.0\coregrind\m\_debuginfo\**debuginfo.c**  (Baris ke 74-1458, 5043-5159, 5472, 5594-5719, 5846) |
| 4 | valgrind-3.11.0\memcheck\**mc\_translate.c**  (Baris ke 6261-6481, 6665, 6708-6716, 6795-6811) |
| 5 | valgrind-3.11.0\coregrind\m\_debuginfo\**tytypes.c**  (Baris ke 47-51, 328-864) |
| 6 | valgrind-3.11.0\coregrind\m\_debuginfo\**priv\_tytypes.h**  (Baris ke 160-165) |
| 7 | valgrind-3.11.0\**config.h** (Baris ke 41) |
| 8 | valgrind-3.11.0\memcheck\**mc\_include.h** (Baris 142-156) |

Berkas-berkas valgrind tersebut dianalisis perbedaannya dengan menggunakan kakas *WinMerge* (*winmerge.org*). Tekniknya dengan membandingkan valgrind versi asli yang diunduh dari *www.valgrind.org*.

Untuk dapat menghasilkan *trace* eksekusi program, kode program dikirim ke *server* yang diolah oleh *ExpressJS*. Kemudian dengan program *Python* dihubungkan dengan *Pipe* ke *terminal* untuk di-*compile* dalam *valgrind debugger*. Dari sini menghasilkan *trace* khusus dari *valgrind* disebut *vgtrace*. Selanjutnya dikonversi ke bentuk *JSON* atau *JSONP* (*JSON with Padding*) oleh *ExpressJS* dan dikembalikan sebagai respon permintaan *AJAX*. Ilustrasi lengkap dapat dilihat pada Gambar II.5.

### II.5.4 *Execution Trace Format*

*OPT back-end* menyimpan informasi *trace* eksekusi dalam standar format tertentu yang kemudian dapat digunakan untuk berinteraksi dengan *visualizer* di *OPT front-end.* Sebagai ilustrasi format *trace* eksekusi program yang dimaksud dapat dilihat pada Gambar II.7 berikut ini.



1. Ilustrasi Format *Trace* Eksekusi Program

Bagan pada Gambar II.7 di atas mengilustrasikan posisi eksekusi kode pada saat masih kosong (*empty state*). *Trace* eksekusi program disimpan dalam bentuk objek JSON yang didefinisikan dalam dua bagian besar, yaitu:

1. “***code***” berbentuk string, yang berisi kode program dari pelajar.
2. “***trace***” berbentuk objek-objek larik, yang setiap objek merepresentasikan posisi eksekusi kode sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 1 | ordered\_globals: [], | // sebuah larik yang merepresen-tasikan urutan visual untuk atribut global |
| 2 | stdout: “”, | // total standar output dari kode program pada posisi eksekusi kode saat ini |
| 3 | func\_name: “<module>”, | // fungsi saat ini yang sedang diekseskusi |
| 4 | stack\_to\_render: [], | // merupakan *list* dari objek-objek, setiap objek direpresentasikan sebagai sebuah *stack frame* |
| 5 | globals: {}, | // sebuah *dictionary global* ‘*stack frame*’ |
| 6 | heap: {}, | // sebuah *dictionary* objek-objek heap |
| 7 | line: 1, | // indikasi baris kode yang sedang dieksekusi |
| 8 | event: “step\_line” | // sebuah parameter *event* yang juga dapat berisi: *user\_call*, *user\_return*, *user\_exception* atau *user\_line* |

### II.5.5 Fitur Pustaka *Data-Driven Documents* (D3)

Teknologi visualisasi utama yang digunakan oleh OPT adalah *Data-Driven Documents* (D3JS) (Bostock dkk., 2011) yang merupakan salah satu pustaka terpopuler untuk implementasi visualisasi berbasis web. Format *trace* eksekusi program dibaca oleh D3JS untuk mendeklarasikan pemetaan data dan atribut yang diperlukan oleh elemen *Document Object Model* (DOM) pada laman *Hyper Text Markup Language* (HTML). Peran D3JS dapat dilihat pada Gambar II.8 berikut ini.



1. Peran Utama D3 sebagai *Framework* Visualisasi

Pustaka D3JS sangat fleksibel terhadap abstraksi data tingkat tinggi. Pustaka ini juga mampu memvisualisasikan berbagai macam data, bahkan cocok untuk visualisasi dengan tingkat kerumitan yang tinggi.

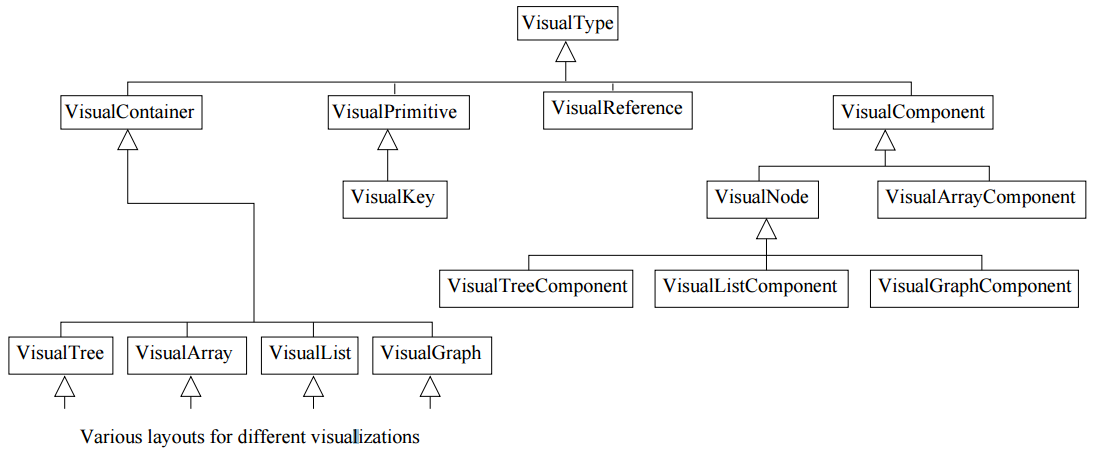
## II.6 *Matrix Framework*

Pada subbab ini akan dibahas *Matrix framework* yang dikembangkan oleh Korhonen dkk. (2004). *Framework* ini dibangun untuk mendukung desain dan implementasi visualisasi pada tingkat abstrak. Awal pengembangannya diimplementasi menggunakan bahasa pemrograman Java. Kemudian dipadukan dengan *XML library* yang berguna untuk membaca konfigurasi berkas *Matrix*. Berikut ini akan dijelaskan enam bagian yang ada di dalam *Matrix framework*, yaitu visualisasi, struktur, konstruksi visual, animasi, simulasi, dan tampilan antarmuka pengguna.

### II.7.1 Visualisasi

Visualisasi struktur data dalam *Matrix framework* berisi empat konsep dasar visual, yaitu *visual container*, *visual component*, *visual reference* dan *visual data*. *Visual container* berupa struktur yang kompleks, menyimpan nilai variabel (node, indeks, dan lainnya) yang saling terhubung dengan cara tertentu. Kemudian tiap variabel di dalam *visual container* terdapat *visual component* yang berguna untuk memvisualisasikan variabel. *Visual reference* sebagai penghubung antar *visual component* dan *visual data* yang berisi tipe data primitif atau struktur data yang lebih kompleks dengan atribut-atributnya. Atribut kunci yang kompleks direpresentasikan menggunakan *visual container* yang bersarang di dalam *visual component*.

Pada Gambar II.9 menunjukkan hirarki *class Matrix framework* yang digunakan untuk implementasi visualisasi. *Class* pada tingkat kedua dari gambar hirarki tersebut merupakan empat konsep visual yang telah dijelaskan sebelumnya. VisualContainer berhubungan dengan *visual container*, VisualComponent dengan *visual component*, VisualReference dengan *visual reference*, dan VisualPrimitive dengan *visual data*.

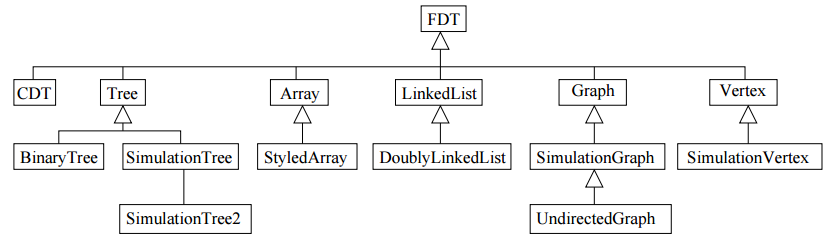


1. Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004)

Visualisasi dengan tipe data berbeda membutuhkan setiap fungsi yang berbeda pula, seperti tipe data *arrays*, graf, *list*, atau pohon. Pada kelas VisualContainer terdiri dari empat sub kelas (lihat Gambar II.9), yaitu VisualTree, VisualArray, VisualList, dan VisualGraph. Setiap sub kelas berkorespondensi dengan salah satu tipe data yang berada di atasnya. Setiap kelas menangani fungsinya masing-masing terhadap visualisasi dan manipulasi tipe data tertentu. Sebagai contoh, kelas VisualGraph berperan ketika pengguna menambahkan sisi (*edge*), sedangkan kelas VisualTree memastikan bahwa graf yang ditampilkan adalah pohon, yang berarti tidak terdapat sisi atau garis yang melintang (*cross*), atau sisi yang mengarah kembali (*back-edges*) ke noktah sebelumnya. Selain itu, pada masing-masing kelas utama terdapat satu atau lebih sub kelas yang menangani tipe data tertentu dalam menguraikan visualisasi dan menggambarkan *layout* tertentu.

### II.7.2 Struktur

*Matrix framework* menggunakan sejumlah *concept interfaces* untuk membedakan antara *fundamental data types* (FDT) dan *abstract data types* (ADT). *Concept interfaces* diimplementasi menggunakan *interface* *Java*. Hirarki dari struktur *interfaces* yang digunakan dapat dilihat pada Gambar II.10. *Interfaces FDT* merupakan *superclass* (*root*) dari semua konsep *interfaces*. Sebuah objek yang dikenali oleh *Matrix framework* akan dapat divisualisasikan dari implementasi *interfaces FDT* ini. Untuk objek yang tidak dapat dikenali (bukan objek elemen *FDT*), maka akan disimpan dan divisualisasikan sebagai objek data primitif (hanya merepresentasikan sebuah *string* dari elemen tersebut).



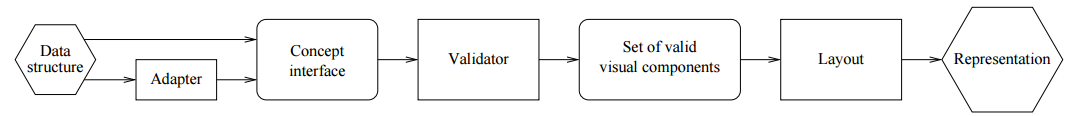
1. Turunan hirarki *concept interfaces* (Korhonen dkk., 2004)

Proses visualisasi struktur data tidak diterapkan melalui *interface FDT*, tetapi menggunakan sub-*interface* yang berada dibawahnya (lihat Gambar II.10). *Interface FDT* memiliki enam sub-*interface*. Empat diantaranya berisi Tree, Array, LinkedList dan Graph sesuai dengan empat tipe data yang disebutkan pada subbab II.7.1. *Interface* Vertex merepresentasikan satu simpul dari graf dan *interface* CDT digunakan untuk mengenali tipe data *conceptual*. Keempat *interface* tersebut sesuai dengan empat sub-*class* pada VisualContainer yang ditunjukkan pada Gambar II.9. *Matrix framework* dapat mengenali sebuah objek dengan menerapkan satu atau lebih dari keempat *interface* tersebut dan mampu untuk memvisualisasikannya menggunakan *layout* tertentu. *Interface* LinkedList berisi beberapa metode untuk *primitive* *simulation* dalam struktur implementasi *interface* ini. *Primitive* *simulation* merupakan kemampuan untuk langsung mengubah variabel-variabel pada sebuah struktur data. Termasuk menambah, menghapus dan mengubah isi variabel.

*Interface* Tree dan Graph berisi sub-*interface* yang memiliki fungsi untuk melengkapi metode yang dibutuhkan oleh *primitive* *simulation*. Untuk tipe data *array*, konsep *primitive* *simulation* tidak berguna, karena beberapa variabel seperti indeks dari *array* tidak dapat diubah. Hal ini harus dicatat, sehingga ketika sebuah struktur data diterapkan, sebagai contoh, *interface* Array dan Tree (tidak termasuk SimulationTree, akan dijelaskan kemudian) hanya dapat dimanipulasi menggunakan *primitive* *simulation* ketika divisualisasikan menggunakan *array*.

### II.7.3 Konstruksi Visual

Proses representasi visual untuk struktur data diilustrasikan seperti pada Gambar II.11. Proses visualisasi dibagi menjadi tiga tahap, yaitu *Adaptation*, *Validation*, dan *Laying out*. Pada proses *Adaptation* (bagian *Adapter*) terjadi pencocokan visualisasi struktur data terhadap konsep visual *interfaces*. Namun keadaan ini akan membuat sulit bagi *programmer* untuk mengimplementasi *interface* mana yang efisien sesuai kebutuhan visualnya. Oleh karena itu, sistem menyediakan *Adapter* untuk menangani visualisasi terhadap beberapa konsep struktur data seperti yang telah dijelaskan pada Gambar II.9 sebelumnya.



1. Proses konstruksi representasi visual (Korhonen dkk., 2004)

### II.7.4 Animasi

Animasi dalam *Matrix framework* dibagi menjadi dua bagian besar: *forward animation*, berfungsi untuk merepresentasikan secara visual struktur data yang akan terjadi perubahan pada langkah berikutnya, dan *backward animation*, berfungsi untuk menampilkan kembali visualisasi struktur data yang telah terjadi perubahan pada langkah sebelumnya.

Sistem dapat secara otomatis mendukung *forward animation* terhadap semua jenis bentuk struktur data terhadap *concept interfaces* yang telah dijelaskan pada Subbab II.7.2. Untuk mendukung *backward animation* dibutuhkan fitur lain yang cukup sulit dalam memrogramnya. Sebuah animasi dalam *Matrix framework* disimpan dalm bentuk rangkaian. Sehingga walaupun pengguna menggunakan fitur *backward animation*, sistem akan mengembalikan bentuk struktur data yang relevan sesuai konfigurasi awal. Namun, hal tersebut sangat sulit menyimpan perubahannya yang dibuat secara acak sebagai data objek Java. Oleh sebab itu, variabel yang berisi struktur data harus disimpan ke objek khusus yang disebut MemoryStructure. Objek-objek tersebut juga merepresentasikan beberapa data primitif Java, objek *array*, dan mendukung penyimpanan pada tiap perubahan yang terjadi pada struktur data. Hal itu akan sulit dilakukan pada *framework* jika tanpa objek khusus tersebut untuk menggunakan fitur *backward animation*.

### II.7.5 Simulasi

Setiap operasi simulasi yang dilakukan oleh pengguna untuk menginterpretasikan sebuah visualisasi dilakukan melalui VisualType (lihat Gambar II.2). Dari VisualType ditentukan bentuk mana yang cocok dengan struktur data yang akan direpresentasikan. Kemudian akan memanggil salah satu sub metode yang berada dibawahnya. Setelah sub metode tersebut cocok dan sukses untuk representasi struktur data, VisualType akan ditandai sebagai *class* yang *invalid*. Sistem akan mengabaikan VisualType dan dilanjutkan proses ke sub metode terkait untuk dilakukan *layout* dan merender ulang visual struktur datanya.

### II.7.6 Tampilan Antarmuka Pengguna

*Graphical user interface (GUI)* atau tampilan antarmuka pengguna merupakan bagian utama aplikasi untuk interaksi antara sistem dan pengguna menggunakan *Matrix framework*. GUI berisi beberapa bagian Panel yang berkaitan dengan fitur visualisasi struktur data atau komponen lain.

## II.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi

Studi literatur yang telah dilakukan memberikan pemahaman mendasar mengenai konsep graf dan macam algoritmanya yang telah berkembang, revolusi proses pembelajaran melalui teknologi web, terminologi visualisasi perangkat lunak, kakas VP yang telah berkembang hingga saat ini, serta peran penting eksekusi kode program dalam pembelajaran pemrograman. Dari hasil eksplorasi kakas VP, diperoleh bahwa tidak ada satu pun kakas VP yang beroperasi berbasis web dengan bahasa C atau C++ yang dapat menampilkan graf. Sehingga ini menjadi peluang untuk penelitian lebih lanjut dalam pengembangannya.

OPT menjadi dasar pengembangan kakas, karena selain telah mendukung basis web, kakas ini juga bersifat bebas dan bersumber kode terbuka. Kakas ini juga masih dalam proses pengembangan dan penelitian oleh pengembangnya[[3]](#footnote-3). Jadi, masih terdapat banyak peluang dan celah untuk diteliti lebih lanjut agar dapat dimanfaatkan dengan sebaik-baiknya. Mengingat teknologi web dan aplikasi piranti bergerak terus berkembang pesat hingga dekade tahun terakhir ini[[4]](#footnote-4).

# Bab III Analisis Masalah Awal

Bab ini menganalisis kakas VP yang akan dikembangkan, selanjutnya dilakukan penentuan ruang lingkup pengembangan kakas terhadap proses visualisasi graf dan peluang pemecahan masalah.

## III.1 Analisis dan Peluang Pemecahan Masalah

Beberapa permasalahan muncul saat mulai mengembangkan kakas OPT dan penambahan fitur graf. Eksplorasi kakas OPT dilakukan dengan meninjau proses kerja sistem, alur kode program dan membaca baris-baris kode pengembangnya. Versi terkahir yang dikembangkan telah bermigrasi ke bahasa pemrograman *Typescript*. Pengembang berargumen agar dapat dikembangkan lebih lanjut dengan teknik modularisasi[[5]](#footnote-5). Ini bermanfaat ketika peneliti atau pengembang lain yang ingin melanjutkan dengan penambahan fitur dapat dilakukan dengan mudah.

Sebagai langkah awal pengembangan kakas dengan metode eksplorasi, diperoleh beberapa poin penting, diantaranya terdapat pada Tabel III.1.

1. Hasil Analisis dan Peluang Pemecahan Masalah

| **No.** | **Domain Persoalan** | **Kakas OPT** | **Rekomendasi Perbaikan** | **Peluang Pengembangan** |
| --- | --- | --- | --- | --- |
| 1 | Aspek Simulasi dan Visualisasi | Hanya mampu menampilkan visual dengan tipe data primitif | Menggunakan *Matrix Framework* untuk mendukung visualisasi graf dengan memanfaatkan D3JS. | Dapat mendukung visual struktur data lain dengan bahasa pemrograman yang lebih variatif |
| 2 | Aspek pertukaran informasi berupa JSON antara *client* dan *server* | Pertukaran data JSON yang masih memiliki latensi tinggi | Memerlukan koneksi *Distributed Data Protocol* (DDP) yang dimiliki oleh pustaka *MeteorJS*[[6]](#footnote-6). | Peningkatan kapasitas fungsi komunikasi antar *client* dan *server* serta mendukung versi *mobile* dan *desktop* |
| 3 | Aspek Pola Desain (*Design Pattern*) | Masih tahap awal migrasi ke pemrograman *TypeScript* | Dapat menggunakan pola *Hierarchical Model View Controller* (HMVC) | Kakas dapat dikembangkan dengan modularisasi |

## III.2 Penunjang Pengembangan Kakas

Untuk menunjang pengembangan kakas diperlukan seperangkat kebutuhan yang harus dipenuhi, baik dari sisi perangkat lunak yang mutakhir maupun perangkat kerasnya. Subbab berikut akan merinci spesifikasi kebutuhan sistem.

### III.2.1 Kebutuhan Perangkat Lunak

*Typescript* merupakan bahasa pemrograman berbasis objek dari *JavaScript* yang cocok untuk pengembangan kakas OPT. Bahasa ini dibuat oleh Microsoft pada tahun 2012 untuk menunjang pengembangan aplikasi web *JavaScript* berskala besar dengan konsep pemrograman berbasis objek.

Rincian kakas dan *library* pendukung untuk pengembangan kakas OPT yang telah terintegrasi dapat dilihat pada Tabel III.2 berikut ini.

1. Kakas dan *library* pendukung pengembangan OPT

| **No.** | **Nama** | **Versi** | **Keterangan** |
| --- | --- | --- | --- |
| 1 | NodeJS | 4.5.0 | *JavaScript runtime* |
| 2 | Npm | 2.15.9 | *Node package manager*, memanajemen *library* dari *node* |
| 3 | Webpack | 3.5.5 | *Bundle js, ts, css, image* |
| 4 | Typescript | 1.8.10 | Bahasa pemrograman utama |
| 5 | Typings | 2.1.1 | *Typescript definitions* |
| 6 | Ace Code Editor | 1.2.8 | *Text editor* pada aplikasi web untuk *input* kode program |
| 7 | D3.js | 2.0 | Pustaka *Data-Driven Documents* |
| 8 | jQuery | 3.0.0 | *Cross-platform JavaScript library* |
| 9 | jQuery.bbq | 1.3pre | *Back Button and Query Library* |
| 10 | jQueryUI | 1.11.4 | *JavaScript Library* untuk interaksi antarmuka pengguna |
| 11 | jQuery.qtip | 2.0.0 | *jQuery tooltip plugin* |
| 12 | RequireJS | 2.1.20 | *JavaScript module loader*. *Framework* untuk manajemen dependensi. |
| 13 | jQuery.jsPlumb | 1.3.10 | *Library* visual *connectivity* untukaplikasi web |
| 14 | jQuery.simplemodal | 1.4.4 | *Lightweight jQuery plugin* untuk antarmuka *modal* dialog |

### III.2.2 Kebutuhan Perangkat Keras

Untuk beroperasi dengan normal, OPT dapat berjalan dengan baik pada komputer dengan spesifikasi minimal sebagai berikut:

* *Processor core i3*
* *RAM 2GB*
* Sistem operasi Linux/Mac/Windows produksi di atas tahun 2010.

# DAFTAR PUSTAKA

Bonk, C.J. (2009): *The world is open: how Web technology is revolutionizing education*, 1st ed, San Francisco, Calif, Jossey-Bass.

Bostock, M., Ogievetsky, V. and Heer, J. (2011): D$^3$ data-driven documents, *IEEE Trans. Vis. Comput. Graph.*, **17**, 2301–2309.

Boulay, B.D. (1986): Some Difficulties of Learning to Program, *J. Educ. Comput. Res.*, **2**, 57–73.

Cetin, I. and Andrews-Larson, C. (2016): Learning sorting algorithms through visualization construction, *Comput. Sci. Educ.*, **26**, 27–43.

Clancy, M., Titterton, N., Ryan, C., Slotta, J. and Linn, M. (2003): New roles for students, instructors, and computers in a lab-based introductory programming course, *ACM SIGCSE Bulletin*, ACM, 132–136, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=611951 (accessed 21 July 2016).

Clark, R.C. and Mayer, R.E. (2011): *E-learning and the science of instruction : proven guidelines for consumers and designers of multimedia learning - 3rd ed.*, San Francisco, CA, Pfeiffer.

Gračanin, D., Matković, K. dan Eltoweissy, M. (2005): Software visualization, Innov. Syst. Softw. Eng., 1, 221–230.

Guo, P.J. (2013): Online python tutor: embeddable web-based program visualization for cs education, Proceeding of the 44th ACM technical symposium on Computer science education, ACM, 579–584.

Harary, F. (1969): *Graph theory*, Canada, USA, Addison-Wesley Publishing Company, Inc.

Helminen, J. dan Malmi, L. (2010): Jype-a program visualization and programming exercise tool for Python, Proceedings of the 5th international symposium on Software visualization, ACM, 153–162.

Holmberg, N., Wünsche, B. and Tempero, E. (2006): A framework for interactive web-based visualization, *Proceedings of the 7th Australasian User interface conference-Volume 50*, Australian Computer Society, Inc., 137–144, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=1151778 (accessed 15 September 2017).

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. and Tenhunen, P. (2004): *Matrix - a framework for interactive software visualization*, Research Report TKO-B 154/04, Department of Computer Science and Engineering, Helsinki University of Technology, 26–35.

Mayer, R.E. (2014): *The Cambridge handbook of multimedia learning - Second Edition*, New York, USA, Cambridge University Press.

Pathania, U. dan Singh, A. (2014): Visualization Tool for Tree and Graph Algorithms with Audio Comments, Int. J. Softw. Web Sci. IJSWS, 14, 51–58.

Rinaldi Munir (2010): *Matematika Diskrit Edisi 3*, Bandung, Informatika Bandung.

Sorva, J. (2012): *Visual program simulation in introductory programming education*, Aalto University publication series Doctoral dissertations, Espoo, Aalto Univ. School of Science.

Sorva, J. dan Sirkiä, T. (2010): UUhistle: a software tool for visual program simulation, Proceedings of the 10th Koli Calling International Conference on Computing Education Research, ACM, 49–54.

Sorva, J., Karavirta, V. dan Malmi, L. (2013): A review of generic program visualization systems for introductory programming education, ACM Trans. Comput. Educ. TOCE, 13, 15.

TP026B, R. (n.d.): Rational Unified Process, diperoleh melalui situs internet: http://www.cis.famu.edu/~cen5055joe/Lectures/1251\_bestpractices\_TP026B.pdf (accessed 16 September 2017).

Valiente, G. (2002): *Algorithms on Trees and Graphs*, Berlin, Heidelberg, Springer Berlin Heidelberg, diperoleh melalui situs internet: http://link.springer.com/10.1007/978-3-662-04921-1 (accessed 15 September 2016).

Wang, P.S. (2015): *From computing to computational thinking*, CRC Press, Florida, USA.

Ware, C. (2004): *Information visualization: perception for design, 2nd edition*, 2nd edition, San Francisco, Kanada, Elsevier Inc.

1. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-1)
2. https://togetherjs.com/ [↑](#footnote-ref-2)
3. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-3)
4. https://kominfo.go.id/index.php/content/detail/3415/Kominfo+%3A+Pengguna+Internet+di+In donesia+63+Juta+Orang/0/berita\_satker [↑](#footnote-ref-4)
5. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-5)
6. https://www.meteor.com/ [↑](#footnote-ref-6)