**PENGEMBANGAN KAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

**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**

**PENGEMBANGAN KAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

Tanggal \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

# DAFTAR ISI

HALAMAN PENGESAHAN i

[DAFTAR ISI ii](#_Toc492533476)

[DAFTAR GAMBAR DAN ILUSTRASI iv](#_Toc492533477)

[DAFTAR TABEL v](#_Toc492533478)

[Bab I Pendahuluan 1](#_Toc492533479)

[I.1 Latar Belakang 1](#_Toc492533480)

[I.2 Rumusan Masalah 3](#_Toc492533481)

[I.3 Tujuan 3](#_Toc492533482)

[I.4 Batasan Masalah 3](#_Toc492533483)

[I.5 Metodologi Penelitian 4](#_Toc492533484)

[I.6 Sistematika Penulisan 5](#_Toc492533485)

[Bab II Tinjauan Pustaka 6](#_Toc492533486)

[II.1 Struktur Data Graf 6](#_Toc492533487)

[II.2 Revolusi Teknologi Web dan *Computational Thinking* 11](#_Toc492533488)

[II.3 Terminologi Visualisasi Perangkat Lunak 14](#_Toc492533489)

[II.4 Peran Penting Eksekusi Kode dalam Pembelajaran Pemrograman 16](#_Toc492533490)

[II.5 Hasil Eksplorasi Kakas Visualisasi Program 17](#_Toc492533491)

[II.6 Eksplorasi Kakas *Online Python Tutor* 19](#_Toc492533492)

[II.6.1 Arsitektur Kakas OPT 20](#_Toc492533493)

[II.6.2 Komponen Kakas OPT 22](#_Toc492533494)

[II.6.3 *Capturing Execution Trace* 22](#_Toc492533495)

[II.6.4 *Execution Trace Format* 23](#_Toc492533496)

[II.6.5 Fitur *Data-Driven Documents* (D3) *Framework* 24](#_Toc492533497)

[II.7 *Matrix Framework* 25](#_Toc492533498)

[II.7.1 Visualisasi 25](#_Toc492533499)

[II.7.2 Struktur 26](#_Toc492533500)

[II.7.3 Konstruksi Visual 28](#_Toc492533501)

[II.7.4 Animasi 28](#_Toc492533502)

[II.7.5 Simulasi 29](#_Toc492533503)

[II.7.6 Tampilan Antarmuka Pengguna 29](#_Toc492533504)

[II.8 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi 29](#_Toc492533505)

[Bab III Analisis Masalah 31](#_Toc492533506)

[III.1 Analisis Masalah dan Peluang Pemecahan 31](#_Toc492533507)

[III.2 Analisis Optimasi Pengembangan Kakas 32](#_Toc492533508)

[III.3 Perumusan Skema Visual Graf 32](#_Toc492533509)

[III.3.1 Menggunakan *Template* 32](#_Toc492533510)

[III.3.2 Perbaikan *Execution Trace Format* 32](#_Toc492533511)

[III.4 Penunjang Pengembangan Kakas 32](#_Toc492533512)

[III.4.1 Kebutuhan Perangkat Lunak 32](#_Toc492533513)

[III.4.2 Kebutuhan Perangkat Keras 33](#_Toc492533514)

[Bab IV Pengembangan Kakas 34](#_Toc492533515)

[IV.1 Perbaikan Arsitektur Kakas 34](#_Toc492533516)

[IV.2 *Execution Trace Format* yang Baru 35](#_Toc492533517)

[IV.3 Implementasi Kakas 35](#_Toc492533518)

[Bab V Pengujian dan Evaluasi Kakas 38](#_Toc492533519)

[V.1 Pengujian Kakas 38](#_Toc492533520)

[V.1.1 Pengujian Fungsional Kakas 38](#_Toc492533521)

[V.1.2 Desain Eksperimen 38](#_Toc492533522)

[V.2 Evaluasi Kakas 38](#_Toc492533523)

[V.2.1 Tujuan Pengembangan Kakas 38](#_Toc492533524)

[V.2.2 Indikator Keberhasilan Kakas 38](#_Toc492533525)

[V.2.3 Perbandingan Kakas Lama dan Baru 38](#_Toc492533526)

[V.2.4 Kelebihan dan Kekurangan Kakas 38](#_Toc492533527)

[Bab VI Kesimpulan dan Saran 39](#_Toc492533528)

[VI.1 Kesimpulan 39](#_Toc492533529)

[VI.2 Saran 39](#_Toc492533530)

[DAFTAR PUSTAKA 40](#_Toc492533531)

# DAFTAR GAMBAR DAN ILUSTRASI

[Gambar II.1 (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969) 6](#_Toc485359913)

[Gambar II.2 Model visual graf secara umum: (a) Pohon dan (b) Graf 11](#_Toc485359914)

[Gambar II.3 Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004) ..... 26](#_Toc485359915)

[Gambar II.4 Turunan hirarki *concept interfaces Matrix* (Korhonen dkk., 2004) ...... 27](#_Toc485359916)

[Gambar II.5 Proses konstruksi representasi visual (Korhonen dkk., 2004) 28](#_Toc485359917)

[Gambar III.1 Tampilan Antarmuka OPT untuk Bahasa Pemrograman C 19](#_Toc485359918)

[Gambar III.2 Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman 20](#_Toc485359919)

[Gambar III.3 Tampilan Simulasi dan Visualisasi Kode C Pointer 20](#_Toc485359920)

[Gambar III.4 Arsitektur OPT untuk visualisasi C dan C++ 21](#_Toc485359921)

[Gambar III.5 Ilustrasi Sederhana Proses Kakas OPT 22](#_Toc485359922)

[Gambar III.6 Ilustrasi Format Galur Eksekusi Program 23](#_Toc485359923)

[Gambar III.7 Peran Utama D3 sebagai *Framework* Visualisasi 24](#_Toc485359924)

[Gambar IV.1 Perbaikan Arsitektur untuk Pengembangan Kakas OPT 34](#_Toc485359925)

# DAFTAR TABEL

[Tabel II.1 Daftar nama algoritma graf dan model visualnya (Valiente, 2002) .....7](#_Toc485360096)

[Tabel II.2 Daftar nama algoritma pohon dan model visualnya (Valiente, 2002) .....9](#_Toc485360097)

[Tabel II.3 Perkembangan kakas VP untuk bahasa pemrograman C/C++ 17](#_Toc485360098)

[Tabel III.1 Hasil Analisis dan Peluang Pemecahan Masalah 31](#_Toc485360099)

# Bab I Pendahuluan

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

## I.1 Latar Belakang

Graf adalah salah satu pokok bahasan perkuliahan Algoritma Pemrograman dan Struktur Data dalam bidang informatika atau ilmu komputer (Valiente, 2002). Graf merupakan topik penting dalam perkuliahan Struktur Data. Graf bermanfaat untuk memodelkan suatu masalah dengan cara merepresentasikan objek-objek diskrit dan hubungan antar objek tersebut. Pohon (*tree*) merupakan salah satu bentuk khusus dari graf (Valiente, 2002). Suatu graf terhubung yang setiap pasangan simpulnya hanya dapat dihubungkan oleh suatu lintasan tertentu, maka graf tersebut dinamakan pohon (*tree*). Pohon (*tree*) merupakan graf tak-berarah yang terhubung dan tidak memiliki sirkuit.

Pada abad 21 ini, memiliki kemampuan dalam pemrograman merupakan keahlian yang sangat bernilai (Helminen dan Malmi, 2010; Margulieux dkk., 2016). Oleh sebab itu, banyak pelajar ingin memasuki informatika atau bidang komputer lainnya (Cetin, 2016; Piteira dan Costa, 2013). Namun, situasi tersebut tidak diimbangi dengan kualitas kemampuan pemrograman yang harus dimiliki. Untuk mempelajari pemrograman, pelajar harus memiliki pengetahuan yang baik tentang deklarasi dan prosedural program, daya ingat yang kuat, pemahaman, penyelesain masalah, abstraksi dan kemampuan logika program (Piteira dan Costa, 2013).

Piteira dan Costa (2013) melakukan identifikasi masalah dalam perkuliahan pemrograman. Aspek yang dianalisis adalah isi materi dari perkuliahan dan aspek pembelajaran. Disimpulkan bahwa konsep pemrograman di perkuliahan yang paling sulit adalah tipe struktur data graf. Struktur data merupakan subjek penting dalam bidang komputasi, namun pelajar masih mengalami kesulitan dalam memahami, mengeksekusi dan mengonsep algoritmanya (Pathania dan Singh, 2014). Dari aspek pembelajaran yang paling signifikan pelajar sangat terbantu melalui contoh-contoh eksekusi kode program, karena merupakan implementasi dari konsep yang dipelajarinya.

Visualisasi telah banyak direkomendasikan menjadi kakas penting untuk memahami konsep pemrograman (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013; Gračanin dkk., 2005; Guo, 2013). Melalui interaksi visual, manusia lebih cenderung menangkap lebih banyak informasi yang diterima dibandingkan melalui indera lainnya (Ware, 2004). Oleh sebab itu visualisasi menjadi bagian penting dalam pembelajaran pemrograman (Sorva dkk., 2013). Istilah “visualisasi perangkat lunak” 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 algoritma, sedangkan VP berkaitan dengan eksekusi kode program dan proses struktur datanya. Melalui VP dapat memberikan pemahaman lebih baik kepada pelajar dalam pembelajaran pemrograman (Sorva, 2012).

Teknologi internet dan web digunakan untuk kemudahan akses dalam pembelajaran pemrograman (Bonk, 2009). Saat ini kakas VP berbasis web yang mampu melakukan visual graf belum tersedia (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*. Pada penelitian tesis ini, penulis akan mengembangkan kakas VP untuk graf berbasis web dari kode sumber *Online Python Tutor* (OPT). Kakas ini selain telah mendukung teknologi web untuk *e-learning*, juga bersifat bebas (*free*) dan bersumber terbuka (*open source*) (Guo, 2013). Awal pengembangan OPT tahun 2010 hanya mendukung bahasa pemrograman *Python*. Namun saat ini telah mendukung banyak bahasa pemrograman seperti *Java*, *C*, *C++*, *Ruby* dan *JavaScript*. Diharapkan pengembangan kakas kali ini dapat mendukung pembelajaran pemrograman graf.

## I.2 Rumusan Masalah

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

1. Bagaimana mengembangkan kakas VP berbasis OPT untuk graf dalam bahasa C dan C++ ?
2. Bagaimana kakas dapat mendeteksi adanya kode program graf atau pohon sehingga proses visualisasi sesuai dengan algoritmanya ?
3. Bagaimana peran VP yang dikembangkan terhadap proses pembelajaran algoritma dan pemrograman untuk graf ?

## I.3 Tujuan

Sebagai solusi dari masalah yang telah diuraikan, maka tujuan dari penelitian tesis ini adalah :

1. Menghasilkan kakas VP berbasis web untuk bahasa C dan C++ yang dapat mendukung pembelajaran pemrograman graf.
2. Menghasilkan kakas VP berbasis web yang dapat mendukung pembelajaran pemrograman graf sesuai algoritma yang sedang digunakan.

## I.4 Batasan Masalah

Latar belakang dan rumusan masalah telah diuraikan, maka untuk memfokuskan kegiatan penelitian tesis ini diperlukan batasan, sehingga cakupan pembahasan dan penyelesaian tidak meluas. Adapun batasan masalah pada tesis ini adalah :

1. Implementasi kakas VP untuk visual graf berdasar kode sumber dari OPT.
2. Bahasa pemrograman yang dapat divisualisasikan adalah C dan C++.
3. Kode program yang dapat divisualisasi minimal memiliki syarat terbentuknya data objek dalam graf, yaitu *array* atau *pointer*.
4. Kakas tidak menerima *standard input* (*stdin*) dan *include multiple class object files*.

## I.5 Metodologi Penelitian

Penelitian dimulai dengan studi literatur, eksplorasi, dan identifikasi masalah untuk memformulasikan pertanyaan-pertanyaan yang berkaitan dengan permasalahan belajar algoritma pemrograman graf dan perkembangan kakas VP. Tinjauan pustaka dari penelitian sebelumnya, buku, jurnal dan artikel ilmiah, dan situs internet yang berelasi dengan pengembangan kakas VP serta konsep dalam pembelajaran algoritma pemrograman graf, sehingga penelitian pada tesis ini akan dilaksanakan melalui beberapa tahapan sebagai berikut.

1. Studi Literatur dan Eksplorasi

Tahap ini dilakukan pendalaman mengenai konsep umum pembelajaran pemrograman, pedagogi, VP, proses cara berpikir komputer (*computational thinking*) (Sorva, 2012), serta implementasi dari kakas yang telah dibangun. Studi literatur menghasilkan kesimpulan awal yang dapat dijadikan landasan dalam perumusan masalah dan pengembangan kakas. Kemudian dilakukan eksplorasi berbagai kakas VP yang telah ada.

1. Analisis Masalah dan Perumusan Skema Graf

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

1. Pembangunan Perangkat Lunak

Hasil analisis dan rancangan sistem pada tahap sebelumnya akan diimplementasikan dalam pembangunan kakas VP berbasis web, kemudian akan dilakukan pengujian dan evaluasi dari hasil implementasi tersebut.

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 efektifitas peran kakas dalam membantu pemahaman terhadap kode program yang sedang dipelajarinya.

## I.6 Sistematika Penulisan

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

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, serta metodologi yang digunakan pada pengerjaan tesis, serta sistematika pembahasan dari laporan tesis.

1. Bab II Tinjauan Pustaka

Bab ini berisi uraian tentang konsep belajar mengajar pemrograman, peran dan perkembangan kakas VP dari hasil penelitian sebelumnya yang berkaitan dengan masalah yang dikaji, sehingga memberikan gambaran perkembangan terhadap masalah yang akan diteliti.

1. Bab III Analisis Masalah

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 Pembangunan Perangkat Lunak

Bab ini menjelaskan proses pengembangan dan implementasi kakas VP berdasarkan rancangan yang telah dirumuskan pada Bab III sebelumnya.

1. Bab V Pengujian dan Evaluasi Kakas

Bab ini menguraikan proses dan hasil pengujian serta evaluasi dari kakas yang dikembangkan pada tesis ini. Pengujian dilakukan untuk membuktikan bahwa kakas berhasil dikembangkan lebih baik dibandingkan sebelumnya. Evaluasi kakas dilakukan untuk menilai efektivitas dan kualitas dari visualisasi untuk mendukung pemahaman dalam pembelajaran algoritma pemrograman.

1. Bab VI Penutup

Bab ini berisi kesimpulan dalam tesis serta saran untuk pengembangan kakas lebih lanjut dari penelitian tesis ini.

# Bab II Tinjauan Pustaka

Pada bab ini dijelaskan mengenai ruang lingkup struktur data graf dan algoritmanya, terminologi visualisasi perangkat lunak, peran kakas VP dalam proses pembelajaran pemrograman, dan hasil eksplorasi dari berbagai kakas VP.

## II.1 Struktur Data Graf

Graf dan pohon merupakan salah satu pokok bahasan dalam Matematika Diskrit dan konsep dasar yang harus dipahami oleh pelajar dalam bidang informatika atau ilmu komputer. Dalam perkuliahan Algoritma Pemrograman dan Struktur Data, terdapat pula topik tentang graf, namun lebih cenderung kepada implementasi terhadap pemrogramannya. Karena memang pelajar bidang informatika atau ilmu komputer dituntut untuk mampu membuat kode program dalam menyelesaikan suatu permasalahan tertentu.

Sejarah mencatat bahwa model graf diterapkan pertama kali sekitar Abad 17 pada tahun 1736, yaitu permasalahan jembatan Königsberg (lihat Gambar II.1). Kota Königsberg (sebelah timur Prussia, Jerman sekarang) yang berubah nama menjadi kota Kaliningrad. Di kota itu terdapat sungai Pregal yang mengalir mengelilingi pulau Kneiphof lalu bercabang menjadi dua buah anak sungai (Rinaldi Munir, 2010).

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

1. (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969)

Pada Gambar II.1(a) terdapat tujuh buah jembatan yang saling menghubungkan antar daratan yang dialiri sungai tersebut. Permasalahannya adalah apakah mungkin dapat melalui ketujuh jembatan tersebut tepat satu kali dan kembali ke tempat semula ?. Beberapa penduduk kota tersebut sepakat bahwa tidak mungkin mereka dapat melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan. Tetapi mereka tidak dapat menjelaskan secara ilmiah alasannya. Euler adalah seorang matematikawan Swiss yang berhasil menemukan jawaban terhadap masalah tersebut pada tahun 1736. Ia memberikan bukti sederhana dengan memodelkan masalah ini ke dalam model graf. Titik (noktah) menyatakan daratan yang dihubungkan oleh jembatan, yang disebut simpul (*vertex*). Garis merepresentasikan jembatan, yang disebut sisi (*edge*). Setiap titik terdapat label huruf A, B, C, dan D. Euler membuat model graf seperti terlihat pada Gambar II.1(b).

Euler mengemukakan jawabannya bahwa orang tidak mungkin melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Derajat merupakan banyak garis yang bersisian dengan noktah. Contoh simpul C memiliki derajat tiga karena ada tiga buah garis yang bersisian dengannya, simpul B dan D juga berderajat tiga, sedangkan simpul A berderajat lima. Karena semua simpul tidak berderajat genap, maka tidak mungkin dilakukan perjalanan berupa sirkuit (yang selanjutnya disebut sirkuit Euler) pada graf tersebut.

Graf memiliki berbagai macam algoritma dan model visualnya masing-masing, sehingga perlu dijabarkan lebih rinci agar mengetahui konsep dasar terhadap proses kerja dari kakas VP yang akan dibangun. Daftar nama-nama algoritma dan model visual graf yang ditampilkan pada Tabel II.1 berikut merupakan yang paling sering digunakan dalam perkuliahan Struktur Data dan Algoritma Pemrograman.

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.2 (a) dan model visual graf pada Gambar II.2 (b).

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

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

## II.2 Revolusi Teknologi Web dan *Computational Thinking*

Bonk (2009) menulis sebuah buku berjudul “*The World Is Open: How Web Technology Is Revolutionizing Education*” yang menjelaskan tentang teknologi web dan internet mengubah cara belajar di dunia pada awal abad 21 ini. Internet telah membuka peluang besar bagi seluruh manusia di bumi. Pembelajaran elektronik (*e-learning*) telah menjadi bagian penting dari setiap komunitas belajar. Saat ini keuntungan bagi para pengajar untuk berbagi ilmu bukan merupakan hal yang sulit. Secara nyata kehadiran internet telah mengubah cara berbagi kepada para pelajar di dunia, kelas, sekolah, kampus, dan masih banyak sisi yang berpotensi terkena dampak positifnya. Melalui internet atau *online sharing* dapat mempengaruhi setiap orang. Khususnya istilah berbagi (*share*) telah populer dalam lingkungan pembelajaran melalui teknologi daring secara kolaboratif.

Abad 21 ini telah membawa informasi ke revolusi dunia digital. Masyarakat umum harus dapat beradaptasi dengan penggunaan komputer, ponsel pintar, dan internet. Dengan kata lain, mereka harus dapat mencapai kemampuan berpikir yang disebut sebagai *computational thinking* (Wang, 2015). *Computational thinking* (CT) merupakan keterampilan mental untuk menerapkan konsep-konsep dasar dan penalaran dalam berpikir, yang berasal dari digital komputer modern, *computer science*, yang mencangkup segala aspek kehidupan, termasuk aktivitas manusia sehari-hari. CT merupakan cara berpikir yang terinspirasi dari cara kerja komputer dan teknologi informasi, dan berbagai keunggulannya, batasan-batasannya, serta permasalahan yang muncul. CT juga memberikan peluang untuk para peneliti tetap berpikir kritis dengan pertanyaan-pertanyaan seperti “bagaimana jika kita mengotomatisasikan hal ini?”, “instruksi dan tindakan apa yang perlu dilakukan terhadap pembelajaran ini kepada anak-anak?”, “seberapa efesienkah metode ini?”, dan “apa dampak negatifnya?”.

CT dapat memperluas daya berpikir, membantu dalam penyelesaian masalah, meningkatkan efisiensi, menghindari kesalahan, dan mengantisipasi terjadinya keterbatasan dalam berpikir, baik dalam berinteraksi dan komunikasi dengan manusia atau mesin. CT dapat membuat manusia lebih sukses bahkan dalam menjalani kehidupan sehari-harinya (Wang, 2015).

Untuk dapat memahami CT, tidak diperlukan menjadi seorang komputer saintis atau insinyur. Dari sudut pandang pengguna awam, CT dapat memberikan tampilan secara terstruktur, mudah, dan sederhana sesuai dengan topik komputer yang akan dipelajarinya. Asumsinya, pengguna memiliki sedikit pengetahuan dasar tentang *computer science* atau pemrograman. Karena ketika ditampilkan sebuah perangkat keras, perangkat lunak, representasi data, algoritma, sistem, keamanan, jaringan, situs web, dan konsep lainnya dalam segala aspek komputer, CT akan secara luas mengimplementasikan konsep-konsep dan keterampilan khusus untuk menjelaskan bagaimana atau di mana mereka dapat menerapkannya dalam kehidupan.

Semua komputer itu bodoh, ia hanya mampu mengolah data dalam bentuk angka-angka *biner* atau *bit* saja (Wang, 2015). Setiap *bit* direpresentasikan sebagai angka nol atau satu. Komputer mengeksekusi semua instruksi yang diprogram, namun hal itu tetap saja direpresentasikan sebagai 0 dan 1. Walaupun demikian, komputer merupakan mesin yang bersifat universal, dapat melakukan tugas-tugas apapun sesuai dengan instruksi program. Untuk memaksimalkan kinerja komputer maka harus diprogram, dikontrol, dan dibuat bekerja sesuai dengan yang diharapkan. Semua itu dapat dilakukan dengan memahami konsep dasar yang disebut sebagai *computational thinking*.

Beberapa aspek penting dari CT adalah sebagai berikut (Wang, 2015):

* **Penyederhanaan melalui abstraksi**. Abstraksi adalah teknik untuk mengurangi kompleksitas dengan cara mengabaikan hal-hal detail yang tidak penting dan fokus terhadap yang penting. Sebagai contoh, sopir melihat mobil langsung terlintas untuk bagaimana mengendarainya, bukan berpikir bagaimana mobil itu bekerja atau dibuat.
* **Kelebihan dalam hal automatisasi**. Menyusun hal-hal penting sehingga mudah untuk dimanajemen dan diotomatisasi. Seperti mengerjakan prosedur secara sistematis, menyusun algoritma, melakukan tugas-tugas secara terstruktur sehingga dapat meningkatkan efisiensi dan produktifitas.
* **Iterasi dan perulangan.** Merupakan teknik strategis untuk mengimplementasikan secara berulang dengan tingkat keberhasilan yang sama dan mengeksekusi beberapa langkah untuk menyelesaikan masalah.
* **Ketelitian dalam penglihatan dan pikiran**. Perubahan angka 0 ke angka 1, atau huruf kapital, dapat berpengaruh besar terhadap instruksi program. Dibutuhkan penglihatan yang teliti, pikiran yang cerdas, dan pendekatan yang cermat. Jika mengabaikan hal-hal tersebut, maka akan berakibat fatal.
* **Ketelitian dalam komunikasi melalui instruksi program**. Komputer tidak dapat langsung memahami apa yang kita maksud, namun dapat melalui kalimat atau perkataan. Oleh karena itu, memberikan instruksi program harus teliti dan lengkap. Karena ketidakjelasan instruksi tidak dapat ditoleransi dan harus dibuat secara eksplisit.
* **Logika deduktif**. Merupakan logika yang kaku atau statis. Hal ini karena tidak mempertimbangkan perasaan atau emosional manusia.
* **Kreatif dalam penyelesaian masalah**. Sebuah instruksi program dibuat untuk menyelesaikan berbagai macam tugas. Komputer tidak seperti manusia yang dapat menyelesaikan masalah berdasarkan pengalaman dan keilmuannya. Sebuah kode program pada tingkatan komputer langsung dieksekusi tanpa mempertimbangkan efeknya, sehingga pengguna harus kreatif memanfaatkan kemampuan komputer yang tersedia.
* **Mengantisipasi masalah**. Otomatisasi oleh komputer tidak selalu menjadi hal praktis dan teliti. Diperlukan rencana antisipasi untuk hal-hal di luar dugaan, seperti kerusakan sistem, tidak adanya filter dalam proses data, *error* atau *malfunction*, dan sebagainya. Berkaitan dengan CT, maka diperlukan rencana kontinjensi untuk hal-hal di luar dugaan tersebut agar tidak menjadi jebakan terhadap dirinya.

Aspek-aspek tersebut hanya beberapa ide utama dari CT, karena sebenarnya CT lebih banyak membuka konsep-konsep dan jalan berpikir yang lebih penting. Melalui peningkatan pemahaman terhadap komputer, maka proses CT telah memberikan berbagai sudut pandang yang berbeda kepada setiap individu.

## II.3 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.4 Peran Penting Eksekusi Kode dalam Pembelajaran Pemrograman

Banyak pelajar di dunia tidak mau untuk belajar eksekusi kode program (Sorva, 2012). Padahal dalam mempelajari pemrograman, harus mempelajari eksekusi kode program agar dapat memahami algoritma dan pemrograman secara utuh. Eksekusi kode program dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari konsep algoritma (Sorva, 2012). Melalui eksekusi kode, pelajar dapat teruji tingkat pemahamannya terhadap konsep algoritma yang telah dipelajarinya. Seperti halnya mengerti dan paham tentang konsep cara berenang, namun tidak pernah praktik langsung ke dalam air untuk berenang.

Ketika sebuah solusi atau algoritma telah dibuat, maka sebuah kode program tetap harus ditulis. Saat ini komputer banyak berperan penting dalam kehidupan manusia sehingga masalah yang dihadapi menjadi lebih besar dan kompleks, yang membutuhkan pengembangan kode program lebih rumit lagi. Tujuan dari penelitian tesis ini adalah untuk menghasilkan kakas VP yang dapat memberikan kemampuan analisis pemrograman dan algoritma secara simultan kepada para pelajar terhadap graf, sehingga mereka dapat mengembangkan program dengan kualitas dan kuantitas yang maksimum.

Dalam pembelajaran pemrograman sebagai disiplin ilmu yang memerlukan ketelitian tinggi ada dua hal penting yang saling melengkapi (Sorva, 2012), yaitu:

* Pemrograman merupakan pendekatan secara komputasi yang eksekusinya didefinisikan dalam sebuah mesin komputer. Dalam dunia nyata, intuisi pelajar sebenarnya dapat memahami eksekusi program khususnya pada sistem interaktif, seperti potongan-potongan kode program yang dibuat oleh pelajar dapat langsung terlihat hasil eksekusinya. Melalui teknik tersebut pelajar tidak lagi bingung apa yang akan terjadi pada program yang dibuatnya sehingga dapat mempercepat proses pemahamannya.
* Pemrograman merupakan pendekatan secara logika sebagai percabangan dari ilmu logika matematika. Logika bukan merupakan eksekusi program akan tetapi merupakan karakteristik dari program yang berjalan, sehingga dapat disebut sebagai abstraksi pada tingkatan tertinggi. Program merupakan konstruksi matematika yang harus sesuai dengan aturan logika yang berlaku. Pemrograman dengan pendekatan secara logika bagi pelajar lebih sulit untuk dipahami karena definisi dari rincian program setiap barisnya tidak akan ditampilkan.

## II.5 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 tampak 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.

Ben-Bassat Levy dkk. (Sorva, 2012) melakukan riset dalam perkuliahan pengantar algoritma pemrograman. Dijelaskan bahwa para pelajar yang menggunakan kakas VP bernama *Jeliot 2000* dalam kurun waktu beberapa tahun secara rutin memiliki dampak yang cukup signifikan dalam peningkatan belajar pemrograman. Namun setelah dianalisis terdapat istilah “*middle effect*” di antara para pelajar.

Isitilah “*middle effect*” yang dimaksud terkait dengan visualisasi dan fungsionalitasnya. Walaupun telah lama digunakan, animasi tidak membantu banyak pelajar, karena pelajar yang memiliki kapasitas intelektual cukup tinggi sudah tidak membutuhkannya, sedangkan pelajar yang memiliki kapasitas intelektual rendah akan kesulitan dalam memanfaatkan kakas. Tetapi untuk kebanyakan pelajar umumnya sebuah model yang konkrit dapat memberikan dampak baik yang cukup signifikan dalam menggapai kesuksesan belajarnya.

## II.6 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.3 ditunjukkan tampilan antarmuka kakas OPT untuk bahasa pemrograman C. Fungsi nomor 1 adalah untuk berbagi antarmuka bagi pengguna lain yang ingin belajar pemrograman secara kolaborasi. Ketika tombol tersebut diklik, tampilan tampak seperti pada Gambar II.4. 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.3 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.5.



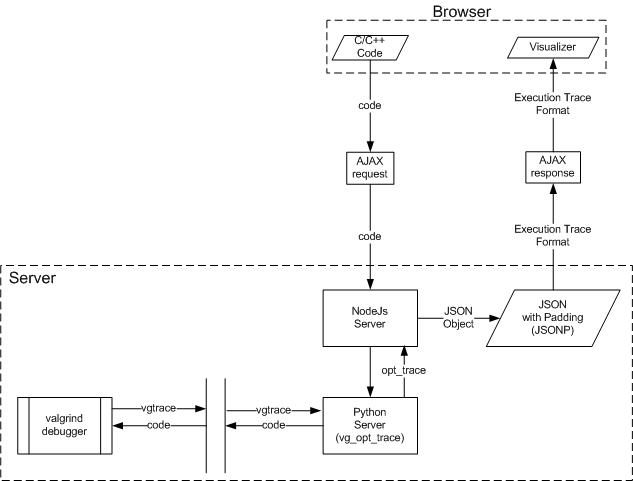
1. Tampilan Simulasi dan Visualisasi Kode C Pointer

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

### II.6.1 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.6 ditampilkan diagram alur komunikasi antara *browser* milik pengguna dengan *server*. Pertukaran informasi kode program dilakukan dengan fitur permintaan *AJAX* (*Asynchronous JavaScript And XML*).



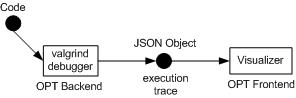
1. Arsitektur OPT untuk visualisasi C dan C++

Untuk lebih jelas tahapan proses pada Gambar II.6 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. Peladen 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.
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.5) proses visualisasi akan membaca poin indeks yang berada pada *trace* eksekusi sesuai dengan banyak langkah yang telah terbentuk.

### II.6.2 Komponen Kakas OPT

Kakas OPT dapat disederhanakan menjadi beberapa inti komponen dengan representasi abstrak seperti tampak pada Gambar II.7 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.5.

### II.6.3 *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.8.

### II.6.4 *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.8 berikut ini.



1. Ilustrasi Format *Trace* Eksekusi Program

Bagan pada Gambar II.8 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.6.5 Fitur *Data-Driven Documents* (D3) *Framework*

Teknologi visualisasi utama yang digunakan oleh OPT adalah *Data-Driven Documents* (D3) (Bostock dkk., 2011) yang merupakan salah satu pustaka terpopuler untuk implementasi visualisasi berbasis web. Format *trace* eksekusi program dibaca oleh D3 untuk mendeklarasikan pemetaan data dan atribut yang diperlukan oleh elemen *Document Object Model* (DOM) pada laman *Hyper Text Markup Language* (HTML). Peran *D3 framework* tampak pada Gambar II.9 berikut ini.



1. Peran Utama D3 sebagai *Framework* Visualisasi

*D3* *framework* sangat fleksibel terhadap abstraksi data tingkat tinggi. Pustaka ini juga mampu memvisualisasikan berbagai macam data, bahkan cocok untuk visualisasi dengan tingkat kerumitan yang tinggi.

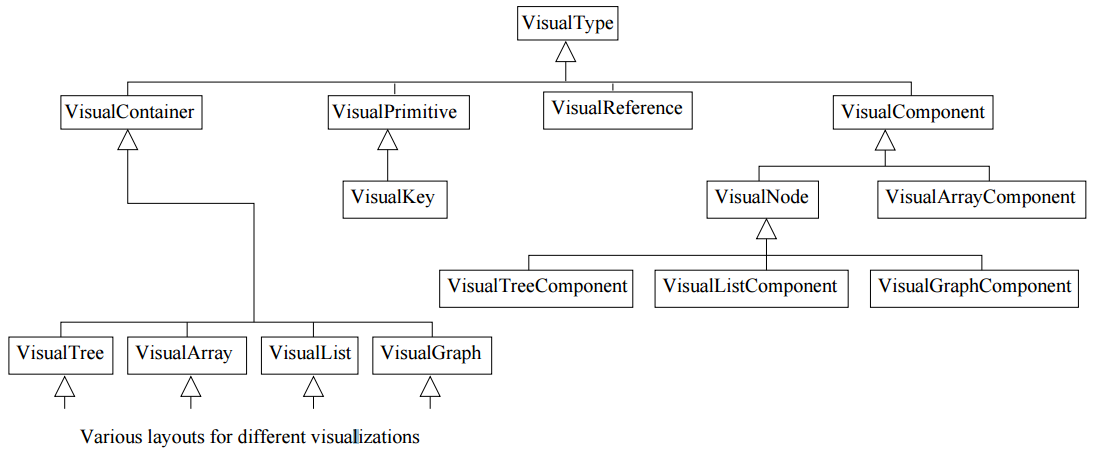
## II.7 *Matrix Framework*

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.10 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*.

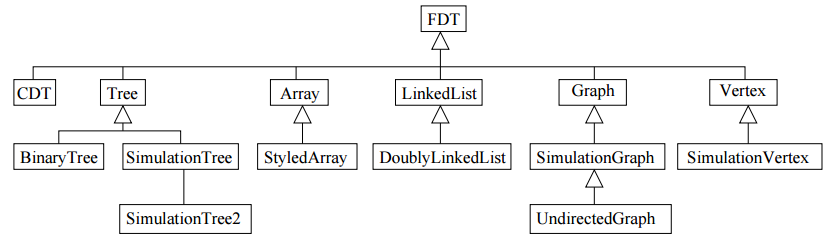


1. Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004)

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.10), 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.11. *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).



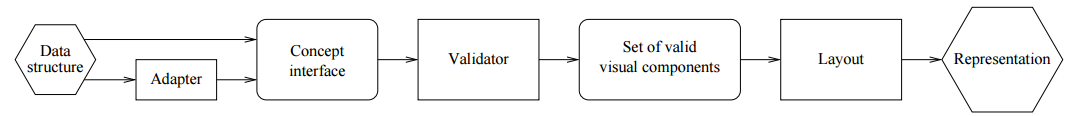
1. Turunan hirarki *concept interfaces* (Korhonen dkk., 2004)

Proses visualisasi struktur data tidak diterapkan melalui *interface FDT*, tetapi menggunakan sub-*interface* yang berada dibawahnya (lihat Gambar II.11). *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.10. *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.12. 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.10 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 *lay out* 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.8 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

Bab ini menganalisis kakas VP yang sedang dikembangkan kemudian diuraikan rumusan metode atau skema yang diperlukan untuk menerjemahkan kode program menjadi visual graf. Selanjutnya dilakukan penentuan ruang lingkup pengembangan kakas terhadap proses visualisasi graf.

## III.1 Analisis Masalah dan Peluang Pemecahan

Beberapa permasalahan muncul saat mulai mengembangkan kakas OPT dan penambahan fitur visual 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 riset dengan penambahan fitur dapat dilakukan dengan mudah. Namun, versi terakhir belum sepenuhnya memiliki pola desain (*design pattern*) pengembangan yang standar. Selain itu, dokumentasi kakas untuk pengembangan masih sangat terbatas dan belum diatur dengan baik. 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 komunikasi *JSON* antar *client* dan *server* | Hanya mampu menampung maksimal 1000 iterasi atau langkah eksekusi poin | 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* |
| 2 | Aspek Pola Desain (*Design Pattern*) | Masih tahap awal migrasi ke pemrograman *TypeScript* | Dapat menggunakan pola *Hierarchical Model View Controller* (HMVC) yang dapat bersifat modular | Kakas dapat dikembangkan dengan modularisasi |
| 3 | Aspek Simulasi dan Visualisasi | Hanya mampu menampilkan visual dengan tipe data primitif | Menggunakan *Matrix Framework* untuk mendukung visualisasi graf dengan dukungan fitur *Reactive Dictionary* yang dimiliki oleh pustaka *MeteorJS*. | Dapat mendukung visual struktur data lain dengan bahasa pemrograman yang lebih variatif |

## III.2 Analisis Optimasi Pengembangan Kakas

## III.3 Perumusan Skema Visual Graf

### III.3.1 Menggunakan *Template*

### III.3.2 Perbaikan *Execution Trace Format*

## III.4 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 memerinci spesifikasi kebutuhan sistem.

### III.4.1 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 | D3js | 2.0 | *Data-Driven Documents Framework* |
| 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.4.2 Kebutuhan Perangkat Keras

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

* *Processor core i3*
* *RAM 2GB*

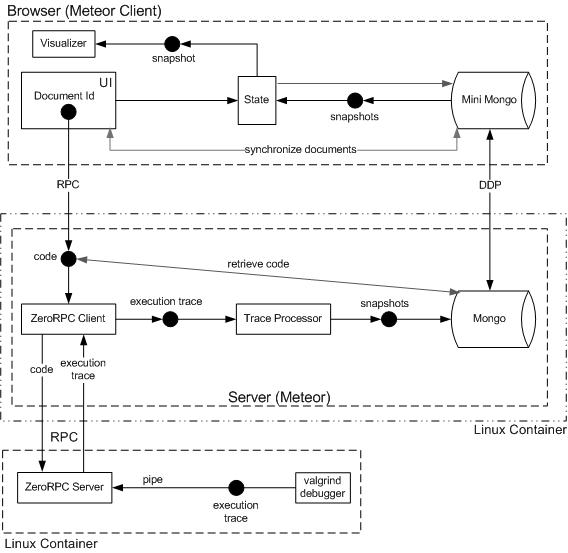
Sistem operasi Linux/Mac/Windows produksi di atas tahun 2010.

# Bab IV Pengembangan Kakas

Bab ini menjelaskan proses pengembangan kakas yang terdiri dari perbaikan arsitektur, perbaikan format *trace* eksekusi dan pola desain untuk pengembangan lebih lanjut. Salah satu tujuan pengembangan kakas ini selain untuk dukungan visualisasi graf adalah dapat dikembangkan dengan modularisasi. Bersifat *modular* akan bermanfaat untuk penelitian dan penambahan fitur-fitur selanjutnya.

## IV.1 Perbaikan Arsitektur Kakas

Apakah perlu perbaikan ini? Apakah sesuai dengan batasan masalah?



1. Perbaikan Arsitektur untuk Pengembangan Kakas OPT

## IV.2 *Execution Trace Format* yang Baru

## IV.3 Implementasi Kakas

Secara garis besar, kakas OPT belum dapat memvisualisasikan graf. Pengembangan kakas ini diharapkan dapat menambah fitur untuk visualisasi graf saat pelajar mengeksekusi kode program yang terdapat algoritma graf. Tujuannya adalah agar membantu pelajar untuk memahami eksekusi kode program graf. Metode pengembangan kakas yang dilakukan adalah sebagai berikut:

1. Lakukan eksperimen untuk merepresentasikan algoritma graf dan pohon yang ada di Tabel II.1 dan Tabel II.2 dalam bentuk visual dengan D3JS
2. Visual pada poin 1 dibangun animasi per tahap sesuai alur algoritma
3. Mencari pola pembentukan visualnya berdasarkan kode C,C++ dan *execution trace format*. Jika ada *array* atau *pointer* dalam kode program, perlu dipertimbangkan untuk menampilkan visualisasi graf
4. Untuk mengetahui cara kerja kakas terbentuknya visual OPT diperlukan sebuah abstraksi dengan membuat class diagram untuk file pytutor.ts, visualize.ts, opt-frontend.ts, opt-live.ts, opt-frontend-common.ts dan opt-shared-session.ts
5. Integrasikan pola visual dengan OPT dan Matrix Framework.

(Apakah perlu ditulis?) Perbedaan D3JS versi 2 dan 3. (<https://iros.github.io/d3-v4-whats-new/#1>)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Ruang Lingkup** | **Versi 2** | **Versi 3** | **Versi 4** |
|  |  |  |  |  |
|  |  |  |  |  |

Class Diagram OPT

|  |  |
| --- | --- |
| **pytutor.ts** | |
| **export class ExecutionVisualizer** | |
|  |  |
| add\_pytutor\_hook(**hook\_name**, **func**)  try\_hook(**hook\_name**, **args**)  generateID(**original\_id**)  render()  \_getSortedBreakpointsList()  addToBreakpoints(**executionPoints**)  removeFromBreakpoints(**executionPoints**)  setBreakpoint(**d**)  unsetBreakpoint(**d**)  findPrevBreakpoint()  findNextBreakpoint()  stepForward()  stepBack()  updateOutput(**smoothTransition = false**)  updateOutputFull(**smoothTransition**)  updateOutputMini()  renderStep(**step**)  redrawConnectors()  activateJavaFrontend()  updateLineAndExceptionInfo()  isOutputLineVisibleForBubbles(**lineDivID**) | |

|  |  |
| --- | --- |
| **pytutor.ts** | |
| **class DataVisualizer** | |
|  |  |
| height()  generateHeapObjID(**ObjID, stepNum**)  getRealLabel(label)  resetJsPlumManager()  precomputeCurTraceLayouts()  isCppMode()  structurallyEquivalent(**obj1, obj2**)  isPrimitiveType(**obj**)  renderDataStructures(**curInstr: number**)  renderPrimitiveObject(**obj, stepNum: number, d3DomElement**)  renderNestedObject(**obj, stepNum: number, d3DomElement**)  renderCompoundObject(**objID, stepNum: number, d3DomElement, isTopLevel**)  renderCStructArray(**obj, stepNum, d3DomElement**)  redrawConnectors() | |

|  |  |
| --- | --- |
| **pytutor.ts** | |
| **class ProgramOutputBox** | |
| constructor(owner, domRoot, heightOverride = null) |  |
| renderOutput(stdoutStr: string) | |

|  |  |
| --- | --- |
| **pytutor.ts** | |
| **class CodeDisplay** | |
| constructor(**owner, domRoot, domRootD3, codToDisplay: string, lang: string, editCodeBaseURL: string**) |  |
| renderPyCodeOutput() | |
| updateCodOutput(smoothTransition = false) | |

|  |  |
| --- | --- |
| **pytutor.ts** | |
| **class NavigationController** | |
| constructor(**owner, domRoot, domRootD3, nSteps**) |  |
| hideUserInputDiv() | |
| showUserInputDiv() | |
| setSliderVal(**v: number**) | |
| setVcrControls(**msg: string, isFirstInstr: boolean, isLastInstr: boolean**) | |
| setupSlider(**maxSliderVal: number**) | |
| renderSliderBreakpoints(**sortedBreakpointsList**) | |
| showError(**msg: string**) | |

|  |  |
| --- | --- |
| **visualize.ts** | |
| **export class OptFrontendWithTestcases extends OptFrontendSharedSessions** | |
| constructor(params={}) |  |
| parseQueryString() | |
| appStateAugmenter(**appState**) | |
| runTestCase(**id, codeToExec, firstTestLine**) | |
| vizTestCase(**id, codeToExec, firstTestLine**) | |
| handleUncaughtException(**trace**) | |
| finishSuccessfulExecution() | |
| experimentalPopUpSyntaxErrorSurvey() | |

# Bab V Pengujian dan Evaluasi Kakas

Bab V ini berisi pengujian dan evaluasi terhadap hasil pengembangan kakas. Pengujian selain fungsional kakas, juga akan dilakukan eksperimen terhadap pelajar sebagai penggunanya. Kemudian hasil pengujian dianalisis untuk dievaluasi terhadap tujuan pengembangan kakas yang diharapkan.

## V.1 Pengujian Kakas

### V.1.1 Pengujian Fungsional Kakas

### V.1.2 Desain Eksperimen

Desain eksperimen kakas dibagi menjadi dua bagian, yaitu melalui eksperimen simulasi mengajar atau studi kasus dan wawancara mendalam. Dalam eksperimen simulasi mengajar, dilakukan dua uji kasus yaitu pelajar menggunakan OPT dan pelajar menggunakan kakas hasil pengembangan visual graf.

## V.2 Evaluasi Kakas

### V.2.1 Tujuan Pengembangan Kakas

### V.2.2 Indikator Keberhasilan Kakas

### V.2.3 Perbandingan Kakas Lama dan Baru

### V.2.4 Kelebihan dan Kekurangan Kakas

# Bab VI Kesimpulan dan Saran

## VI.1 Kesimpulan

## VI.2 Saran

# 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. dan Heer, J. (2011): D3js 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. (2016): Preservice Teachers’ Introduction to Computing: Exploring Utilization of Scratch, *J. Educ. Comput. Res.*, 735633116642774.

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

Clancy, M., Titterton, N., Ryan, C., Slotta, J. dan 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. dan 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.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. dan 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.

Margulieux, L.E., Catrambone, R. dan Guzdial, M. (2016): Employing subgoals in computer programming education, *Comput. Sci. Educ.*, **26**, 44–67.

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.

Piteira, M. dan Costa, C. (2013): Learning computer programming: study of difficulties in learning programming, *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ACM, 75–80, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=2503871 (accessed 8 September 2016).

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.

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)