**PENGEMBANGAN KAKAS**

**VISUALISASI EKSEKUSI KODE UNTUK GRAF**

**SEBAGAI MEDIA BELAJAR 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**

**Desember 2017**

**PENGEMBANGAN KAKAS**

**VISUALISASI EKSEKUSI KODE UNTUK GRAF**

**SEBAGAI MEDIA BELAJAR PEMROGRAMAN**

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

Tanggal \_\_ Januari 2018

|  |  |
| --- | --- |
| 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, batasan implementasi, metodologi penelitian, dan sistematika penulisan laporan tesis.

## I.1 Latar Belakang

Visualisasi adalah proses mengubah data menjadi bentuk visual yang memungkinkan pengguna untuk mengamati data tersebut (Diehl, 2007). Fitur yang tersembunyi di dalam sebuah data memungkinkan untuk ditampilkan secara visual. Namun demikian, eksplorasi dan analisis data masih tetap harus dilakukan terhadap visualisasi tersebut sehingga dapat diperoleh informasi yang diinginkan.

Saat ini teknologi internet dan web memberikan kemudahan akses untuk berbagi informasi (Bonk, 2009). Aplikasi web terus berkembang dari sekadar penyampaian informasi kontekstual statis, sehingga 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).

Perkembangan kakas visualisasi program (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 kakas pada *browser*. Salah satu kakas VP berbasis web yang masih aktif adalah *Online Python Tutor* (OPT), yang dapat langsung diakses di alamat *http://www.pythontutor.com*.

OPT memanfaatkan pustaka D3JS untuk melakukan visualisasi. D3JS adalah sebuah pustaka *JavaScript* bersumber terbuka sebagai “*kernel*” visualisasi untuk mengurangi ketergantungan *plug-in* pada *browser* (Bostock dkk., 2011). Namun saat tesis ini ditulis, OPT hanya dapat memvisualkan data primitif berupa tabel dan tumpukan data (*stack*)(Guo, 2013). Untuk data yang lebih kompleks seperti graf, OPT belum bisa merepresentasikannya sesuai persepsi visual graf. Permasalahannya adalah tidak setiap kode memiliki alur proses operasi pembentukan graf, sehingga membutuhkan fitur khusus untuk dilakukan visualisasi secara otomatis.

Pada penelitian tesis ini, pengembangan kakas dilakukan 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. Hasil pengembangan kakas ini diharapkan dapat memudahkan pengguna untuk mempelajari kode yang terdapat graf didalamnya.

Alasan lain pengembangan kakas ini adalah karena telah berkembang pesat model belajar secara *online* atau yang saat ini disebut MOOC (*Massive Open Online Course*). Para pengguna internet dapat belajar dari jarak jauh melalui portal web yang disediakan oleh lembaga atau komunitas tertentu. Adanya kakas ini diharapkan dapat membantu sebagai media belajar pemrograman jarak jauh.

## I.2 Rumusan Masalah

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

1. Apa strategi pengembangan kakas untuk mendeteksi adanya graf dalam kode sumber sehingga dapat divisualisasikan ?
2. Desain interaksi apa yang sesuai untuk visualisasi graf sehingga dapat mendukung sebagai media belajar pemrograman ?

## I.3 Tujuan Penelitian

Tujuan dari penelitian tesis ini adalah untuk menghasilkan modifikasi kakas OPT yang dapat melakukan visualisasi graf dari eksekusi kode.

## I.4 Batasan Implementasi

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++ dengan maksimal 500 baris kode;
3. Visualisasi graf dilakukan dengan maksimal seratus simpul;
4. 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 yang 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 Model Visualisasi

Tahap ini dilakukan kajian terhadap masalah yang akan diteliti kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Selanjutnya, dirumuskan suatu model untuk mendeteksi graf pada kode menjadi sebuah model animasi graf.

1. Pengembangan Kakas

Pengembangan kakas dilakukan dengan beberapa tahap, yaitu (a) eksplorasi OPT dilakukan dengan meninjau proses kerja sistem; (b) menganalisis dan merancang diagram alur sistem; (c) merumuskan hipotesis solusi dan tahapan visualisasi data; dan (d) implementasi dan perbaikan kakas.

1. Pengujian dan Evaluasi Kakas

Hasil pengembangan kakas perlu diuji dengan memasukkan berbagai jenis kode 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. Selain itu, eksplorasi perkembangan kakas VP dilakukan untuk menganalisis masalah dan merumuskan hipotesis atau solusi yang mungkin.

1. Bab III Analisis Masalah dan Model Visualisasi

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.

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.

# Bab II Tinjauan Pustaka dan Eksplorasi

Pada bab ini dijelaskan mengenai terminologi visualisasi, proses konstruksi visualisasi data, 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*) adalah salah satu 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 atau 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. Dengan kata lain, visualisasi perangkat lunak tidak terlalu banyak fokus terhadap proses konstruksi program, akan tetapi lebih kepada analisis program dan proses pengembangan perangkat lunak.

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 terhadap algoritma, konsep dan langkah kerja suatu perangkat lunak, sedangkan VP berkaitan dengan cara kerja eksekusi kode program dan proses struktur datanya.

## II.2 Metodologi dan Prinsip Visualisasi Data

Kompleksitas data tidak akan bermakna menjadi sebuah informasi yang bermanfaat jika memahaminya dengan cara yang kurang tepat. Proses visualisasi dilakukan dengan mengombinasikan berbagai bidang disiplin ilmu, seperti statistik, *data mining*, desain grafis, dan visualisasi informasi. Desain grafis tidak menangani jutaan data, akan tetapi *data mining* yang memiliki kapasitas itu. Visualisasi perangkat lunak membuat interaksi dan berbagai jenis representasi data abstrak, namun metode yang digunakan untuk visual dinilai kurang aestetik karena tidak adanya keterlibatan dari desain grafis. Oleh karena itu, ketiga bidang disiplin ilmu tersebut harus saling melengkapi menjadi sebuah proses yang dapat mengomunikasikan data lebih efektif untuk direpresentasikan (Fry, 2008).

Untuk dapat melakukan visualisasi data, dibutuhkan beberapa tahapan yang merupakan kombinasi dari ketiga bidang disiplin ilmu tersebut. Meskipun ketiga bidang itu memiliki prinsipnya sendiri, tetap harus disesuaikan setiap bagian menjadi kesatuan proses. Prosesnya dimulai dengan memahami seperangkat data mentah dan mengajukan sebuah pertanyaan yang berkaitan dengan data tersebut. Secara rinci ada tujuh tahapan proses konstruksi visualisasi data (Fry, 2008):

1. *Acquire* (sumber data); cara perolehan sebuah data, apakah dari sebuah berkas di dalam *harddisk* komputer atau bersumber dari jaringan internet.
2. *Parse* (pengelompokan); mengurai struktur data menjadi makna tertentu dan mengategorikannya.
3. *Filter* (menyaring); menghapus data yang tidak penting sehingga tersisa data yang menjadi fitur.
4. *Mine* (menggali informasi); terapkan metode dari ilmu statistik atau *data mining* sebagai cara untuk memahami data sesuai konteksnya.
5. *Represent* (merepresentasikan); pilih visual model dasar seperti diagram batang, diagram garis, atau graf.
6. *Refine* (perbaikan visual); lakukan perbaikan terhadap representasi visual dasar untuk membuat lebih jelas dan menarik.
7. *Interact* (interaksi); tambahkan interaksi untuk memanipulasi data atau sebuah kontrol terhadap fitur data yang disajikan.

Tahapan tersebut tidak harus diikuti semua. Pada visualisasi tertentu hanya empat dari tujuh tahapan yang bisa diterapkan, namun pada visualisasi data lain mungkin semua dari tahapan tersebut harus diterapkan. Pada Gambar II.1 menunjukkan tahapan secara berurutan dan keputusan akhir berdasarkan tahapan yang telah dilakukan sebelumnya. Setiap tahap pada proses ini saling terhubung karena saling memengaruhi satu dengan yang lain.



1. Keterhubungan antar tujuh tahapan (Fry, 2008)

Keterhubungan antar setiap tahapan dalam proses tersebut menggambarkan sangat pentingnya kerja sama antar individu atau kerja tim dalam kesatuan proyek. Biasanya seorang *programer* menangani bagian teknis, seperti cara perolehan dan pengolahan data, dan *desainer* fokus terhadap pemilihan warna dan tampilan antarmuka. Ketika proses pengambilan data, biasanya ditentukan kapan data itu harus tersedia, setiap waktu atau sekali dalam sebulan. Hal ini juga menentukan desain grafis yang cocok untuk masalah ketersediaan data berikutnya.

Ada tiga prinsip utama untuk membuat visualisasi data (Fry, 2008), yaitu :

1. Sadari bahwa setiap proyek visualisasi memiliki spesifikasi kebutuhan yang unik, karena tidak setiap data dapat direpresentasikan ke bentuk visual yang sama;
2. Hindari visual data yang terlalu rumit atau kurangi rincian data yang dapat mengganggu fitur data tersebut, sehingga pengguna tidak dapat melihat apa yang menjadi fokus dalam visualisasi tersebut; dan
3. Pahami siapa konsumen atau pengguna dari visualisasi tersebut. Visualisasi yang ditampilkan menggunakan komputer meja dibandingkan dengan gawai, tentu berbeda dalam hal interaksi visual dan tampilan kerumitan datanya.

## II.3 Teori Desain Interaksi

Desain interaksi secara khusus memiliki makna untuk menciptakan pengalaman pengguna (*user experience* / UX) dalam meningkatkan fungsi suatu produk sehingga dapat saling berkomunikasi dan berinteraksi dengan manusia. Winograd (Preece, 2002) mendeskripsikan bahwa desain interaksi adalah ruang untuk berkomunikasi dan berinteraksi dengan manusia yang bertujuan untuk mendukung aktivitas manusia. Komunikasi dan interaksi merupakan pencerminan dari aksi dan reaksi terhadap sesuatu. Media interaktif dikatakan efektif dan aestetik ketika dapat memberikan reaksi terhadap setiap aksi yang dilakukan oleh pengguna sehingga tercipta komunikasi.

Untuk dapat memahami konsep interaksi, dapat melihat pada Gambar II.2. Pada gambar tersebut, ada tiga esensi utama untuk bisa dikatakan sebagai sebuah interaksi pada desain produk (baik itu perangkat lunak atau keras), yaitu *designer*, pengguna, dan sistem. Secara tidak langsung, ketiga komponen ini saling berkaitan membentuk konseptual model (Fry, 2008) sebagai berikut:

* Model desain—model yang dibuat oleh *designer* bagaimana sistem itu seharusnya bekerja;
* Sistem—bagaimana sistem itu sebenarnya bekerja;
* Model pengguna—bagaimana pengguna dapat memahami sistem itu bekerja.



1. ­­Konseptual model komponen interaksi (Fry, 2008)

Idealnya dalam dunia nyata, ketiga komponen tersebut saling memetakan dengan komponen lainnya. Pengguna harus dapat melakukan aktivitas atau tugasnya sesuai yang didesain untuk dapat berinteraksi dengan sistem. Jika pengguna tidak dapat memahami model desain, maka tidak akan dapat memahami kerja sistem sehingga sistem akan berjalan dengan tidak efektif dan muncul kesalahan sistem (*error*).

Aspek kognisi (Fry, 2008) adalah salah satu kemampuan yang dimiliki oleh pengguna seperti berpikir, memperhatikan, mempelajari, mengingat, persepsi, membuat keputusan, merencanakan, membaca, berbicara, dan mendengar. Aspek ini sangat penting dan memiliki relevansi terhadap model desain dan proses interaksi. Sebuah tampilan antarmuka dapat dikatakan baik jika pengguna dapat mempersepsikan, mempelajari, dan mengingatnya untuk menyelesaikan tugas atau aktivitasnya. Kognisi memiliki tiga perspektif atau pendekatan (Fry, 2008), diantaranya:

* Mental model—didefinisikan sebagai pemahaman dan pengalaman dasar yang dimiliki pengguna terhadap kebiasaan menggunakan suatu produk;
* Memroses informasi—pendekatan ini memberikan informasi yang mudah dipersepsikan bagi pengguna untuk menggunakan produk;
* Kognisi eksternal—merupakan proses kognisi untuk berinteraksi dengan berbagai bentuk representasi atau media eksternal dalam membantu pengguna untuk mendukung aktivitasnya.

Secara esensi proses pengembangan desain interaksi memiliki empat tahap utama (Preece, 2002), yaitu:

1. Mengidentifikasi kebutuhan dan menetapkan persyaratan khusus;
2. Mengembangkan alternatif desain yang sesuai dengan persyaratan tersebut;
3. Membangun purwarupa versi desain interaktif sehingga dapat saling berkomunikasi dan dinilai kelayakannya;
4. Mengevaluasi apa yang sedang dibangun selama proses pengembangan desain berlangsung.

Keempat tahap tersebut saling berkaitan dan harus selalu diulang untuk memastikan desain yang dibangun telah memenuhi syarat dan kebutuhan. Sebagai contoh, kegunaan (*usability*) suatu produk yang telah dibangun dapat diukur dengan memperhatikan setiap perbaikan yang telah dilakukan atau dengan melihat persyaratan tertentu yang belum terpenuhi.

Pengguna mengharapkan sebuah sistem dan produk yang mudah untuk dipelajari dan digunakan dengan efektif, efisien, aman, dan memuaskan. Tampilan yang menarik dan atraktif juga merupakan esensi dari beberapa produk. Namun, semua itu tidak dapat secara langsung terwujud, akan tetapi diperlukan evaluasi secara terus-menerus dan berulang melalui pengujian dan survei kepada pengguna. Terdapat tiga prinsip yang direkomendasikan dalam hal ini (Fry, 2008), yaitu (1) fokus terhadap pengguna dan aktivitasnya terhadap produk tersebut; (2) amati, ukur, dan analisis interaksi pengguna dengan produk atau sistem; dan (3) lakukan desain secara berulang dengan versi berbeda. Evaluasi ini dibutuhkan untuk memastikan pengguna dapat menggunakan produk atau sistem dengan nyaman.

## II.4 Properti Visual Graf

Graf memiliki berbagai macam algoritma dan model visual, sehingga perlu dirinci 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 ini.

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 adalah bagian dari jenis graf terhubung yang tidak mengandung sirkuit (Knuth, 1997). Teori pohon (*tree*) adalah salah satu konsep penting, karena memiliki banyak manfaat di bidang ilmu komputer dan informatika. Konsep ini telah banyak dikaji secara intensif sebagai objek matematika terapan yang banyak terkait 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. Daftar nama algoritma dan model visual pohon ditampilkan pada Tabel II.2 berikut ini.

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

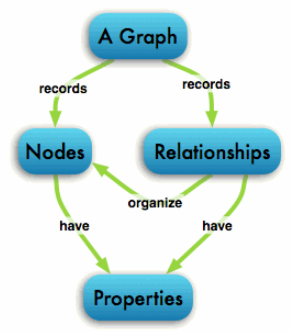
| **No.** | **Nama Algoritma Pohon** | **Model Visual** |
| --- | --- | --- |
| 1 | *Binary Search Tree* (BST) |  |
| 2 | *Digital Search Tree* |  |
| 3 | *Radix Tree* | https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/An_example_of_how_to_find_a_string_in_a_Patricia_trie.png/220px-An_example_of_how_to_find_a_string_in_a_Patricia_trie.png |
| 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.3 (a) dan model visual graf pada Gambar II.3 (b).

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

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

Pada Gambar II.4 menggambarkan diagram properti graf yang saling melengkapi. Sebuah graf merekam simpul dan relasi yang saling mengatur koneksi antar simpul. Setiap simpul dan relasi memiliki properti masing-masing, seperti deskripsi simpul, arah relasi, dan bobot antar simpul jika memilikinya.



1. Diagram properti graf[[1]](#footnote-1)

## 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, 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 dekade tahun terakhir 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++.

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. Kekurangan kakas ini adalah membutuhkan konfigurasi lebih lanjut terhadap *plug-in Java* di *browser* pengguna. Hal ini sangat berbeda dengan kakas OPT yang dapat langsung dikunjungi (*pythontutor.com*) dan digunakan secara langsung dari *browser* pengguna tanpa perlu memasang *plug-in* tambahan.

## II.6 Eksplorasi Kakas *Online Python Tutor*

*Online Python Tutor* atau disingkat OPT (Guo, 2013) adalah kakas VP yang bersifat bebas dan bersumber terbuka. Kakas ini telah memiliki banyak fitur yang mendukung berbagai macam bahasa pemrograman, seperti *Python, Java, C, C++, Ruby, JavaScript, TypeScript* dan masih terus dikembangkan[[2]](#footnote-2). 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.5 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, maka tampilan berubah seperti pada Gambar II.6. Fungsi ini menggunakan kode pustaka dari *TogetherJS*[[3]](#footnote-3) 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.5 adalah untuk memasukkan kode program. Pengguna dapat mulai belajar bahasa pemrograman dengan mengetikkan kode di *form editor* tersebut. Selanjutnya dapat menekan tombol pada nomor 3 untuk melihat simulasi dan visualisasi dari kode program yang telah dibuatnya, seperti pada Gambar II.7 berikut ini.



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.8 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 kode C dan C++

Untuk lebih jelas tahapan proses pada Gambar II.8 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*.
6. Ketika pengguna menekan tombol “*Forward*” atau “*Back*” (lihat Gambar II.7) proses visualisasi akan membaca poin indeks yang berada pada *trace* eksekusi sesuai dengan banyak langkah yang telah terbentuk.

Kakas OPT dapat disederhanakan menjadi beberapa inti komponen dengan representasi abstrak seperti terlihat pada Gambar II.9 berikut ini.



1. Ilustrasi Sederhana Proses Kakas OPT

*OPT backend* berfungsi untuk merespon terhadap terbentuknya *trace* eksekusi program. Kemudian *trace* eksekusi diubah menjadi standar format tertentu yang dipaketkan berbentuk *JSON*. *OPT frontend* (antarmuka *browser*) menerjemahkan *JSON* menjadi visualisasi yang dikombinasikan dengan pustaka *D3JS*. Maka visualisasi dapat tampil kepada pengguna seperti pada Gambar II.7.

### II.6.2 *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 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.6.3 *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.10 berikut ini.



1. Ilustrasi Format *Trace* Eksekusi Program

Bagan pada Gambar II.10 di atas mengilustrasikan posisi eksekusi kode pada saat masih kosong (*empty state*). *Trace* eksekusi program disimpan dalam bentuk JSON yang didefinisikan dalam dua bagian besar, yaitu:

1. “***code***” berbentuk string, yang berisi kode program dari pengguna.
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 dapat berisi: *user\_call*, *user\_return*, *user\_exception* atau *user\_line* |

### II.6.4 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.11 berikut ini.



1. Peran Utama *D3JS* 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.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi

Studi literatur yang telah dilakukan memberikan pemahaman mendasar mengenai terminologi visualisasi dan kegunaannya, metodologi dan prinsip visualisasi data, teori desain interaksi, serta kakas VP yang telah berkembang hingga saat ini. Kakas VP dibuat sebagai alat peraga atau suplemen untuk membantu proses pembelajaran pemrograman. Adanya kakas VP tidak menjamin pengguna dapat langsung mengerti, karena masih harus melalui proses konstruksi dan analisis terhadap visualisasi tersebut.

Dari hasil eksplorasi kakas VP, diperoleh bahwa tidak ada satu pun kakas VP berbasis web dengan bahasa C atau C++ yang dapat menampilkan visual graf. Sehingga ini menjadi peluang untuk penelitian lebih lanjut dalam pengembangannya. Terutama penggunaan *Typescript* sebagai dasar pengembangan kakas yang masih baru dan memiliki prospek ke depannya dalam teknologi web terkini.

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[[4]](#footnote-4). 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.

# Bab III Analisis Masalah dan Model Visualisasi

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

## III.1 Analisis Masalah

Langkah awal sebelum pengembangan adalah eksplorasi kakas OPT yang 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.

Permasalahan utama pada penelitian ini adalah menemukan data graf pada kode untuk dapat dilakukan visualisasi sebagaimana persepsi visual graf itu beroperasi. Perhatikan pada Gambar III.1 terlihat bahwa untuk merepresentasikan graf (salah satu contoh kode graf *Breadth First Search*) masih dilakukan dengan metode primitif, yaitu berupa tabel data larik atau matriks. Selain itu, pada Gambar III.2 yang menggunakan *pointer* sebagai abstraksi untuk relasi antar simpul, tidak membuat menjadi lebih jelas relasinya karena saling tumpang tindih visual yang ditampilkan. Tentu ini membuat pengguna untuk menelusuri kode dengan kecocokan visualnya menjadi lebih rumit.

Perubahan visual antar setiap langkah pada baris kode kurang interaktif, karena setiap ada perubahan data dari langkah sebelum dan sesudahnya tidak ada perubahan warna atau penanda khusus sebagai bentuk atensi kepada pengguna. Adanya proses ini akan memudahkan proses kognisi pengguna dalam menangkap informasi yang tampil pada visual tersebut.



1. Visual primitif berupa tabel matrik



1. Visual primitif berupa *struct* dan *pointer*

Data *JSON execution trace* adalah sumber data yang digunakan untuk visualisasi. Salah satu contoh data *JSON* berupa matriks yang dihasilkan dari Gambar III.1 dapat dilihat pada Tabel III.1 berikut ini. Data ini adalah contoh variabel *‘a’* berupa matriks yang memiliki dimensi 7 x 7. Nilai berupa "C\_MULTIDIMENSIONAL\_ARRAY" adalah jenis data *n-array*; nilai baris kedua "0x601100" adalah posisi alamat memori yang digunakan; kemudian ukuran dimensi yang digunakan berada di posisi ketiga [7, 7]. Sisa baris berikutnya adalah isi data setiap elemen di dalam variabel matriks tersebut.

1. Variabel *‘a’* berupa matriks berdimensi 7 x 7

|  |
| --- |
| "a": [  "C\_MULTIDIMENSIONAL\_ARRAY",  "0x601100", [  7,  7  ],  [  "C\_DATA",  "0x601100",  "int",  0  ],  [  "C\_DATA",  "0x601104",  "int",  0  ],  [...]  ] |

Sedangkan untuk data *JSON* dari Gambar III.2 dapat dilihat pada Gambar III.3. Objek *heap* ini berisi alamat memori yang digunakan oleh *pointer* dan *struct*. Pencarian pola visualisasi akan dijelaskan lebih lanjut pada subbab III.2. Untuk memudahkan dalam proses pengolahan data *JSON,* maka digunakan bantuan kakas web yang dapat diakses melalui alamat http://www.json2table.com. Kakas ini membantu mengolah dan mencari pola visualisasi dalam data *JSON* dengan mengonversi ke bentuk tabel dan daftar poin-poin.



1. Contoh data *JSON pointer* pada *heap*

## III.2 Hipotesis Solusi Penyelesaian Masalah

Pada Tabel III.1 dan Gambar III.3 menunjukkan hasil data *JSON trace* yang akan dilakukan proses visualisasi data sesuai pada subbab II.2 tentang tujuh tahapan proses konstruksi visualisasi data. Dari data *JSON* dapat dibagi menjadi dua kategori utama untuk representasi graf dalam kode, yaitu (1) bersumber dari tabel informasi berupa matriks atau *n-*dimensi *array*, dan (2) berupa *struct* yang berisi *pointer*.

1. Isi alamat memori pada objek *heap*

|  |
| --- |
| "0x5402210": [  "C\_ARRAY",  "0x5402210", [  "C\_STRUCT",  "0x5402210",  "node", [  "vertex", [  "C\_DATA",  "0x5402210",  "int",  2  ]  ],  [  "next", [  "C\_DATA",  "0x5402218",  "pointer",  "0x5402170"  ]  ]  ]  ] |

Untuk dapat melakukan visualisasi secara otomatis, diperlukan fitur tertentu dari data-data tersebut. Ada beberapa fitur yang bisa menjadi acuan, yaitu :

* "C\_MULTIDIMENSIONAL\_ARRAY" digunakan untuk menemukan variabel data matrik yang menyimpan nilai-nilai simpul dan relasinya serta bobot suatu graf (akan dibahas lebih lanjut pada Bab IV);
* "C\_STRUCT" digunakan untuk menemukan variable *struct* yang masih harus dilanjutkan dengan pengecekan fitur ketiga berikut;
* "pointer" di dalam "C\_DATA" yang mengindikasikan adanya relasi antar simpul dengan menggunakan *pointer* di dalam *struct*.

Fitur-fitur tersebut merupakan kunci utama, selain harus dicari pola visual lain seperti variabel pendukung lain yang sedang digunakan dalam kode program.

Visualisasi graf memiliki beberapa properti yang dibutuhkan untuk menampilkan hasil pengolahan data dari *JSON trace*. Pada Tabel III.3 menunjukkan berbagai jenis properti visual graf yang mungkin dapat direpresentasikan menjadi visualisasi. Dari daftar itu tidak semua properti visual digunakan, harus dapat menyesuaikan data yang sedang diproses dalam sebuah kode yang sedang dieksekusi.

1. Properti visual graf untuk visualisasi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Graf** | **Pohon** | **Nama Properti** | **Visualisasi** |
| 1 | √ | √ | Label Simpul (Vertex) | x  atau |
| 2 | √ | √ | Bentuk Simpul (Vertex) | atau atau |
| 3 | √ | √ | Bentuk Sisi (Edge) | atau atau |
| 4 | √ | X | Label Sisi | x  x  atau |
| 5 | √ | √ | Warna (Vertex & Edge) | atau |
| 6 | √ | √ | Ketebalan Garis (Vertex & Edge) | atau |

## III.3 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.3.1 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 berorientasi objek.

Rincian kakas dan *library* pendukung untuk pengembangan kakas OPT yang telah terintegrasi dapat dilihat pada Tabel III.4 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 | 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.3.2 Kebutuhan Perangkat Keras

OPT dapat berjalan dengan baik pada komputer dengan spesifikasi yang direkomendasikan sebagai berikut:

* *Processor core i3*
* *RAM 2GB*
* Sistem operasi Ubuntu 16.04.

Kebutuhan perangkat itu untuk proses pengembangan kakas, namun untuk produksi versi publik yang dipasang pada komputer server, dibutuhkan spesifikasi RAM sekitar 4GB – 8GB. Hal ini karena sistem *sandbox docker engine* membutuhkan ruang memori lebih jika memroses sekitar 500 baris kode untuk dapat menghasilkan data *trace* eksekusi *JSON*.

# Bab IV Pengembangan Kakas

Bab ini menjelaskan proses pengembangan kakas yang terdiri dari proses visualisasi data, perbaikan diagram kelas dan implementasi kakas. 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 Metode Konstruksi Visualisasi Data

Tujuh tahap berikut menjelaskan metode konstruksi visualisasi dari sumber perolehan data hingga perancangan desain interaksi. Tahapan ini telah dibahas pada subbab II.2 tentang metodologi dan prinsip visualisasi data. Sebelum mengembangkan kakas, diperlukan ketujuh tahapan ini untuk menganalisis data sehingga dapat dibentuk visualisasi graf yang interaktif.

1. *Acquire* (sumber data)

Tahap ini memastikan perolehan data yang dapat diproses lebih lanjut. Sumber data berupa *JSON* yang diperoleh dari proses *ajax*. Pada ... adalah contoh data eksekusi *trace JSON*. Data ini tersedia ketika ada permintaan dari pengguna melalui browser. Jadi, sumber data yang digunakan bersifat dinamis dan tergantung dari kode sumber yang sedang dieksekusi oleh pengguna.

<gambar--contoh data JSON>

1. *Parse* (pengelompokan)

Data yang digunakan dibagi menjadi dua bagian besar, yaitu 2-dimensi *array* (berupa matriks) dan *pointer* (biasanya dalam bentuk *struct* atau *list*). Kedua tipe data ini yang digunakan sebagai dasar pembentukan visualisasi graf.

1. Data eksekusi *trace JSON* terdapat 2-dimensi *array* (matriks) atau pointer.
2. Variabel global atau lokal
3. Berbobot atau tidak
4. Berarah atau tak-berarah

Daftar label untuk membantu pengelompokan data dapat dilihat pada ... berikut ini:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Matriks** | **Pointer** | **Global** | **Berbobot** | **Berarah** |
| Ya | ✓ | - | ✓ | - | - |
| Tidak | - | ✓ | - | ✓ | ✓ |

Hasil dari tabel tersebut adalah data berupa matriks dengan variabel global, tak berbobot, dan tak berarah. Penjelasan masing-masing label pada kolom tersebut adalah sebagai berikut:

1. Matriks adalah array berdimensi dua yang ditandai dengan atribut “C\_MULTIDIMENSIONAL\_ARRAY” di dalam data *JSON*.
2. *Pointer* adalah label yang digunakan untuk menandai adanya variabel yang didefinisikan sebagai *pointer* di dalam kode sumber. Label ini digunakan ketika di dalam data *JSON* terdapat atribut ‘*pointer*’.
3. Label ‘*Global’* adalah variabel yang dideklarasikan di luar fungsi main() atau fungsi lain, sedangkan variabel *private* dideklarasikan di dalam fungsi main() atau fungsi lain. Variabel *private* ditandai dengan nilai ‘tidak’ pada label ‘*global’* ini.
4. Label ‘berbobot’ digunakan ketika terdapat nilai rentang antara 0 – 999. Label ini digunakan untuk bobot edge atau ...
5. Label ‘berarah’ digunakan ...
6. *Filter* (proses memilah data)

Proses ini mengambil fitur data yang dibutuhkan dan mengabaikan data lain.

Karena dalam JSON tersebut memiliki banyak jenis data yang tidak berkaitan dengan terbentuknya visual graf. Pada ... adalah satu contoh data JSON yang berisi

1. *Mine* (menggali informasi)

Data yang diproses selanjutnya dicari informasi yang dapat menunjukkan adanya graf dalam kode sumber.

1. *Represent* (representasi)

Pada bagian ini proses representasi data dimulai dengan perancangan visualisasi. Pustaka D3JS membantu dalam proses visualisasi data. Format dasar data yang dapat digunakan oleh D3JS untuk visual graf adalah sebagai berikut:

<format dasar data graf>

1. *Refine* (perbaikan visual)

Bagian ini merupakan perbaikan desain sehingga visualisasi memiliki makna yang sesuai persepsi.

1. *Interact* (interaksi)

Menambahkan fitur interaksi untuk memudahkan pengguna dalam mengeksplorasi visualisasi data.

1. Proses Visualisasi Data

## IV.2 Perbaikan Struktur Kelas

Saat memulai eksplorasi ditemukan struktur berkas dan kelas yang sulit untuk dibaca alur sistemnya. Berkas dokumentasi untuk pengembangan dari peneliti sebelumnya juga tidak lengkap, sehingga mempersulit untuk proses membaca kode sumber. Walaupun kode sumber menggunakan bahasa *Typescript* yang bersifat pemrograman berorientasi objek.

Modular memudahkan pengembang untuk manajemen fitur

Alur sistem yang terstruktur membantu proses analisis kebutuhan sistem



Pada ... tersebut terdapat kelas GraphViz (kotak berwarna merah), yaitu penambahan fitur untuk visual graf yang sedang dikembangkan dalam penelitian ini. Kelas ini berisi beberapa fungsi, diantaranya model visualisasi, proses rendering

## IV.3 Implementasi Kakas

Secara garis besar, kakas OPT belum dapat memvisualisasikan graf. Pengembangan kakas ini diharapkan dapat menambah fitur untuk visualisasi graf saat pengguna mengeksekusi kode sumber yang terdapat algoritma graf. Tujuannya adalah agar membantu pengguna untuk memahami eksekusi kode 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 diagram kelas.
5. Integrasikan model visualisasi dengan OPT.

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://neo4j.com/developer/graph-database/ [↑](#footnote-ref-1)
2. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-2)
3. https://togetherjs.com/ [↑](#footnote-ref-3)
4. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-4)
5. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-5)