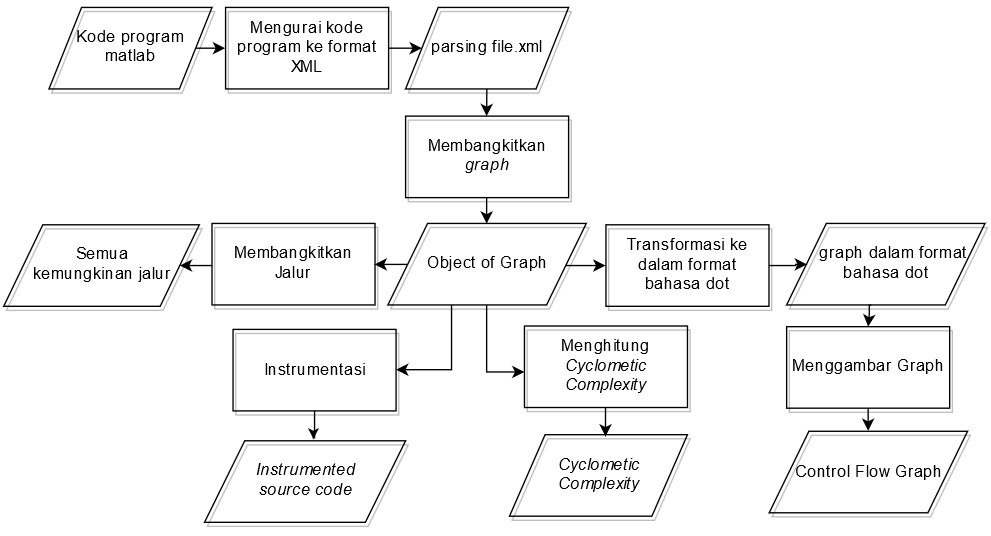
* **BLACK BOX TESTING** Memeriksa **fungsionalitas** apakah **output** sudah **sesuai** dengan yang ditentukan **tanpa mengetahui** bagaimana output tsb dapat dihasilkan
* **WHITE BOX TESTING** Memeriksa struktur internal dan alur logika (proses) sebuah perangkat lunak.
  + Dapat mengetahui cara kerja PL secara keseluruhan
  + Dapat mendeteksi jika ada kesalahan logika
* **Path Testing:** salah satu metode pengujian struktural yang **menggunakan kode program** untuk **menemukan semua jalur** yang **mungkin** dapat **dilalui** program dan dapat digunakan untuk **merancang data uji**. Metode ini **memastikan** semua kemungkinan jalur **dijalankan setidaknya satu kali**
* Untuk melakukan monitoring jalur mana yang diambil oleh sebuah masukan pada saat eksekusi program, maka diperlukan penanda yang dapat memberikan informasi cabang mana yang dilalui.
* **Instrumentasi :** proses menyisipkan sebuah penanda (tag) di awal atau di akhir setiap blok kode perintah, baik sebelum atau sesudah kondisi terpenuhi atau tidak
* ***Control Flow Graph* (CFG) :** graph berarah yang merepresentasikan aliran dari sebuah program. Terdiri dari *nodes* dan *edges*.
* **Manfaat**
  + Instrumentasi Otomatis -> Mempercepat
  + Membangkitkan jalur -> Membangkitkan data uji
  + Membangkitkan diagram CFG -> Memahami struktur dan alur dari suatu program → *re-engineering* perangkat lunak



* ***Extensible Markup Language* (XML)** : **bahasa** yang dapat **mendeskripsikan sebuah dokumen**. XML memiliki banyak bagian yang tidak memiliki struktur yang pasti.
  + XML terdiri atas dua bagian utama
    - Elemen yang dapat disebut sebagai *node* merupakan bagian penting yang dapat **menggambarkan struktur** dari XML.
    - Atribut merupakan bagian yang dapat digunakan sebegai **informasi tambahan** dari setiap elemen (Hartwell 2017).
* **Membangkitkan *Graph***:
  + Setiap elemen XML yang merupakan kontrol struktur pada program akan menjadi *node* baru.
  + Dibangkitkan dengan membaca Setiap *node* berisi informasi nomor baris dan nomor kolom yang akan digunakan untuk melakukan instrumentasi. Sehingga terbentuklah sebuah objek *graph* yang terdiri dari sekumpulan *node* dan *edge*.
* **Membangkitkan Jalur:** Jalur dibentuk dengan cara menelusuri objek *graph* yang sudah dibentuk sebelumnya.
* **Transformasi ke Dalam Format Bahasa *Dot :*** Bahasa ini dapat mendeskripsikan 3 macam objek, yaitu *graph, nodes,* dan *edges* (Ganser *et all* 2015).
* **Memvisualisasikan *Graph* dalam bentuk CFG**
* **Menghitung *Cyclometic Complexity***

*Cyclomatic complexity* merupakan suatu sistem pengukuran untuk menentukan banyaknya *independent path* dan menunjukan tingkat kompleksitas dari suatu program.

*Independent path* adalah jalur yang melintas dalam program yang sekurang-kurangnya terdapat kondisi baru.

* **Instrumentasi**
  + Instrumentasi merupakan sebuah proses menyisipkan sebuah penanda (*tag*) di awal atau di akhir setiap blok kode.
  + Penyisipan dapat dilakukan sebelum atau sesudah kondisi terpenuhi atau tidak.
  + Dalam path testing, penanda ini dapat digunakan untuk memonitor jalur yang dilalui program ketika dijalankan dengan masukan data uji tertentu (Arkeman et al. 2014).
  + Instrumentasi akan dilakukan dengan cara menambahkan dulu variabel keluaran bernama *traversedPath*.
  + Variabel ini digunakan untuk menyimpan informasi *node* mana saja yang dilalui ketika diberikan inputan dengan nilai tertentu. Lalu setiap sebelum dan sesudah *node* percabangan, dilakukan penyisipan kode program berupa perintah untuk memasukkan nilai *node* yang dilalui.
  + Sehingga ketika program tersebut dijalankan, akan menghasilkan keluaran tambahan bernama *traversedPath.*

**Implementasi**

* Setelah file dengan format bahasa dot terbentuk, CFG divisualisasikan dengan menggunakan *library Graphviz.Net.*
* *Graphviz.Net* adalah pembungkus C# untuk generator grafik *Graphviz* yang dibangun oleh Dixon (2013).
* Keluaran yang dikembalikan ketika mengeksekusi *Graphviz.Net* berbentuk *byte* dalam *array* sehingga dapat diolah kembali sesuai dengan kebutuhan.
* *Graphviz* merupakan *library* yang dapat digunakan untuk menvisualisasi jalur ke dalam bentuk *graph* berarah (Gansner 2015).

**Testing**

* **Uji validasi** dilakukan dengan cara membandingkan hasil yang ada pada penelitian sebelumnya dengan hasil yang dikeluarkan oleh aplikasi.
  + Pada penelitian sebelumnya, *graph* yang dibangun adalah *graph* yang hanya menggambarkan notasi percabangan dan tanpa penomoran *node*.
  + Agar dapat dibandingkan dengan hasil yang dikeluarkan oleh aplikasi, *graph* yang ada pada penelitian sebelumnya direpresentasikan ke dalam bentuk *adjacency list* terlebih dahulu secara manual.
* **Uji efisiensi** dilakukan dengan membandingkan waktu eksekusi yang dilaukan secara manual dengan waktu eksekusi oleh aplikasi.
  + Pengujian manual akan dilakukan dengan meminta satu atau dua orang yang sudah memiliki pengalaman dalam pemrograman sebagai sampel untuk melihat berapa lama waktu yang dibutuhkan untuk membangkitkan CFG, membangkitkan semua kemungkinan jalur, menghitung *cyclomatic complexity*, dan melakukan instrumentasi.
  + Dalam pengujian efisiensi, waktu yang diperoleh dari pengujian melalui aplikasi adalah waktu eksekusi dari aplikasi yang dijalankan di dalam local komputer tanpa menggunakan koneksi internet.

**Simpulan**

* Penelitian ini **berhasil membangun sebuah aplikasi** yang dapat digunakan untuk melakukan **instrumentasi** secara **otomatis**, membangkitkan CFG, menghitung ***cyclomatic complexity***, dan **membangkitkan** segala kemungkinan **jalur** yang dapat dilewati dari kode program matlab untuk ***path testing****.*
* Hasil penelitian menunjukkan bahwa **kecepatan eksekusi jauh lebih cepat** dibandingkan dilakukan secara manual sehingga dapat **menghemat sumber daya** dalam melakukan pengujian perangkat lunak.

**Saran**

* Penelitian selanjutnya diharapkan **dapat mengakomodir bahasa lain** selain bahasa matlab.
* **Instrumentasi dapat dilakukan dinamis sesuai dengan kondisi yang dibutuhkan.** Seperti yang dilakukan oleh Hermadi, selain menyimpan informasi jalur mana yang dilewati, instrumentasi yang dilakukan pada penelitian tersebut juga menyisipkan kode program untuk menghitung nilai *fittness*.

relasi antar class

* **Assosiasi** : hubungan antara object.
* **Agregasi** : tidak kebergantungan 1 sama lain. Jika kelas parent dihapus, kelas child akan tetap ada.



* **Komposisi** :

