**TUGAS PEMROGRAMAN BERBASIS KOMPONEN MERANGKUM ARTIKEL “KEEP IT SIMPLE”**

Nama: M. Eko Budicahyono

NIM : 2013-65-027

Prodi : Teknik Informatika S1

**MUDAHKANLAH**

Jika Alexander The Great bisa menguasai dunia pada usia 18 tahun, bisa kah kita berpikir bahwa orang dewasa bisa menguasai “the bits of complexity” yang ada di dalam komputer Sangat disayangkan bahwa “the bits of complexity” ini tidaklah mudah seperti anggapan beberapa orang. Esdger Dijkstra berkata bahwa komputer yang ada dihadapan kita ini adalah tantangan intelektual radikal yang baru yang belum pernah ada dalam sejarah.

Djikstra membantah jika pemrograman itu dibuat untuk mengimbangi batas kemampuan otak kita. Beberapa kompleksitas sebuah software dilekatkan pada masalah yang kita coba untuk pecahkan, tapi sebagian besar bergantung pada solusi sebagai masalah. Dari solusi itulah telah disadari bahwa betapa kecilnya kemampuan otak kita.

**HIRARKI DAN ABSTRAKSI**

Hiraki dan abstraksi adalah dua cara yang paling efektif untuk memanajemen kompleksitas. Hirarki itu terstruktur yang mana masalah-masalah itu dibagi menjadi beberapa level yang urut. Dalam dunia nyata hirarki ada pada modul hirarki desain yang fungsional, tapi juga ada pada hirarki pewarisan dalam desain berbasis objek, sekumpulan struktur data,dll. Orang-Orang sudah mulai terbiasa dalam menggunakan hirarki, seperti disaat kita menggambar sebuah objek yang kompleks (rumah contohnya), kita pasti menggambar garis rumahnya dulu, kemudian jendela dan pintu, dan berikut detail-detail lainnya.

Abstraksi adalah cara lain untuk mengurangi kompleksitas. Abstraksi merupakan konsep yang lebih umum dari pada hirarki seperti contoh: rumah adalah abstraksi dari kombinasi kaca, kayu, dan paku. Produktivitas pemrograman telah berkembang pesat seiring peningkatan keabstraksitas komponen-komponen program. Disini para programmer bisa lebih fokus pada pemrograman tanpa memikirkan detail-detail dari setiap bagian-bagiannya.

Kehadiran pemrograman visual telah sangat mengurangi kompleksitas dari pembuatan GUI aplikasi. Pemrograman visual membuat programmer bekerja pada level abstraksi yang mana mereka bisa fokus pada bagian aplikasi ketimbang detail-detail GUI itu sendiri.

**Desain Subsistem**

Pada level arsitektur software, kita bisa memudahkan masalah dengan membagi-baginya kedalam subsistem. Disini programmer bisa lebih fokus pada satu bagiannya, satu persatu.

**Class dan Module**

Tanpa class dan module, kita mungkin bisa mengerti setiap satu aktivitas kode programnya, namun itu nanti akan membuat sistem sulit untuk memahami terhadap semua aktivitas kode program dalam sebuah aplikasi. Class dan module, dan subsistem tentunya merupakan perantara antara satu aktivitas kode program dan seluruh system dalam sebuah aplikasi. Dengan class dan module kita dapat mempersingkat aktivitas kode programnya tapi dikombinasikan kedalam grup-grup untuk mempertahankan ledakan kompleksitasnya pada level seluruh sistem.

**Kepaduan dan perangkaian**

Kepaduan yang kuat dan perangkaian yang bebas (bisa dirangkai menjadi bermacam bentuk) merupakan cara yang tepat dalam memanajemen kompleksitas.

**Fan-Out**

Fan-out harus dibatasi (jumlah aktivitas kode program yang muncul) untuk mengurangi kompleksitas dalam sebuah program. Meskipun komputer mampu mengerjakan berapa pun besarnya fan-out, akan tetapi manusia yang sebagai pengembang software pun tidak mampu utnuk memikirkannya secara bersamaan dikarenakan terbatasnya kemampuan otak kita.

**Persembunyian Informasi**

Persembunyian Informasi adalah latihan dari desain persembunyian dan detail implementasi dibelakang aktivitas kode program yang abstrak, module, dan interface class. Dengan persembunyian informasi, (menyembunyikan detail) kita bisa mengurangi kompleksitas ketika menampilkannya dari berbagai bagian sudat pandang.

**Global Data**

Global data secara virtual dapat berinteraksi dengan bagian-bagian lain dari program sepanjang operasinya pada data yang sama. Dengan global data kita tidak perlu membuat kode program yang sama berulang-ulang kali.

**Goto**

Goto tidak secara penting mengikuti berbagai pola, kita tidak bisa memudahkan operasina dalam berbagai cara yang standar. Goto mungkin bisa fleksibel, namun dapat meningkatkan kompleksitas program dan oleh karena itu harus dihindari. Jika kita menggunakan goto untuk mengimbangi kekurangan dalam bahasa pemrograman, maka lakukanlah, jika itu bisa mengurangi kompleksitas program.

**Standarisasi Coding**

Manfaat utama dari standarisasi coding adalah untuk mengurangi kompleksitas dengan setiap baris kode yang kita tuliskan. Ketika kita menstandarisasikan setiap keputusan, kita bisa meningkatkan mental kita dalam menghadapi permasalahan pemrograman. Standarisasi sangat berguna ketika kita menghadapi masalah pembuatan dan mempertahankan keputusan yang berubah-ubah.

Jika kode program itu menggunakan struktur desain yang mudah dibaca, kita bisa dengan cepat untuk memahaminya. Perlu diketahui bahwa otak kita tidak mampu secara penuh mengcakup detail-detail dari sebuah program. Tapi, jika kita memahami bahwa betapa kecilnya otak kita, kita past mampu untuk memahami pemrograman dengan cara kita sendiri.