Awal dari Segalanya

Cita cita, peta jalan, objek sepeda, teori Darwin sampai teknologi kue lapis.

Hermawih Hasan
benocat@plasa.com

Yang akan kami jelaskan di sini adalah topik yang besar – ada buku-buku yang keseluruhannya dibuat hanya untuk membahas masalah perencanaan sistem, buku desain dengan orientasi objek, buku programming dengan orientasi objek, buku untuk "Many Tier Architecture ". Dari topik-topik itu saja penulis menemukan berpuluh-puluh buku. Dan pembaca jangan mengharapkan penulis untuk menyinggung secara detil tentang topik-topik tersebut secara penuh. Apa yang menjadi sasaran dalam membuat artikel ini adalah bukan untuk mengajarkan anda dasar-dasar objek dan program tetapi lebih kepada konsep/ilustrasi . Topik ini terutama sekali ditujukan bagi bahasa *VBA* atau *VB*, tetapi bisa juga untuk bahasa yang lain karena topik yang kami sebutkan di sini adalah topik yang umum.

Artikel ini akan menjadi peta bagi pengembang tingkat pemula atau menengah. Kami sebut peta karena nantinya anda sendiri lah yang harus mencari sendiri untuk menemukan jawaban-jawaban atas masalah-masalah. Bila seorang pengemudi tidak mengetahui jalan maka solusinya adalah dengan membaca peta atau menanyakan kepada orang di jalan. Demikian juga anda, bila anda tidak mengetahui pemecahan atas suatu permasalahan maka anda membaca artikel, buku, kursus, sekolah dan lain sebagainya.

Dengan adanya artikel kecil ini, semoga anda menjadi termotivasi untuk mencari lebih jauh lagi tentang topik yang dibahas ini dari pustaka yang diberikan.

Pentingnya Sebuah Tujuan

Tunjukkan petanya dan kami akan sampai ke tempat tujuan.

Napoleon Hill dalam bukunya *Law of Success* menekankan pentingnya sebuah tujuan utama dari kehidupan kita. Dalam kata-katanya ia memberikan nama dari hukum tersebut sebagai *A Definite Chief of Aim*. Dalam risetnya dia menemukan bahwa orang-orang sukses ternyata mempunyai kesamaan dalam hal mereka itu telah mempunyai tujuan utama yang pasti. Tujuan itu akan mendorong pikiran, tenaga dan semua sumber daya manusia untuk merealisasikannya. Penulis tidak bisa membantah sedikitpun betapa benarnya dia.

Covey dalam bukunya *7 Habits Of Highly Effective People* juga menekankan pentingnya tujuan ini dengan kata-kata yang lain yaitu kita harus mempunyai peta perjalanan hidup kita. Kerja keras tidak menjamin keberhasilan katanya, karena bila arah tujuannya salah maka semakin keras kita berusaha semakin kita menjauh dari apa yang sebenarnya kita inginkan. Kata-katanya tersebut dapat diilustrasikan dengan orang yang ingin pergi ke sebuah kota tanpa peta jalan yang benar. Semakin pengemudi itu mengemudikan mobilnya dengan cepat semakin jauh pula kota tujuan yang ingin dicapai.

Microsoft® juga menekankan pentingnya sebuah tujuan dengan motonya *Where do you want to go today?* Apa yang kita kerjakan sehari-hari adalah ke mana kita hendak menjadi.
Apa hubunganya semua petikan di atas dengan pengembangan sebuah aplikasi? Hubungannya sangat erat, karena orang yang ingin mengembangkan sebuah aplikasi harus mempunyai tujuan yang ingin dicapai yang diterjemahkan dalam sebuah perencanaan. Cita cita atau *Definite Chief of Aim* untuk seorang anak adalah sebuah berkat yang orang tuanya pun hanya mempunyai sedikit kontrol. Yang dimaksud di sini sebagai cita-cita atau *Definite Chief of Aim* adalah bukan keinginan yang bisa berubah-ubah seperti halnya anak kecil kebanyakan. Ketika ditanyakan ingin menjadi apa kalau besar, hari ini menjawab ingin menjadi dokter, besok ditanya ingin menjadi pilot dan besoknya ingin menjadi penyanyi. Itu adalah obrolan kosong dari anak kecil. Yang dimaksud di sini adalah keinginan utama yang akan dikejar dengan segala daya dan pengorbanan baik uang maupun tenaga. Jika untuk keinginan anak kecil di atas orang tuanya pun mempunyai sedikit kontrol tidak demikian halnya dengan tujuan dari aplikasi yang ingin dibuat. Jadi kalau anda ditanya tujuan aplikasi anda dan anda tidak bisa menjawabnya dengan benar maka jangan pernah anda mengatakan bahwa anda tidak diberikan berkat untuk itu. Yang penulis maksud dengan tujuan dari aplikasi di sini juga bukan keinginan secara lisan dari pembuat sistem yang bisa berubah-ubah setiap saat seperti halnya anak kecil ketika ditanya ingin menjadi apa. Namun tujuan dari aplikasi yang harus dimiliki adalah perencanaan aplikasi dalam bentuk tulisan yang mengikuti format-format yang baik dalam penulisan sebuah perencanaan. Mungkin pada saat penulisannya anda memang akan mengubah isinya setiap hari atau bahkan sampai beberapa bulan. Namun yang jelas, tujuan itu dapat dijadikan sebagai gambaran nyata yang secara jelas dapat memberikan proyeksi fisik apa yang akan kita buat nantinya.

Sayangnya dalam kenyataannya proses membuat perencanaan ini tidaklah mudah dan sesederhana seperti mengatakannya. Walaupun anda banyak membaca buku tentang bagaimana menjadi seorang penulis yang baik, namun hanya dengan mempraktekannya selama bertahun-tahun sajalah kemampuan itu akan dapat terasah dengan baik. Demikian juga dengan membuat perencanaan, hanya dengan membaca puluhan buku tentang perencanaan tidak akan membuat anda menjadi perencana yang baik. Hanya dengan mendapatkan gelar bidang komputer di perguruan tinggi tidak akan membuat anda otomatis menjadi perencana yang baik. Pengalaman - yang baik maupun yang buruk - yang akan membimbing anda mengasah kemampuan terpendam anda.

Dengan adanya perencanaan - peta perjalanan pembuatan aplikasi - anda atau teman kerja anda lebih mempunyai kepastian apa yang akan dibuat. Dan penerapan dari perencanaan itulah yang akan menjadi hasil atau produk. Bila beberapa pekerja yang pernah menjadi tim pengembang, tidak lagi menjadi tim perusahaan anda, anda sebagai pemilik tidaklah terlalu dipusingkan dengan hal itu karena peta perjalanannya masih anda miliki.

Stan Leszynski membandingkan pembuat program tanpa rencana sebagai *itchy fingers* - si koboy gatal tangan, yang ingin segera menarik pelatuk senjatanya untuk diarahkan kepada lawannya. Si pembuat program tanpa memikirkan dengan seksama sasaran yang ingin dicapai, segera membuka program favoritnya dan langsung membuat kode-kode program.

Mudah-mudahan dengan sedikit penjelasan tentang arti pentingnya sebuah tujuan, kami telah memberikan sebuah bahan pikiran yang bisa mengubah cara anda membuat sebuah program.

Objek Oriented Programming

"*OOP is the only way to Program*" - Dan Appleman.

Mungkin bila kami memberikan pendahuluan OOP ini dengan teori atau istilah istilah *Encapsulation, Inheritance, Aggregation, Data Hiding,* anda akan segera menutup artikel ini dan berkata: "Ini bukan untuk saya". Dan bila ternyata kami salah dan anda ingin mengetahui tentang istilah-istilah tersebut maka kami akan berargumentasi bahwa artikel ini bukan tentang teori sehingga bila anda tertarik tentang itu kami akan menyarankan anda untuk membaca buku-buku dalam daftar bacaan.

Kami akan membawa perspektif berbeda tentang OOP ini. Bayangkan anda sedang berkunjung kepada sebuah pabrik mobil. Ah tidak, perakitan mobil masih terlalu rumit bagi penulis karena banyak sekali komponen yang tidak dikenal. Bagaimana kalau kita mengunjungi pabrik sepeda saja? Dipilih pabrik sepeda - bukan mobil - karena kita sekalian mungkin lebih mengenal bagian-bagian dari sepeda. Sepeda yang umum mempunyai ban, stang, pedal, jeruji ban, jok, lampu dan baud-baud. Untuk merakitnya, kita tidak membutuhkan seorang jenius atau seorang dengan pendidikan yang tinggi yang harus dilatih bertahun-tahun. Gampang? Tentu saja gampang.

Bayangkan anda hidup pada tahun 100 di mana sepeda belum ditemukan. Membutuhkan beratus-ratus tahun ketika manusia akhirnya menemukan ban dengan bentuk bulat dengan bahan karet, karena ternyata bentuk yang bulat dan bahan karet mempunyai hambatan yang lebih kecil dibandingkan dengan ban berbentuk segi delapan dengan bahan kayu. Untuk yang lupa pada pelajaran fisika ini ada baiknya anda menanyakan pada adik, atau anak anda yang sekarang sedang duduk di kelas 3 SD. Hanya bagian ban dari sepeda saja manusia membutuhkan waktu yang lama untuk menemukannya. Akhirnya manusia menemukan lampu sepeda yang menggunakan dinamo dengan bantuan medan magnet, pedal, stang, jeruji ban, jok. Itulah objek dari sepeda yang telah ditemukan oleh manusia selama ratusan tahun dihitung dari tahun 100. Bila kita telah mempunyai bagian-bagian dari sepeda tersebut, maka walaupun anda tidak pernah mendapat pelajaran perakitan sepeda di sekolah, anda mungkin hanya membutuhkan waktu beberapa jam untuk merakitnya. Tunggu sebentar . Sebelum anda merancang objek sepeda tersebut anda harus menentukan dahulu kegunaan sepeda tersebut , apakah sepeda yang digunakan untuk sirkus , sepeda untuk anak yang ingin belajar pertama kalinya , sepeda balap , sepeda untuk alat angkut , sepeda untuk ibu ibu pergi kepasar , sepeda untuk orang kaya , sepeda untuk kolektor , sepeda untuk orang tua . Dan daftarnya mungkin akan semakin panjang bila tidak distop disini . Jawaban jawaban dari pertanyaan tersebut akan menentukan bentuk , sifat , cara kerja - sepedanya bisa berjalan mundur dan atau maju, kecepatan , warna , jumlah ban , jumlah persneling , jenis bahan bakar (sepeda dengan mesin) .

Kembali kepada programming berorientasi objek . Seperti halnya dengan sepeda diatas kita juga harus memberikan sifat sifat dan pekerjaan (apa yang akan dilakukan) kepada objek objek . Inilah yang dalam pemrograman berorientasi objek disebut *property* (Sifat objek) dan *Method* (kerja). Bila objek-objek dari programming telah tersedia yang dibuat oleh seseorang yang mempunyai pengalaman dalam pengembangan dengan metoda OOP ini maka seseorang dengan pelatihan minimal dapat mengkombinasikan objek-objek itu menjadi sesuatu yang berguna. Jadi OOP ini menyembunyikan kompleksitas dari objek tersebut. Kita tidak tahu bagaimana objek tersebut dibuat tetapi kita bisa menggunakannya dengan mudah. Apakah ada di antara kita yang bukan tim pengembang ADO (*ActiveX Data Object*) yang mengetahui bagaimana cara kerja ADO di dalamnya secara detail? Mungkin tidak dan mungkin juga tidak perlu karena yang perlu kita ketahui adalah

bagaimana cara menggunakannya.

Seperti halnya dalam sejarah pembuatan sepeda, yang terpenting adalah menemukan objek-objek seperti ban, jeruji ban, stang, lampu, stang, dinamo lampu sehingga dapat membuat objek yang lebih tinggi hierarkinya yaitu sepeda. Kalau anda perhatikan objek ban juga adalah hasil dari perakitan objek-objek yang levelnya lebih rendah yaitu dari ban, jeruji ban, dan pelek ban. Demikian pula dalam OOP, yang terpenting adalah menemukan objek-objek yang akan digunakan sebagai bagian dari objek tingkat tinggi yang pada gilirannya dapat digunakan untuk membuat objek yang hierarkinya lebih tinggi lagi. Bila objek-objek tersebut telah ditemukan dan dibuat maka untuk merakitnya adalah pekerjaan yang jauh lebih sederhana dan mudah.

Yang penting untuk diingat, seperti halnya objek sepeda di atas, adalah bahwa objek-objek tersebut harus bekerja sama dalam harmoni. Bisa dibayangkan bila harmoni atau keselarasan itu tidak bisa dicapai dalam perakitan sepeda karena baud yang digunakan untuk memasang ban sepeda terlalu besar. Adalah tugas kita sebagai pengembang untuk membuat objek-objek aplikasi yang dapat saling bekerja satu sama lainnya dalam harmoni. Tidak ada contoh yang paling tepat untuk menggambarkan harmoni ini selain dengan contoh pergelaran musik, baik itu musik pop, paduan suara atau orkestra klasik. Apa jadinya bila tidak terjadi harmoni antara suara alat musik yang satu dengan suara alat musik lainnya, ataupun antara suara alat musik dengan suara penyanyinya? Ketika sedang membuat artikel ini, imajinasi penulis dibantu oleh alunan merdu dari suara Pavarotti dan Andrea Bocelli, penyanyi opera, dan mencoba menirukan alunan suara mereka, dan hasilnya orang-orang di sekeliling penulis menutup telinganya. Akibat dari ketidakselarasan suara penulis dengan suara merdu mereka hanyalah perasaan tidak menghargai suara penulis saja, yang tidaklah terlalu berbahaya. Anda bisa bayangkan pada pengemudi yang mengendarai sepeda yang baud bannya kebesaran? Pada kecepatan yang tinggi baudnya bisa lepas, bannya copot dan pengendaranya bisa terluka parah. Apa jadinya bila objek-objek sistem tidak bisa bekerja sama dalam harmoni? Kemungkinannya bervariasi dan yang terburuk adalah reputasi anda akan hilang.

Anda lihat bahwa konsep objek yang kami utarakan di sini sederhana sekali. Dan memang itulah tujuan kami mempermudah masalah yang kompleks. Seperti yang telah kami katakan, bagi anda yang ingin lebih mengetahui lebih mendalam tentang ini akan kami berikan daftar referensi yang telah membantu kami dalam memahami masalah yang cukup kompleks ini. Mudah mengatakannya tetapi sulit dalam prakteknya karena untuk menemukan objek-objek yang akan digunakan juga merupakan pekerjaan yang tidak mudah, apalagi bila ditambahkan kata objek-objek yang dapat bekerja, menyatu satu sama lain dalam harmoni. Proses menemukan objek-objek itu dinamakan OOA (*Object Oriented Analysis*) dan proses pembuatan desainnya disebuat OOD (*Object Oriented Design*). Semoga anda tidak memerlukan ratusan tahun hanya untuk menemukan objek-objek bagi aplikasi anda seperti halnya yang terjadi pada sejarah penemuan sepeda. Semoga beruntung.

Metafora

Jika anda masih mengingat pelajaran Bahasa Indonesia, tentu kita tidak lupa apa artinya metafora. Kita banyak menjumpai bahwa banyak sekali metafora dalam karya-karya dari para penemu, seperti halnya metafora burung untuk menemukan kapal terbang. Apakah yang anda ketahui tentang metafora dalam olah raga bela diri seperti kungfu - jurus ular, jurus bangau, jurus macan. Di atas pada saat membicarakan objek, bukan maksud kami membuat metafora pengembangan sebuah aplikasi sama dengan membuat sepeda. Contoh pabrik sepeda hanyalah untuk menerangkan arti dari objek. Metafora dari pengembangan aplikasi mengalami perubahan-perubahan sesuai perkembangan kompleksitas pembuatan aplikasi itu sendiri. Pada awalnya pengembangan dari aplikasi dibandingkan dengan pembuatan sebuah karangan. Pembandingan dengan pembuatan sebuah karangan karena bila kita salah dalam membuat sebuah kalimat maka kita hanya tinggal menghapusnya dan mengganti dengan kalimat yang baru. Jadi, tidak ada usaha-usaha ke arah perencanaan. Namun seiring dengan berkembangnya peralatan dari pengembangan, maka metafora seperti itu banyak mendapat kritik, yang katanya tidak sesuai dan tidak mungkin lagi digunakan untuk pengembangan-pengembangan

aplikasi yang kompleks. Kemudian kita mengenal metafora pengembangan aplikasi seperti pengembangan sebuah bangunan. Pada saat kita ingin membuat sebuah bangunan maka kita perlu merencanakan tata letak, kekuatan struktur bangunan. Sebelum memulai pembuatan bangunan maka cetak biru (*Blue Print*) dari bangunan itu harus sudah jadi terlebih dahulu.

Bank – Mikro Ekonomi Sebuah Negara

Penulis harap tidak diharuskan menerangkan apa artinya mikro dan makro ekonomi bukan? Akan membuat penulis pusing kepala bila harus memberikan detil teori ilmu ekonomi tersebut yang telah penulis lama lupakan. Kekuatan metafora! Suaranya melengking seperti geledek memecah kesunyian malam.

Penulis akan menggunakan kekuatan dari metafora untuk menerangkan database dengan sebuah bank. Apa hubungannya antara bank dengan database? Untuk saat ini mungkin hanya penulis yang tahu, tetapi sesaat lagi anda sebagai pembaca akan dapat menebak hubungannya.

Apa operasi utama dari sebuah bank? Bila tebakan anda adalah: mengumpulkan dana dari masyarakat yang berbentuk tabungan dan deposito dan menyalurkannya kepada masyarakat dalam bentuk kredit, maka tebakan anda itu benar. Perhatikan kata mengumpulkan dana yang artinya masuk ke dalam bank dan menyalurkan dana yang artinya keluar dari bank.

Demikian pula dengan sebuah database yaitu kumpulan data-data yang sejenis yang dikumpulkan dalam satu tempat. Ada empat operasi utama dari database yang disingkat dengan CRUD (*Create Read Update Delete*). Keempat operasi itu bisa juga dijadikan menjadi dua kategori operasi yaitu masuk ke dalam database *Create* dan *Update* kemudian operasi keluar yaitu *Read* dan *Delete*. Tidak terlalu salah bukan memberikan metapora database dengan sebuah bank ?

Bayangkan anda mempunyai akun di BPR (Bank Perkreditan Rakyat) baik berupa tabungan atau kredit untuk menunjang usaha anda. Untuk beberapa tahun perkembangan usaha anda, bank kecil di BPR itu masih bisa menunjang kegiatan usaha anda. Karena usaha bisnis anda berkembang dengan pesat, anda membutuhkan kredit yang lebih besar dan mobilitas yang lebih tinggi. BPR yang menjadi langganan anda tidak bisa memberikan kredit yang lebih besar karena keterbatasan modal dan peraturan yang diterapkan oleh Bank Indonesia kepada BPR. Bank itu pun tidak memberikan pelayanan yang lebih fleksibel karena bank itu hanya mempunyai satu cabang di mana usaha anda berdiri. Usaha anda mengharuskan anda mempunyai juga tabungan di kota lain. Walaupun hubungan anda dengan BPR itu sudah berjalan dengan baik, namun karena keharusan, anda akhirnya membuka akun di bank umum, katakanlah namanya Bank ABC. Kemudian perkembangan usaha anda pada akhirnya juga tidak bisa didukung oleh bank ABC itu karena ternyata anda juga membutuhkan mata uang asing dalam urusan ekspor dan impor sehingga anda harus juga menjadi nasabah dari bank devisa yang mempunyai jasa untuk itu. Dan operasi sebuah bank juga mengalami perkembangan lainnya dengan diperkenankannya bank untuk menjadi broker jual beli saham di mana masyarakat dapat membeli saham melalui sebuah bank. Operasi sebuah bank tidaklah sederhana lagi.

Demikian pula dengan database. Database yang diciptakan oleh pembuatnya mempunyai sasaran atau manfaat yang berbeda-beda antara satu sama lainnya. Ketika kita sudah mengenal database seperti Ms Access dan Ms SQL Server, kemudian kita juga diberikan ide tentang data-data yang tidak hanya disimpan dalam sebuah database atau ide tentang *Distributed Database*. Kita diperkenalkan dengan ide tentang data-data yang disimpan dalam E-mail, file text, file gambar dan lain sebagainya. Itu juga sebuah database. Kemudian bermunculan teknologi-teknologi seperti XML, XSLT yang dapat merepresentasikan data . Dalam bidang bahasa programming, Microsoft® menciptakan .NET. Mungkin ketika dia membuat moto: *Where do you want to go today?* Microsoft akan menjawab: *Distributed Computing*. Dalam bidang database, walaupun OO database masih belum menyebar luas karena belum adanya standarisasi namun bisa jadi pada masa akan datang OO database ini akan menggantikan *Relational Database*. Teori database tidaklah sederhana lagi - Betapa indahnya masa lalu .

Jatuh Cinta pada Pandangan Pertama

Apa yang bisa menjadikan seorang pemuda jatuh cinta pada pandangan pertama. Pasti ada sesuatu pada si gadis yang membuatnya tertarik. Pada pandangan pertama itulah si pemuda ingin lebih mengenal sang gadis dan memberanikan diri untuk mendekatinya. Jika ternyata pandangan pertamanya sesuai dengan keinginan lainnya tentu hubungan itu akan berlanjut. Bila tidak tentu hubungan itu tidak akan berlanjut. Perhatikan kalimat di atas, pandangan atau apa yang dilihat oleh mata. Pandangan pertama pengguna aplikasi tentu saja adalah pada *User Interface* atau *Form*. *User Interface* yang tidak menarik tidak akan mengundang pengguna. Bagi seorang pengguna aplikasi, apakah dia peduli atau mengetahui secara detil bagaimana aplikasi dibuat? Apakah aplikasi dibuat dengan *Many Tier Architecture* atau dengan *Objek Oriented Programming* tidaklah menjadi penting bagi pengguna aplikasi. Bagi mayoritas pengguna maka *User Interface* ini akan menjadi pilihan pertama. Tampilan dan kemudahan penggunaannya adalah beberapa kesan yang ditimbulkan pertama kalinya oleh *User Interface*.

Jika anda ingin membaca sebuah buku tentang *User Interface* maka buku itu berjudul *About Face* yang ditulis oleh Alan Cooper & Robert Reimann.

Three Tier Programming/Many Tier Programming

Teknologi komputer mengenal banyak sekali buzzword yang kadang-kadang membuat seorang programmer yang berpengalaman pun bingung dibuatnya. Cobalah anda terangkan apa artinya istilah-istilah ini: File Server, Client Server, OLE (Object Linking Embedded), COM (Component Object Model), COM1 (Component Object Model 1), DNA (Distributed Network Architecture) dan sekarang yang sedang populer .Net.

Penulis masih berpikir bagaimana menerangkan arti Three Tier Programming ini dengan kata-kata atau metafora yang tepat. Bagaimana kalau anda sekarang membayangkan sedang membeli kue lapis di pasar. Kue lapis mempunyai variasi-variasi dalam jumlah lapisannya, ada yang tiga lapis dengan warna merah, kuning, putih dan ada juga yang lima lapis dengan warna yang macam-macam pula. Mengapa pembuat kue lapis itu membuat dengan banyak lapis? Satu alasan yang mungkin benar adalah agar terlihat menarik dengan warna-warninya. Alasan lainnya mungkin agar pembeli mempunyai variasi dalam hal memakannya. Bisa langsung dimakan sepotong demi sepotong, bisa juga dengan cara memakannya lapis demi lapis. Alasan yang tepat mungkin pembaca perlu melakukan sedikit survey dengan menanyakannya secara langsung kepada pembuatnya.

Kembali kepada program, mengapa ada seorang iseng yang melontarkan ide untuk membuat program seperti kue lapis. Dalam hal pembuatan kue pun, membuat banyak lapis kue seperti itu membutuhkan lebih banyak kerja, apalagi membuat program dengan banyak lapis seperti kue. Tidak diragukan lagi bahwa membuat program dengan cara lapis-lapis seperti ini akan jauh lebih banyak memakan waktu. Alasannya apa sehingga banyak pembuat program mau saja mengikuti ide dari orang iseng itu? Bila alasan dalam pembuatan kue masih diperdebatkan kegunaannya, maka untuk alasan lapis pada pembuatan program, penulis merasa yakin akan kegunaannya. Seperti halnya dalam pembuatan kue yang mempunyai variasi dalam jumlah lapisnya maka dalam program pun demikian di mana jumlah lapisnya tergantung dari perancang aplikasi.

Bila hanya ada satu kata untuk menjawab kegunaan banyak lapis program maka kata itu adalah fleksibilitas. Bila ingin adanya perubahan pada lapis User Interface - Form maka yang hanya perlu diubah adalah form itu sendiri dan tidak perlu mengubah seluruh struktur aplikasi. Bila seorang pengusaha Mini Market yang sukses, yang menggunakan aplikasi Ms Access kemudian menjadi pengusaha menengah dan merasa Ms Access ini sudah tidak mencukupi lagi kebutuhannya, karena sekarang komputer jaringannya sudah ada 30 komputer. Dia ingin mengubah sistem mini marketnya.

Apakah pengusaha itu harus membuat baru semuanya dari awal? Alangkah malangnya si pengusaha itu bila demikian. Pada suatu hari si pengusaha menelpon kepada pembuat programnya untuk datang dan terjadilah percakapan antara pengusaha dan pembuat program.

Pengusaha: “Tiga tahun yang lalu ketika saya ingin membuat sistem mini market, saya memutuskan untuk membeli program dari anda karena walaupun harganya beberapa kali lipat dari penawaran yang masuk, kami mendengar nama baik perusahaan anda. Sekarang sistem yang anda buat sudah tidak bisa lagi mendukung perkembangan perusahaan karena penggunanya sudah 30 orang. Saya ingin mengubah sistem ini sehingga dapat mendukung operasi pekerjaan. Tolong buat perincian biayanya dan berapa lama sistem ini dapat selesai.”

Pengembang: “Harga Rp x.xxx.xxx dan waktunya sekarang juga akan kami kerjakan dan mudah mudahan sore nanti selesai.”

Pengusaha ini terdiam sejenak, kebingungan. Dalam hatinya dia berpikir apakah orang ini tidak waras. Bagaimana bisa mengubah sistem aplikasi yang telah dibuat selama satu tahun lebih dapat diubah hanya dalam hitungan jam dan harganya juga hanya 7 angka yang bisa diterjemahkan antara Rp 1 juta s/d Rp 9.999.999. Itu rahasia mereka berdua.

Pengusaha: “A aaa … apakah anda tidak bergurau?”

Pengembang: “Tidak, Pak. Karena semua fasilitas sudah ada di situ dan saya hanya tinggal mengubah datanya dari Ms Access ke SQL server. Dan saya rasa itu hanya membutuhkan waktu antara satu sampai dengan dua jam. Namun untuk lebih amannya saya menjanjikan sampai dengan tutup kantor ini semuanya beres.”

Dengan mengucapkan syukur pengusaha menjabat tangan pengembang itu dan berkata: “Pantaslah teman-teman saya menganjurkan nama perusahaan anda ketika saya minta saran perusahaan pengembang. Saya berjanji akan memberikan nama perusahaan anda kepada semua relasi-relasi saya bila mereka membutuhkan sistem untuk usahanya. Apa sih, rahasianya sehingga bisa secepat itu?”

Dengan rendah hati si pengembang mengucapkan terima kasih dan berkata, “Saya hanya menggunakan teknologi kue lapis, Pak.”

Benar saja, dalam 90 menit semua masalah si pengusaha itu terselesaikan dengan baik. Dia pulang dengan senyum di bibirnya dan perasaan sangat puas karena perjuangannya untuk selalu mendorong dirinya menjadi yang terbaik dari potensi dirinya terbayar, bukan saja dari materi tetapi membuat orang bahagia karena usahanya.

Dari uraian-uraian di atas, dapatkah anda mendefinisikan apa artinya Three Tier Programming?

Bagi penulis, apapun definisinya yang penting adalah kegunaannya. Sekali lagi penulis tidak ingin mencoba memberikan teori panjang lebar tentang masalah ini karena banyak buku yang memberikannya lebih baik. Salah satu buku yang dirasa sangat baik menjelaskan masalah ini adalah buku dari Rockford Lhotka, Visual Basic .Net Business Object. Pembaca mungkin kaget dengan kata .Net. Saya hanya pernah belajar Ms Access dan VBA dan tidak mengerti .Net. Apa urusannya buku itu dengan VBA? Kalau perasaan awal anda seperti itu mungkin kemampuan programming anda tidak akan pernah bisa mencapai ke level yang tinggi karena dari buku tentang masalah pengembangan berorientasi objek yang ada, sepengetahuan penulis tidak ada satu pun buku yang judulnya OOP untuk Ms Access. Cukup banyak buku tentang OOP untuk VB tapi tidak untuk Access. Jika anda berpegang pada patokan kata Ms Access untuk membeli buku tentang OOP maka anda akan terjebak dalam lingkup pengetahuan yang sempit. Bila ini memberikan perasaan yang lebih baik kepada anda, pada buku Business Object itu, penulis akan berterus terang mengatakan bahwa penulis pun hanya bisa mengerti secara penuh pada bab satu Distributed Architecture dari

sebelas bab yang ada. Telah begitu lama penulis merasa ragu dengan buku-buku yang menganjurkan pemisahan antara bisnis objek dengan data objek karena meningkatnya kompleksitas dan overhead yang ditimbulkan. Penulis telah sejak lama memikirkan bagaimana caranya memperbaiki arsitektur ini dengan tetap menjaga agar fleksibilitas pada perubahan database tidak hilang. Kemudian buku .Net dari Lhotka melontarkan ide tentang data portal di mana pada intinya Lhotka menyarankan menyatukan data objek itu dalam bisnis objek. Masalah yang telah mengganggu penulis sejak lama didapat dari sebuah buku yang sebenarnya bukan ditujukan untuk VBA. Seorang bijak mengatakan, ideas are free. Betapa benarnya dia, hampir. Jika anda belum mengerti mengapa penulis setuju dengan pendapat penulis itu, bayangkan anda mempunyai 100 bisnis objek maka anda bisa membutuhkan 100 lagi untuk data objek. Tetapi dengan idenya tentang data portal maka yang dibutuhkan hanya beberapa kelas untuk data portal.

Usaha-usaha seorang perancang aplikasi dengan membuat beberapa layer programming ini dapat dikatakan sedang melakukan proses arsitek dalam sebuah program. Apa itu arsitektur? Jika anda ingin definisi ilmiah tentang arsitektur ini, penulis tidak akan menyalahkan tetapi penulis tidak akan mengikuti. Apa bedanya antara arsitektur dengan analisis dan desain? Apa bedanya antara software architecture dengan software Engineering? Beberapa ahli menginginkan pemisahan disiplin ilmu di antara keduanya tetapi yang lain tidak. Biarkanlah mereka bertarung sendiri. Penulis akan memberi definisi sendiri tentang arsitektur ini, walaupun jauh dari tepat. Jika anda senang dengan permainan Puzzle yang menggabungkan gambar-gambar yang dipotong-potong kecil menjadi puluhan atau bahkan ratusan potongan gambar, dan jika anda sedang menyusunnya maka penulis akan dengan berani mengatakan bahwa anda sedang melakukan proses arsitektur dalam arti menempatkan sesuatu pada tempatnya yang sesuai. Dan bila anda telah bisa melakukan itu dengan baik, penulis akan memberi gelar kepada anda Puzzle Architect - Jangan mengambil definisi penulis ini secara serius.

Evolusi Pengembangan Aplikasi

“Revolusi atau mati. Ah tidak, kami ingin evolusi.”

Evolusi adalah kata-kata yang menggambarkan tentang perubahan sesuatu sedikit demi sedikit secara terkontrol. Bill Gates mengatakannya dengan kalimat, *You do it bit by bit*. Pepatah Amerika mengatakannya dengan kalimat, *Do not bite off more than what you can chew* - jangan menggigit lebih banyak daripada yang dapat dikunyah. Teori Darwin tentang evolusi mengajarkan asal muasal manusia. Steven Hawkins, ilmuwan Inggris menjelaskan teori tentang asal muasal dunia dengan teori *Black Hole* yang menjelaskan asal dunia dalam evolusi jutaan tahun.

Teori evolusi seharusnya menjadi dasar bagi pengembangan sebuah aplikasi karena tanpa didasari hal itu sebuah aplikasi ingin memecahkan semua masalah dalam sekaligus yang hasilnya tidak jelas hendak ke mana aplikasi akan dibuat.

Untuk memberikan contoh evolusi sebuah aplikasi, maka contoh klasik pada buku-buku program adalah Northwind Trader. Kami pun akan mengikutinya dengan mengambil Northwind sebagai contoh aplikasi.

Phase Pertama Northwind: Phase pertama dari aplikasi contoh ini adalah apa yang anda dapatkan. Aplikasi ini mempunyai karakteristik semua objek *Ms Access* seperti *Tabel, Query, Form, Report* berada dalam satu database.

Phase Kedua Northwind: Phase kedua bisa dibuat menjadi bermacam-macam bentuk. Karena phase pertama Northwind ini adalah aplikasi yang sederhana maka kita dapat membuatnya menjadi lebih baik dengan usaha yang minimal. Misalnya, kita dapat saja membuat phase kedua ini dengan hanya memisahkan datanya (*table*) dan *query* ke dalam database tersendiri.

Phase Ketiga Northwind: Pada phase ketiga ini, bukan hanya database saja yang dipisah dari aplikasi namun bisa saja aplikasinya kemudian dibagi menjadi beberapa subsistem yang saling berhubungan.

Ciri-ciri dari fase ketiga yang mungkin bisa dikembangkan adalah sebagai berikut:

- *OOP*. Aplikasi yang dibangun akan dibuat dengan metoda *objek oriented programming*.
- *Distributed Data*. Data (*tabel*) dan *Query* akan dibuat dalam database yang terpisah dan data-data tersebut akan dipecah lagi menjadi beberapa database. Misalnya *Database Master*, *Database Product*, *Database Penjualan* .
- Aplikasi dibagi menjadi beberapa subsistem seperti misalnya *Subsistem Master*, *Subsistem Produk* dan *Subsistem Penjualan*.
- Aplikasi akan dibagi menjadi banyak lapis atau lebih dikenal dengan nama *Three Tier* atau *Many Tier Programming*.
- Aplikasi dapat menggunakan *Ms Access* database ataupun *Ms SQL Server*.

Mungkin bagi yang telah mengenal teori-teori *Objek* akan mengatakan betapa beraninya penulis mengatakan bahwa aplikasi *VBA* (*Visual Basic For Aplication*) dapat menerapkan *OOP* karena *VB* (*Visual Basic*) saja dikritik karena tidak bisa menerapkan metoda *OOP* secara penuh. Kami tidak dapat menyalahkan mereka yang mengkritik karena pada kenyataannya memang *VB* atau *VBA* tidak bisa menerapkan *OOP* secara penuh karena tidak adanya fungsi *Inheritance*. Apapun kritik-kritik mereka penulis tetap akan menamakannya demikian. Kalaupun terpaksa harus diralat mungkin akan diganti dengan nama *OOP Tanpa Inheritance*.

Penutup

Setiap pertemuan selalu ada perpisahan.

Satu hal yang pasti adalah bahwa berapa pun besarnya pengetahuan yang akan anda dapatkan dari artikel, buku-buku atau sekolah, pengetahuan tersebut akan menjadi berguna bila bisa diterapkan pada pembuatan sebuah sistem aplikasi di tempat kerja anda, baik perusahaan anda sendiri atau orang lain. Dan pada akhirnya tujuan dari segala jerih payah kita sekalian adalah mendapatkan sesuatu, baik itu bermotif ekonomi atau bermotif sosial. Pertanyaannya adalah apakah anda telah siap dengan kemungkinan yang terburuk yang bisa terjadi menimpa kita sekalian. Bagi perancang atau pengembang, adalah hal yang menakutkan apabila produk yang dibuat ternyata tidaklah terlalu berguna bagi penggunanya. Dan untuk mengetahui apakah produk yang akan kita buat bisa sukses atau tidak di pasaran adalah sebuah taruhan tersendiri. Dan taruhan itu akan bertambah besar resikonya bila produk yang kita buat adalah produk yang akan dijual secara retail bukan produk internal yang digunakan pada perusahaan di mana kita bekerja. Apakah anda siap dengan kekejaman dunia ini? *No gut no glory*.

Dan bila direncanakan dengan baik, dari segi desain database, objek, user interface dan desain arsitektur, maka perencanaan itu akan mengurangi resiko yang akan timbul karena potensinya dalam penggunaan ulang yang jauh lebih besar bila pemrogramannya berorientasi objek dan banyak lapis arsitektur.

Dan sekarang pertanyaan sulit yang hanya bisa dijawab oleh anda sendiri adalah apakah benar bidang teknologi informasi inilah bidang yang menjadi pilihan hati anda? Banyak kenyataan yang terjadi pada orang yang kita kenal yang setelah bergelut dengan bidang ini selama bertahun-tahun dan bahkan telah mendapatkan gelar master di bidang ini pada akhirnya meninggalkan segala jerih payahnya karena ternyata mereka tidak mau memperbaharui perkembangan pesat dari ilmu teknologi informasi. Contoh dari perkembangan ini adalah pada saat perubahan dari DOS ke Windows dan yang terjadi pada *Microsoft Visual Basic®* dengan dikeluarkannya produk *VB .NET* - betapa ngerinya kita pada revolusi. Apakah ada perasaan ngeri pada kita semua? Sekarang kita kembali lagi ke titik nol, atau mungkin setengah?

I am the master of my own soul. I am the captain of my own destiny.
I am the master of my own soul. I am the captain of my own destiny.
……

……
I am the master of my own soul. I am the captain of my own destiny.

Jika anda sekalian mendengar penulis mengatakan kalimat itu pada diri sendiri berulang ulang sampai 100 kali setiap hari, jangan kaget dan takut karena penulis sendiri juga sedang menutupi, mengatasi atau mungkin menghindari perasaan ngeri itu.

Apa pendapat anda? Semoga artikel ini dapat sedikit memberikan pencerahan kepada anda. Penulis ingin mendapatkan input baik berupa kritik, saran yang membangun sehingga penulis juga dapat belajar dari pembaca sekalian.

Pustaka

Thou shall read books.

Seorang bijak mengatakan, pengetahuan seorang tidak diukur dari tingginya pendidikan atau banyaknya gelar yang dimiliki, tetapi dari bagaimana dia bisa menerapkan pengetahuan yang dimilikinya pada saat dan situasi yang tepat.

Ah mungkin pendapat di atas ada benarnya. Jadi bila pembaca merasa pendidikan anda tidak tinggi jangan merasa kecil hati karena banyak contoh orang yang telah memberikan sumbangsihnya yang besar bagi dunia justru dari mereka yang pendidikannya tidaklah tinggi. Tetapi satu hal yang pasti adalah bahwa mereka adalah orang yang selalu ingin belajar. Cara belajar bisa bermacam-macam, bisa dari sekolah, kursus, tukar pikiran, pekerjaan dan tentu saja membaca buku. Karena menyadari pentingnya buku dalam penyebaran pengetahuan, maka di bawah ini penulis akan memberikan buku-buku yang mungkin bisa membantu pembaca untuk meningkatkan pengetahuannya.

Dan perlu diingat bahwa artikel ini adalah hanya sebuah peta kecil yang berguna untuk menemukan sesuatu, dan sesuatu itu adalah pengetahuan yang ada dalam buku-buku. Jadi artikel ini akan jauh lebih berguna bila kemudian anda melanjutkannya dengan membaca dan tergantung dari kemampuan anda sekarang, anda mungkin membutuhkan waktu satu sampai lima tahun untuk benar-benar mengerti topik topik yang penulis sebutkan di atas. Dan jika anda telah membaca, mengaplikasikannya dan mendapat pengetahuan dari buku-buku tersebut, anda harus memberikan penghargaan kepada diri sendiri dengan memberikan gelar tambahan yang bisa saja anda karang sendiri *MCS - Master of Computer by Self*. Tidak ada yang memberikan anda pekerjaan rumah atau perintah untuk membaca bab sekian sampai bab sekian karena akan ada ulangan harian atau ulangan umum. Dan tidak ada pula yang akan memberikan anda nilai dengan memberikan rapot atau daftar nilai. Anda adalah siswa yang kesepian - *kasihan deh loe*. Tetapi penulis yakin, pada akhirnya anda akan merasakan kenikmatan, kebahagian tersendiri setelah anda mendapatkan pengetahuan itu, karena justru pada saat anda berhenti atau lulus sekolah lah pencarian ilmu pengetahuan itu baru dimulai – dan pencarian itu tidak akan pernah ada akhirnya. Adalah kenyataan yang menyedihkan bila seseorang justru berhenti belajar setelah mendapatkan gelar akademi dari institusi pendidikan . *Have a pleasant journey* atau mungkin lebih tepat *Have pleasant journeys!*

Daftar Pustaka:

Programming VB, VBA, VB .NET

- Access VBA Programming for the Absolute Beginner. Michael Vine. Premier Press. 2003.
- Access 97 Expert Solutions: Expert Advice for the Serious Developer! Stan Leszynski. QUE. 1997.
- ACCESS 2000 Developer's Handbook™. Ken Getz , Paul Litwin, Mike Gilbert. SYBEX.1999.
- ACCESS 2002 Enterprise Developer's Handbook™. Paul Litwin, Ken Getz, Mike Gunderloy. SYBEX. 2002.
- VBA Developer's Handbook™. Ken Getz & Mike Gilbert. SYBEX. 1997.
- Developing COM/ActiveX™ Components with Visual Basic® 6. A Guide To The Perplexed. Dan Appleman. SAMS. 1999.
- Moving to VB .Net: Strategies, Concepts and Code. Dan Appleman. Apress. 2003.
- Office 2003 XML for Power Users. Matthew MacDonald. Apress. 2004.
- Building N-Tier Applications with COM and Visual Basic® 6.0. John Wiley & Sons, Inc. 1999.
- Access 97 Developer's Handbook. Third Edition. Paul Litwin. Ken Getz. Mike Gilbert. SYBEX. 1997.
- Ready-to-Run Visual Basic® Code Library: Tips, Tricks, and Workarounds for Better Programming. Rod Stephens. John Wiley & Sons, Inc. 1999.
- Programming Microsoft® Office 2000 Web Components. Dave Stearns. Microsoft Press. 1999.

Objek Oriented

- Beginning Objects with Visual Basic 5. Peter Wright. Wrox Press Ltd. 1998.
- Visual Basic 6. Object-Oriented Programming. Gene Swartzfager, Ramesh Chandak, Purshottam Chandak, Steve Alvarez. CORIOLIS. 1999.
- Doing Objects in Visual Basic® 6. The Authoritative Solution. Deborah Kurata. SAMS. 1999.
- Visual Basic .NET Business Objects. Rockford Lhotka. Wrox Press Ltd. 2003.
- Object-Oriented Methods. Pragmatic Considerations. James Martin & James J. Odell. Prentice-Hall International, Inc. 1996.
- Object Models: Strategies, Patterns, & Applications. Peter Coad, David North, Mark Mayfield. Yourdon Press Computing Series.1997.
- Case Studies in Object Oriented Analysis & Design. Edward Yourdon & Carl Argila. Yourdon Press Computing Series. 1996.
- Designing Flexible Object-Oriented Systems with UML. Charles Richter. Macmillan Technical Publishing. 1999.
- Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Craig Larman. Prentice Hall PTR. 1998.
- Visual Basic® Developer's Guide to UML and Design Patterns. Yair Alan Griver, Matthew Arnheiter, Michael Gellis. SYBEX. 2000.

User Interface

- About Face 2.0. The Essentials Of Interaction Design. Alan Cooper & Robert Reimann. Wiley Publishing, Inc. 2003.

Database

- An Introduction to Database Systems. C.J. Date. Addison-WesleyPublishing Company. 1995.
- Seluk Beluk Database Relasional. Edisi Kedua. Mark Whitehorn & Bill Marklyn. Erlangga. 2003.
- Fundamentals of Database System. Third Edition. Ramez Elmasri & Shamkant B. Navathe.

Addison-Wesley. 2000.
- Database Design For Mere Mortals: A Hands-On Guide To Relational Database Design. Michael J. Hernandez. Addison-Wesley Developers Press.1997.
- Building Enterprise Information Architecture: Reengineering Information Systems. Melissa A. Cook. Prentice-Hall PTR. 1996.

Lainnya

- How Computer Programming Works. Daniel Appleman. Apress. 2000.
- Planning Smarter: Creating Blueprint-Quality Software Specifications. Tyson Gill. Prentice Hall PTR. 2002.
- Rapid Development: Taming Wild Software Schedules. Steve McConnell. Microsoft Press. 1996.
- The Art of Software Architecture. Design, Methods and Techniques. Stephen T. Albin. Wiley Publishing, Inc. 2003.
- Software Requirements. Karl E. Wiegers. Microsoft Press. 1999.
- Practical Standards for Microsoft Visual Basic®. James D. Foxall. Microsoft Press. 2000.
- eXtreme Programming in Action: Practical Experience from Real World Projects. Martin Lippert, Stefan Roock, Henning Wolf. John Wiley & Sons, Ltd. 2002.

Riwayat Hidup

Hermawih Hasan. Lahir di Pamanukan, Jawa Barat, 27 Agustus 1961. Menamatkan Sekolah Menengah Atas di SMA St. Aloysius Bandung pada tahun 1981. Menyelesaikan program S1 di jurusan "Information System" di University Of The Pacific, Stockton, California, USA pada tahun 1987. Menyelesaikan Program Master of Business Administration di Jakarta dan Magister Management di Universitas Krisnadwipayana di Bekasi pada program akhir pekan. Pengalaman kerja di berbagai bidang, di antaranya: perhotelan, perikanan, perbankan dan terakhir di bidang piranti lunak. Sekarang sedang memfokuskan usahanya pada bidang pengembangan piranti lunak – khususnya bidang pendidikan.