REKAYASA ULANG (REENGINEERING)

Saat manager memodifikasi aturan-aturan bisnis untuk mencapai keefektifan dan komposisi yang lebih besar, perangkat lunak harus tetap berjalan maju. Artinya penciptaan sistem berbasis komputer yang besar berarti memodifikasi dan atau membangun aplikasi yang sudah ada sehingga menjadi kompeten untuk memenuhi kebutuhan bisnis pada masa yang akan datang.

I. Rekayasa Ulang Proses Bisnis/ Business Process Reengineering (BPR).

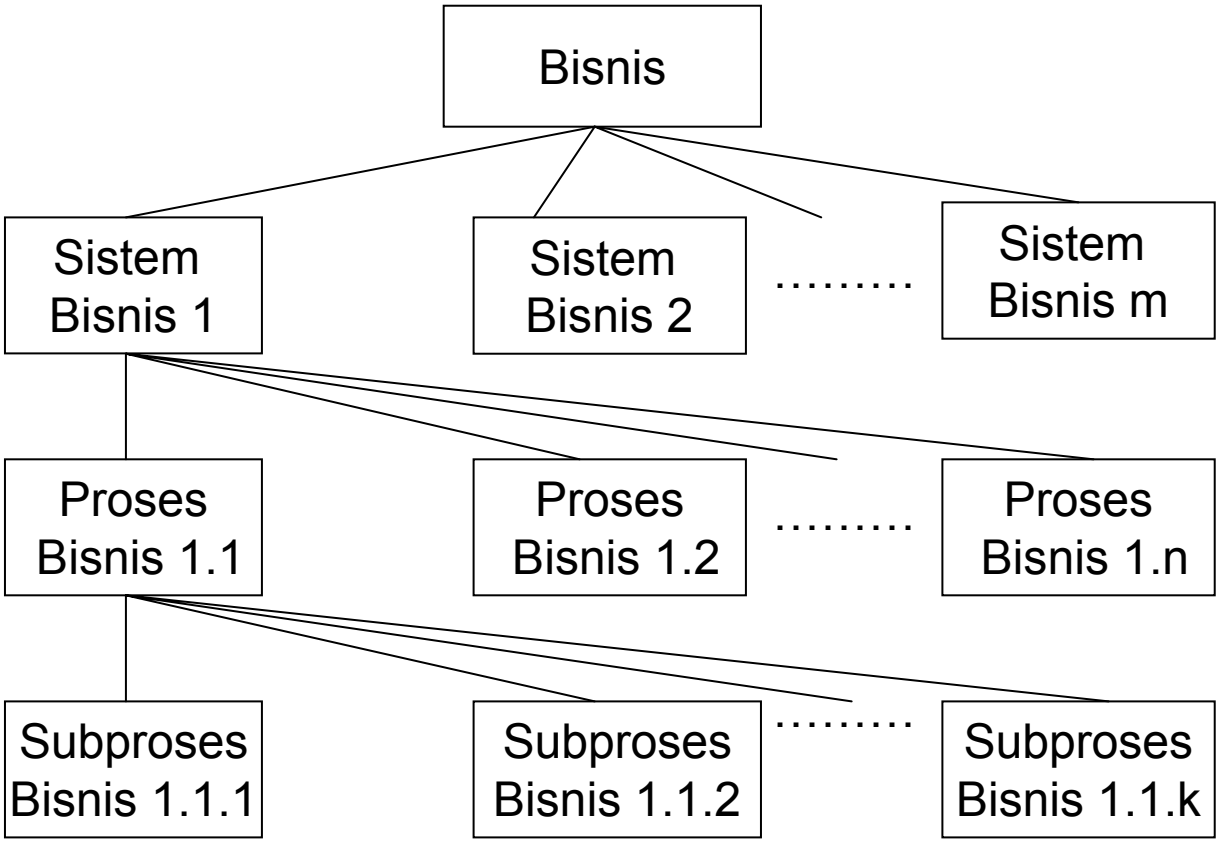
BPR meluas jauh diluar lingkup teknologi informasi dan rekayasa perangkat lunak.

A. Proses Bisnis.

Adalah serangkaian tugas yang dihubungkan secara logis yang dilakukan untuk mencapai hasil akhir bisnis yang telah ditentukan.

Contoh proses bisnis: Perancangan produk baru, pembelian jasa dan suplai, merekrut tenaga kerja baru, pembayaran pemasok.

Masing-masing memerlukan serangkaian tugas dan memiliki sumber daya yang berbeda dalam bisnis tersebut. Setiap proses memiliki pelanggan terbatas yang menerima hasil akhir. Proses bisnis mengharuskan kelompok organisasi yang berbeda berpartisipasi dalam “tugas-tugas yang dihubungkan secara logis” yang menentukan proses.



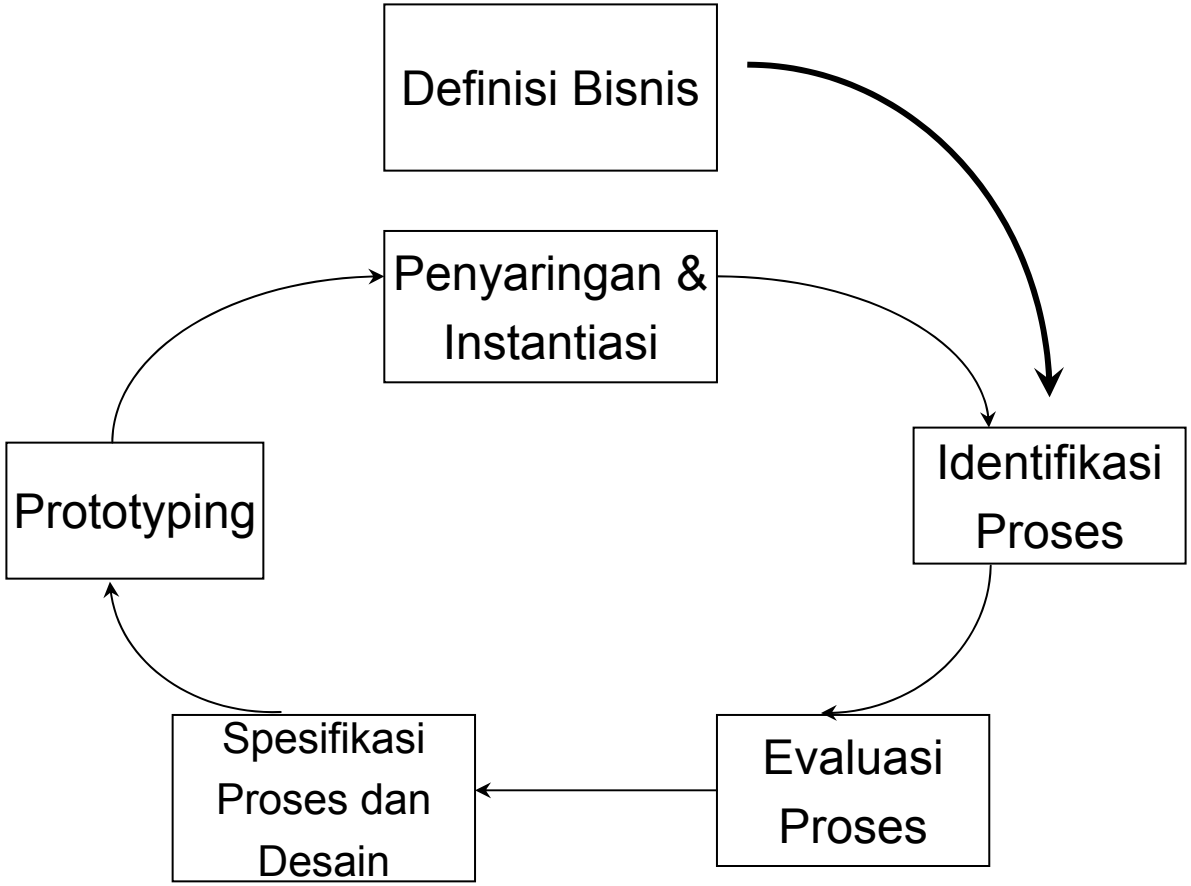
Gambar Hirarki bisnis suatu sistem

B. Prinsip-prinsip BPR.

Dalam setting yang ideal, BPR harus terjadi dalam cara top down.

- Kumpulkan disekitar hasil akhir, bukan tugas
- Buatlah mereka yang menggunakan output proses tersebut melakukan proses itu
- Gabungkan kerja pemrosesan informasi ke dalam usaha nyata yang menghasilkan informasi mentah
- Perlakukan sumber daya yang tersebar secara geografis seolah-olah mereka tersentralisasi
- Sambungkan aktifitas paralel sebagai pengganti pengintegrasian hasil mereka
- Letakkan titik keputusan dimana kerja mereka, dan bangunlah kontrol ke dalam proses
- Tangkaplah data sekali, pada sumbernya

C. Model BPR


Definisi Bisnis
Penyaringan & Instantiasi
Identifikasi Proses
Evaluasi Proses
Spesifikasi Proses dan Desain
Prototyping

II. Rekayasa Ulang Perangkat Lunak

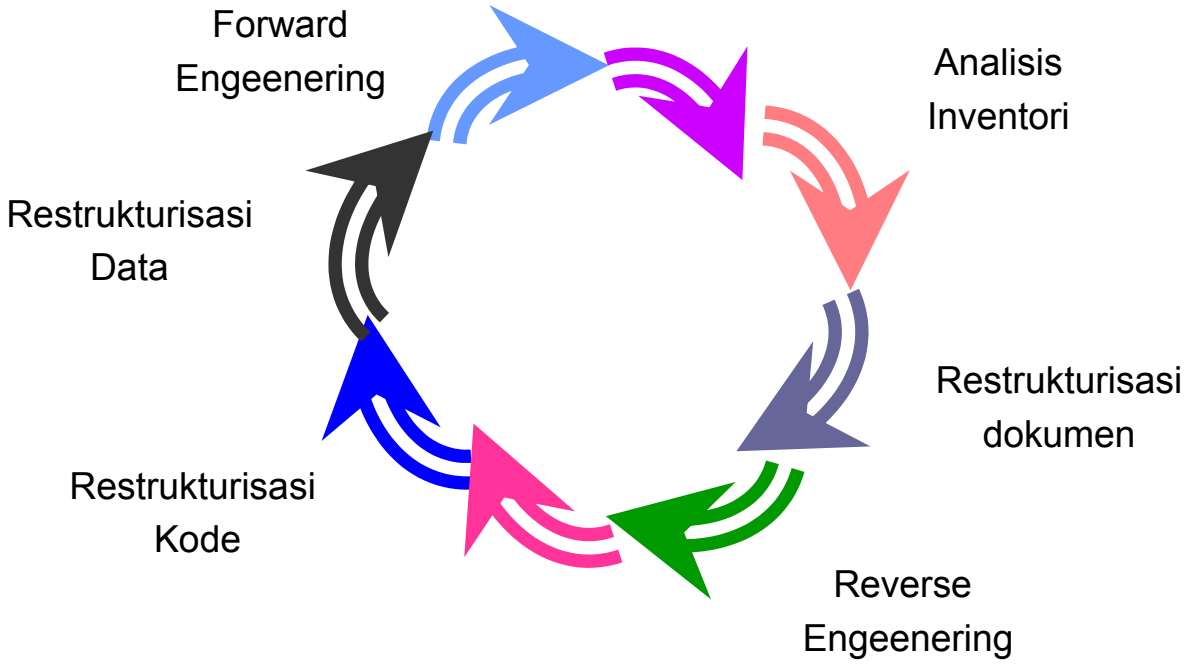
A. Pemeliharaan Perangkat Lunak

Hanya sekitar 20% dari semua usaha pemeliharaan untuk membetulkan kesalahan dan 80% untuk menyesuaikan sistem terhadap perubahan dalam lingkungan eksternalnya, dengan membuat peningkatan yang dibutuhkan oleh pemakai, dan perekayasaan kembali suatu aplikasi untuk digunakan di masa yang akan datang

B. Model Proses Rekayasa Ulang Perangkat Lunak

Reverse Engineering/ Rekayasa Terbalik (Pemahaman kerja internal dari suatu program) mungkin harus terjadi sebelum restrukturisasi dokumen dapat dimulai.

C. Model Proses Rekayasa Kembali



III. Reverse Engineering

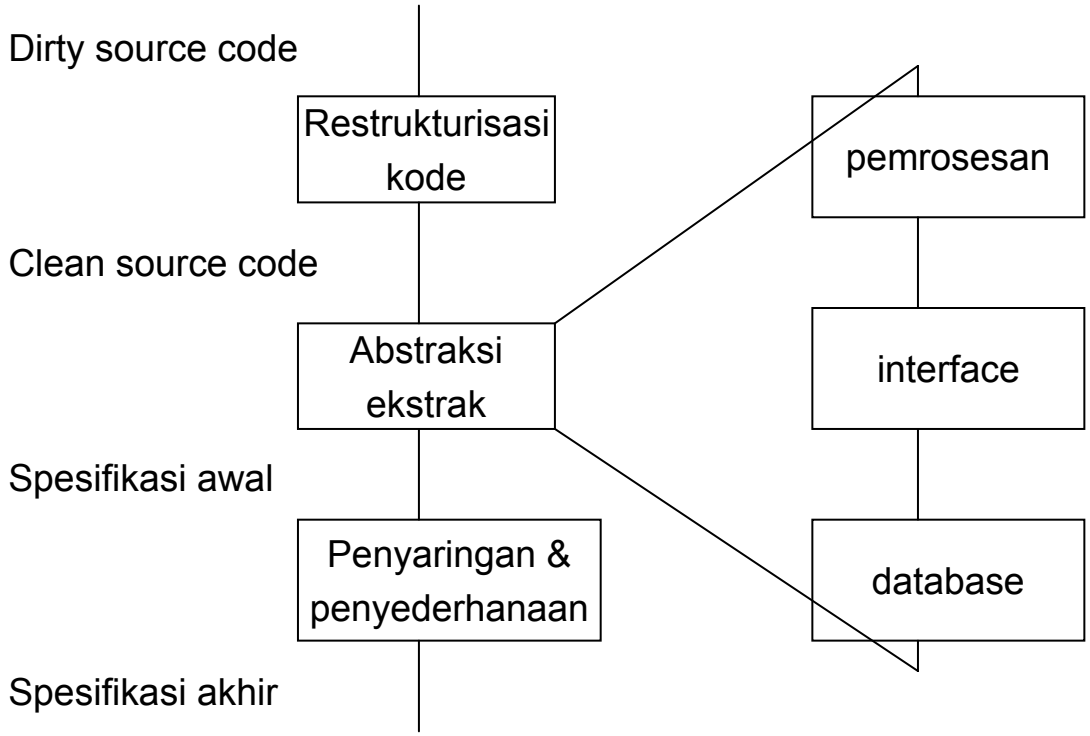
Reverse engineering dapat mengekstrak desain dari kode sumber, tetapi tingkat abstraksi, kelengkapan dokumentasi, tingkat dimana peranti dan analis bekerja sama dan direksionalitas proses sangat bervariasi.

◊ Tingkat abstraksi; proses reserve engineering harus mampu menggunakan:

- Representasi prosedural (tingkat yang rendah)
- Program dan informasi struktur data (tingkat yang lebih tinggi)
- Data dan model aliran kontrol (tingkat yang sangat tinggi)
- Model hubungan entitas (tingkat yang tinggi)

◊ Kelengkapan proses mengacu pada tingkat detail yang diberikan pada suatu tingkat abstraksi. Kelengkapan meningkat berbanding lurus dengan jumlah analisis yang dilakukan.

◊ Direksionalitas; bila satu jalur maka semua informasi yang diekstrak dari kode sumber diberikan kepada perekayasa yang dapat menggunakannya selama pemeliharaan. Bila dua arah informasi diisikan ke peranti rekayasa ulang yang akan merestrukturisasi atau memunculkan lagi program lama.



A. Reverse Engineering untuk Memahami Pemrosesan

Aktivitas reverse engineering real pertama-tama dimulai dengan usaha memahami kemudian mengekstrak abstraksi prosedural yang direpresentasikan oleh kode sumber.

- Fungsionalitas keseluruhan dari sistem harus dipahami sebelum kerja reverse engineering yang lebih detail dilakukan.
- Teknik segmentasi program sebagai cara untuk mengidentifikasi pola prosedural dengan sebuah model dan kemudian mengemas lagi pola-pola itu ke dalam sebuah fungsi yang penting.

B. Reverse engineering untuk memahami data.

Pada tingkat sistem, stuktur data global (misal file, database)sering direkayasa ulang untuk mengakomodasi paradigma manajemen database baru (misal gerakan dari flat file ke sistem database relasional atau OO)

- Struktur Data Internal

  Pendekatan untuk kelas reverse engineering:

  1. Identifikasi flag dan struktur data lokal pada program yang merekam informasi penting mengenai struktur data global.
  2. Tetapkan hubungan antara flag dan struktur data lokal dan global.
  3. Untuk setiap variabel yang merepresentasikan array atau file, daftarlah semua variabel lain yang memiliki hubungan logis dengannya.

- Struktur database

Langkah-langkah mendefinisikan model data yang ada ke model database baru.

1. Bangun model obyek awal.
2. Tentukan kunci calon.
3. Saring kelas-kelas tentatif
4. Definisikan generalsisasi.
5. Temukan hubungan.

C. Interface Pemakai Reverse Engineering

Untuk memahami secara penuh interface (UI) yang sudah ada, struktur
dan tingkah laku interface harus ditentukan. Tiga pertanyaan
mendasar yang harus dijawab pada saat reverse engineering suatu UI dimulai:

1. Apakah aksi dasar yang harus diproses interface?
2. Apa deskripsi respon perilaku sistem terhadap aksi?
3. Konsep ekivalensi apa yang relevan di sini?

Aljabar proses dapat digunakan untuk merepresentasikan tingkah laku suatu interface dalam cara yang formal.

Nama	Notasi Ajabar	Arti
Agen inaktif	0	
Konstant	*A*	Agen *A*
Prefiks	*a.E*	Lakukan aksi *a* dan bertindak seperti agen *E*
Penjumlahan	E_1+E_2	Agen E_1 digabung dengan agen E_2
Komposisi	E_1|E_2	Berlaku seperti agen E_1 atau agen E_2
Restriksi	E_1\L	Agen *E* dibatasi untuk aksi pada *L* aturan
Pelabelan ulang	*E[f]*	Penamaan ulang agen *E* meurut fungsi *f*
Substitusi	*E/X*	Mengganti agen *X* dengan agen *E*
Rekursi	Fax(*X*=*E*)	Agen *X* seperti *X*=*E*

Contoh: P = c.D + m.M

Menyatakan bahwa agen *P* bertingkah laku dalam suatu cara yang identik dengan aksi *c* dan tingkah laku resultan *D* atau aksi *m* serta tingkah resultan agen *M.*

IV. Restrukturisasi

Manfaat yang dapat diperoleh bila perangkat lunak direstrukturisasi:

- Membuat program memiliki kualitas lebih tinggi.
- Meningkatkan produktivitas dan membuat proses belajar menjadi lebih mudah.
- Mengurangi usaha yang diperlukan untuk pemeliharaan.
- Membuat perangkat lunak menjadi lebih mudah diuji dan debug.

A. Restrukturisasi kode

Dilakukan untuk menghasilkan desain yang menghasilkan fungsi sama tetapi dengan kualitas yang lebih tinggi daripada program semula.

B. Restrukturisasi data

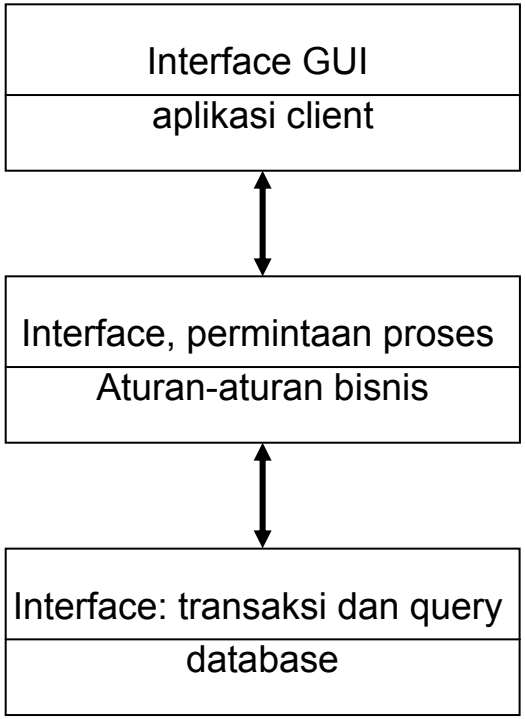
Tujuannya adalah mengekstrak item dan obyek data, untuk memperoleh informasi aliran data, dan memahami struktur data yang ada yang telah diimplementasikan. Aktivitas ini disebut juga analisis data.

V. Forward Engineering

Proses ini menerapkan prinsip-prinsip rekayasa perangkat lunak, konsep dan metode untuk membuat ulang suatu aplikasi yang sudah ada. Tidak hanya membuat ekivalen modern dari program lama, tetapi lebih mengintegrasikan pemakai baru dan persyaratan teknologi ke usaha rekayasa ulang.

A. Forward Engineering untuk Arsitektur Client/Server

Rekayasa ulang untuk aplikasi C/S dimulai dengan analisis yang mendalam terhadap lingkungan bisnis yang mencakup main frame yang ada.



B. Forward Engineerung untuk Arsitektur Berorientasi Obyek.
Bila sistem yang direkayasa ulang meluaskan fungsionalitas atau tingkah laku aplikasi semula, maka use case diciptakan. Model data yang dibuat selama reverse engineering kemudian digunakan dalam hubungannya dengan pemodelan CRC (Pemodelan kelas – tanggung jawab – kolaborator) untuk membangun basis bagi definisi kelas. Hirarki kelas, model hubungan obyek, model tingkah laku obyek dan subsistem didefinisikan dan desain OO dimulai.

C. Interface Pemakai Forward Engineering
Model untuk merekayasa ulang interface pemakai:

1. Memahami interface original dan data yang bergerak di antaranya serta sisa aplikasi.
2. Modelkan lagi tingkah laku yang diimplikasikan oleh interface yang ada ke dalam sederetan abstraksi yang memiliki arti konteks GUI.

3. Lakukan peningkatan yang membuat mode interaksi lebih efisisen.
4. Bangun dan integrasikan GUI yang baru.

VI. Ekonomi Rekayasa Ulang.

Model analisis-manfaat untuk rekayasa ulang dengan sembilan parameter:

P_1 = biaya pemeliharaan tahunan untuk suatu aplikasi

P_2 = biaya operasi tahunan untuk suatu aplikasi

P_3 = nilai bisnis tahunan untuk suatu aplikasi

P_4 = biaya pemeliharaan tahunan yang diprediksi setelah rekayasa ulang

P_5 = biaya operasi tahunan yang diprediksi setelah rekayasa ulang

P_6 = nilai bisnis tahunan yang diprediksi setelah rekayasa ulang

P_7 = perkiraan biaya rekayasa ulang

P_8 = perkiraan waktu kalender rekayasa ulang

P_9 = faktor risiko rekayasa ulang (P_9 = 1,0 adalah nominal)

L = hidup sistem yang diharapkan (dalam tahun)

Biaya yang berhubungan dengan pemeliharaan terus-menerus dari aplikasi calon ditetapkan sebagai

$$C_{maint} = [P_3 – (P_1 + P_2)] X L$$

Biaya sehubungan dengan rekayasa ulang ditentukan dengan rumus

$$C_{reeng} = [P_6 - (P_4 + P_5) X (L – P_8) – (P_7 X P_9)$$

Maka keuntungan dari rekayasa ulang

$$\text{Cost benefit} = C_{maint} – C_{reeng}$$

Aplikasi yang memperlihatkan keuntungan biaya yang tinggi dapat direkayasa ulang, sementara kerja pada yang lain dapat ditunda sampai sumber-sumber daya dapat diperoleh.