REKAYASA PERANGKAT LUNAK

Oleh : Dr. Asep Juarna

(Sumber: Richard Fairley. *Software Engineering Concepts*. McGraw-Hill Int'l Ed.)

Bab I. Pengantar

- **Definisi :**
  Rekayasa perangkat lunak adalah disiplin manajerial dan teknis yang berkaitan dengan pembuatan dan pemeliharaan produk perangkat lunak secara sistematis, termasuk pengembangan dan modifikasinya, yang dilakukan pada waktu yang tepat dan dengan mempertimbangkan faktor biaya.

- **Tujuan :**
  Tujuan rekayasa perangkat lunak adalah memperbaiki kualitas produk perangkat lunak, meningkatkan produktivitas, serta memuaskan teknisi perangkat lunak.

- **Pengertian produk perangkat lunak :**
  Produk perangkat lunak adalah perangkat lunak yang digunakan oleh berbagai pengguna, bukan untuk pengguna pribadi.

- **Hal-hal yang perlu diperhatikan dalam pengembangan sebuah produk perangkat lunak :**
  - kebutuhan dan batasan-batasan yang diinginkan pengguna harus ditentukan dan dinyatakan secara tegas,
  - produk perangkat lunak harus dirancang sedemikian rupa sehingga mampu mengakomodasi paling tidak kepentingan tiga pihak berikut : pelaksana implementasi, pengguna, dan pemelihara produk,
  - penulisan *source code* harus dilakukan dengan hati-hati dan senantiasa melalui tahap uji,
  - dilengkapi dengan dokumen-dokumen pendukung seperti : prinsip pengoperasian, *user's manual*, instruksi instalasi, dokumen pemeliharaan,
  - menyiapkan bantuan pelatihan.

- **Tugas-tugas pemeliharaan perangkat lunak meliputi :**
  - analisa terhadap permintaan perubahan,
  - perancangan ulang dan modifikasi terhadap *source code* yang diikuti dengan serangkaian proses uji,
  - dokumentasi perubahan dan pembaruan dokumen-dokumen yang berkaitan dengan modifikasi,
  - penyebaran produk yang telah mengalami modifikasi ke situs-situs pengguna.

- **Jarak inteletual**
  - Pemetaan antara model dengan realitas yang dimodelkan dikenal sebagai jarak intelektual antara suatu persoalan dengan komputerisasi solusi atas persoalan tersebut.
  - Prinsip dasar rekayasa perangkat lunak adalah merancang produk perangkat lunak yang meminimalkan jarak intelektual.

♦ **Modul**

- Prinsip dasar untuk menangani kerumitan dalam rekayasa perangkat lunak adalah dengan melakukan dekomposisi terhadap sistem yang berukuran besar ke dalam beberapa subsistem yang lebih kecil.
- Unit dekomposisi tersebut dinamakan modul.
- Dalam dekomposisi tersebut harus ditetapkan pengantarmukaan (*interfacing*) antar setiap subunit, baik pengantarmukaan kendali maupun data.
- Pengantarmukaan kendali dilakukan dengan mekanisme hubungan pemanggilan (*calling*) antar modul.
- Pengantarmukaan data dilakukan dengan mekanisme penyampaian parameter (*parameter passing*) antar modul.

♦ **Programmer**

Programer adalah individu yang bertugas dalam hal rincian implementasi, pengemasan, dan modifikasi algoritma serta struktur data, dituliskan dalam sebuah bahasa pemrograman tertentu.

♦ **Software Engineer**

- *Software engineer* bertugas melakukan analisa, rancangan, uji dan verifikasi, dokumentasi, pemeliharaan perangkat lunak, serta pengelolaan proyek.
- *Software engineer* harus mempunyai keterampilan dan pengalaman seorang programmer

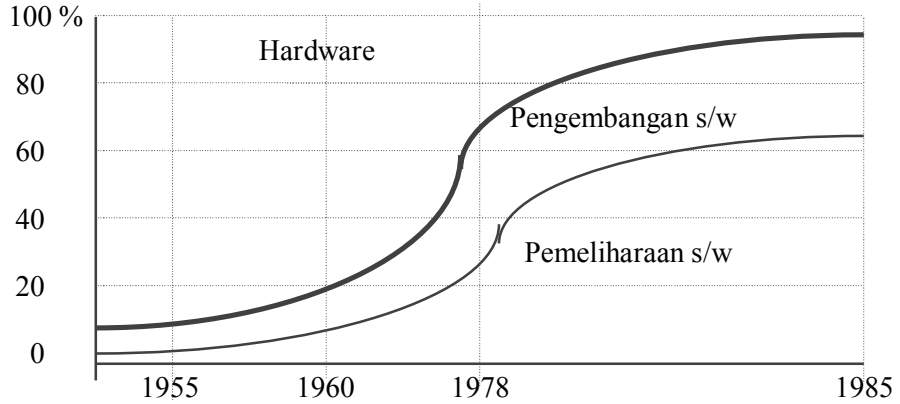
♦ **Kualitas Produk Perangkat Lunak**

Beberapa atribut yang merupakan ukuran kualitas perangkat lunak adalah :

- kegunaan, yaitu pemenuhan terhadap kebutuhan pengguna,
- keandalan, yaitu kemampuan melaksanakan fungsi yang diinginkan,
- kejelasan, yaitu penulisan program dilakukan secara jelas dan mudah dimengerti,
- efisiensi, terutama dalam waktu eksekusi dan penggunaan memory,

II. Beberapa Ukuran Yang Berkaitan Proyek

♦ **Total Upaya Yang Dicurahkan Untuk Perangkat Lunak**



Perubahan rasio biaya untuk h/w terhadap biaya untuk s/d 1985

♦ **Distribusi Upaya**

- Masa hidup sebuah produk perangkat lunak adalah 1 s/d 3 tahun dalam pengembangan dan 5 s/d 15 tahun dalam pemakaiannya (pemeliharannya).
- Distribusi upaya antara pengembangan dan pemeliharaan bervariasi antara 40/60, 30/70, dan bahkan 10/90.
- Tiga aktivitas pengembangan perangkat lunak adalah : analisa dan perancangan, implementasi dan pengujian.
- Tiga aktivitas pemeliharaan perangkat lunak adalah : peningkatan kemampuan produk, penyesuaian produk dengan lingkungan pemrosesan baru, dan perbaikan.

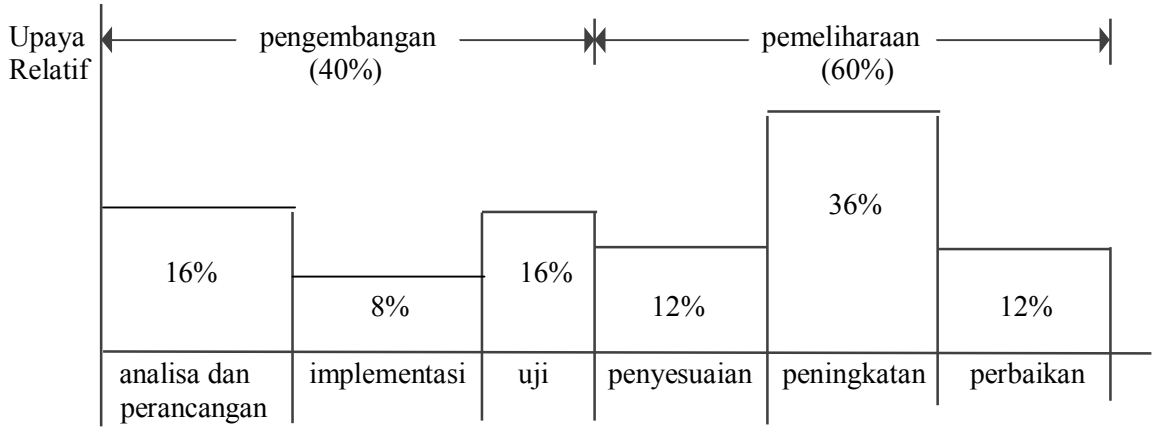


Diagram distribusi upaya dalam putaran hidup sebuah perangkat lunak (*software life cycle, SLC*)

♦ **Katagori Ukuran Proyek**

Katagori	Jumlah programmer	Lama pengerjaan	Jumlah Baris	Contoh proyek
Trivial	1	1-4 minggu	500	keperluan pribadi seorang programmer
Kecil	1	1-6 bulan	1K-2K	penyelesaian numerik masalah sains
Menengah	2-5	1-2 tahun	5K-50K	*compiler* berukuran tidak terlalu besar
Besar	5-20	2-3 tahun	50K-100K	paket *data base*
Sangat Besar	100-1K	4-5 tahun	1M	sistem operasi besar
Extra Besar	2K-5K	5-10 tahun	1M-10M	sistem pertahanan balistik

♦ **Penggunaan Waktu Seorang Programmer**

Tabel aktivitas programmer (Bell Lab. Study, sampel : 70 programmer)

Aktivitas	% Waktu	Keterangan
Penulisan program	13	waktu efektif
Membaca program dan membuat manual	16	upaya perbaikan atas kegagalan (48%)
Mengkomunikasi pekerjaan	32	
Kegiatan pribadi (libur, sakit, dsb)	13	waktu *overhead* (39%)
Lain-lain (kamar kecil, telefon pribadi, rehat kopi, dsb)	15	
Pelatihan	6	
Surat menyurat	5	

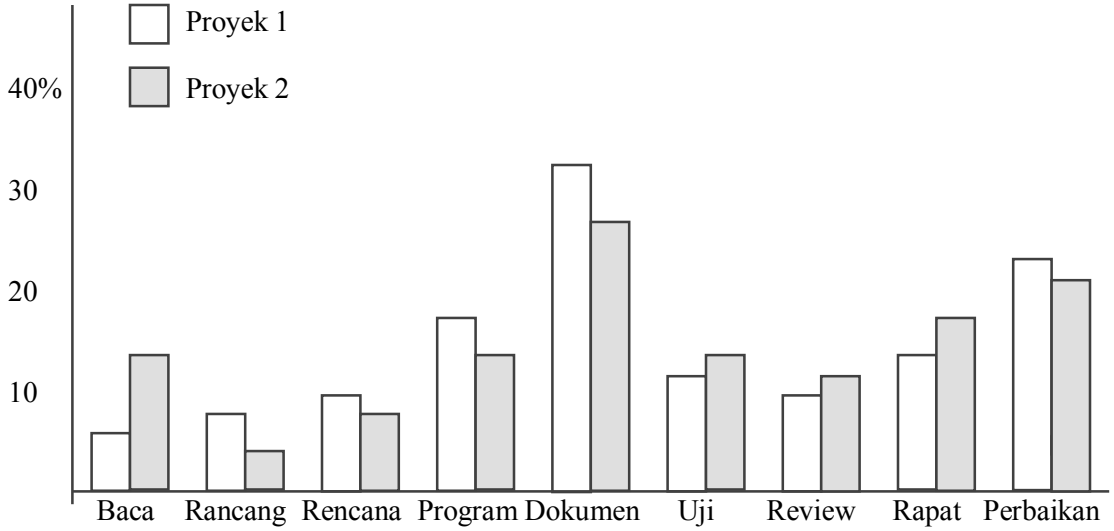


Diagram distribusi aktivitas programer untuk dua buah proyek perangkat lunak. Perhatikan bahwa sekitar 40% aktivitas adalah : baca, review, rapat, dan perbaikan

♦ **Faktor Yang Mempengaruhi Kualitas Produk dan Produktivitas Programmer**

- Kemampuan pribadi :
  - dua aspek dasar kemampuan : kecakapan umum dan terbiasa dengan aplikasi tertentu.
  - seorang yang cakap dalam pemrograman belum tentu cakap pula dalam aplikasi sains, atau sebaliknya.
  - ketidakakraban dengan lapangan aplikasi akan menghasilkan produktivitas rendah dan kualitas yang buruk.
  - yang dimaksud dengan kecakapan umum adalah kemampuan dasar dalam menulis program komputer dengan benar sedangkan ukuran produktivitas seorang programmer adalah banyak baris yang dihasilkan oleh programmer tersebut per hari.
- Komunikasi team
  - meningkatnya ukuran produk yang dihasilkan akan menurunkan produktivitas programmer akibat meningkatnya kerumitan antara komponen-komponen program dan akibat meningkatnya komunikasi yang perlu dilakukan antara programmer, manajer, dan pelanggan.
  - jumlah lintasan komunikasi antar programmer yang terjadi dalam sebuah proyek adalah n(n-1)/2, dimana n adalah jumlah programmer yang terlibat dalam proyek tersebut.
  - penambahan lebih banyak programmer dalam sebuah proyek yang sedang berjalan akan menurunkan produktivitas, kecuali jika para programmer baru tersebut mempunyai tugas yang tidak bergantung kepada hasil kerja programmer lama.
  - Hukum Brooks : *Adding more programmers to a late project may make it later*.

- Kerumitan produk
  - tiga level kerumitan produk : program aplikasi, program *utility*, program level sistem.

level produk	produktivitas relatif terhadap level sistem	contoh produk	bahasa pemrograman
aplikasi	25 s/d 100	pengolahan data	Pascal, Cobol
utility	5 s/d 10	*compiler*	Pascal, C
sistem	1	sistem operasi	bahasa rakitan

- Notasi yang tepat
  - bahasa pemrograman menetapkan notasi (baca : token, *reserve word*) baku, terutama untuk hal-hal yang berkaitan dengan matematika.
  - penetapan notasi antar programer (baca : perancang produk) harus dilakukan sehingga dapat dimengerti dengan jelas.
- Pendekatan sistematis
  - sistem menetapkan teknik dan prosedur baku.
  - pembakuan dalam pengembangan dan pemeliharaan perangkat lunak masih belum mantap.
- Kendali perubahan
  - kelenturan sebuah produk perangkat lunak merupakan sebuah kekutan, tetapi di pihak lain juga merupakan sumber kesulitan dalam proses perancangannya.
  - perubahan terhadap produk harus tetap meminta persetujuan manajer sebagai penanggung jawab proyek.
  - dampak perubahan harus dapat ditelusuri, diuji, dan didokumentasikan.
- Tingkat teknologi
  - peran penggunaan teknologi dalam proyek perangkat lunak misalnya menyangkut bahasa pemrograman, lingkungan mesin yang digunakan, teknik pemrograman, dan penggunaan *tools* tertentu.
  - bahasa pemrograman modern menyediakan fasilitas penyesuaian pendefinisisan dan penggunaan data, konstruksi aliran kendali, fasilitas modular, dan *concurent programming*
- Tingkat keandalan
  - setiap produk harus mempunyai keandalan standar.
  - peningkatan keandalan dihasilkan melalui perhatian yang sangat besar pada tahap analisa.
  - peningkatan keandalan akan menurunkan produktivitas.
  - Boehm : rasio produktivitas antara dua produk dengan keandalan terendah dengan yang tertinggi adalah 2:1.
- Pemahaman permasalahan
  - pelanggan adalah penyumbang utama terhadap kegagalan dalam memahami masalah adalah : (1) tidak memahami permasalahan perusahaannya, (2) tidak mengerti kemampuan dan keterbatasan komputer, (3) tidak mempunyai pengetahuan dasar tentang logika dan algoritma.
  - *software engineer* tidak memahami lapangan aplikasi, gagal mendapatkan informasi kebutuhan pelnggan karena pelanggan bukan seorang *end user*.

- Ketersediaan waktu
  - penetapan lama proyek dan jumlah programmer terlibat harus mempertimbangkan kemampuan pribadi setiap programmer serta kemapuan komunikasi antar mereka.
  - jumlah programmer yang makin banyak akan meningkatkan *overhead* di antaranya akibat keperluan komunikasi.
  - jumlah programmer yang makin sedikit berarti memperbanyak beban kerja kepada setiap programmer.
  - proyek 1 bulan dengan 6 programmer bisa saja diganti dengan proyek 6 bulan dengan 1 programmer ata proyek 3 bulan dengan 3 programmer.
- Persyaratan keterampilan
  - berbagai keterampilan harus ada dalam sebuah proyek perangkat lunak, misalnya : (1) keterampilan berkomunikasi dengan pelanggan untuk memastikan keinginannya dengan sejelas-jelasnya, (2) kemampuan dalam pendefinisian masalah dan perancangan, (3) kemampuan implementasi dengan penulisan program yang benar, (4) kemampuan *debugging* secara deduktif dengan kerangka "*what if* ", (5) dokumentasi, (6) kemampuan bekerja dengan pelanggan.
  - semua keterampilan tersebut harus senantiasa dilatih.
- Fasilitas dan sumber daya
  Fasilitas non teknis yang tetap perlu diperhatikan yang berkaitan dengan motivasi programmer misalnya : mesin yang baik, serta tempat yang tenang, atau ruang kerjanya dapat ditata secara pribadi.
- Pelatihan yang cukup
  Banyak programmer yang dilati dalam bidang-bidang : ilmu komputer, teknik elektro, akuntansi, matematika, tetapi jarang yang mendapat pelatihan dalam bidang teknik perangkat lunak.
- Kemampuan manajemen
  - seringkali manajer proyek tidak mempunyai, atau hanya sedikit mengetahui, latar belakang teknik perangkat lunak.
  - di sisi lain terjadi promosi jabatan menjadi manajer dimana yang berpromosi tidak atau kurang mempunyai kemampuan manajemen.
- Sasaran yang tepat
  Sasaran utama dari tekni perangkat lunak adalah pengembangan produk-produk perangkat lunak yang tepat untuk digunakan.
- Peningkatan kualitas
  Dua aspek yang menimbulkan keinginan untuk meningkatkan kualitas produk adalah : (1) seberapa banyak fungsi, keandalan, dan kemampuan dapat diberikan melalui sejumlah pengembangan, (2) masalah mendasar dariketerbatasan teknologi perangkat lunak.
- Faktor lain
  Faktor lain di antaranya adalah : keakraban, akses, dan stabilitas terhadap sistem komputer dalam mengembangkan atau memodifikasi perangkat lunak.

♦ **Masalah Manajerial**

- Sejumlah masalah yang seringkali muncul :
  1. buruknya perencanaan proyek
  2. buruknya prosedur dan teknik pemilihan manajer proyek
  3. tidak diketahui dengan tepat siapa penanggung jawab buruknya hasil proyek
  4. buruknya kemampuan untuk menetapkan sumber daya yang dibutuhkan proyek
  5. kriteria keberhasilan proyek tidak tepat
  6. buruknya struktur organisasi proyek
  7. pemilihan teknik menajemen yang tidak tepat
  8. tidak diterapkan prosedur, metoda, dan teknik dalam perancangan sistem kendali yang memungkinkan manajer dapat mengontrol proyeknya dengan baik
  9. tidak digunakannya prosedur, teknik, dan strategi yang dapat menyediakan kejelasan perkembangan proyek
  10. standar dan teknik untuk mengukur kualitas kerja dan produktivitas programmer dan analis pengolahan data tidak dijalankan

- Beberapa solusi yang dapat dilakukan :
  1. lakukan pendidikan dan pelatihan terhadap menejen puncak dan manajer proyek
  2. laksanakan penggunaaan standar, prosedur, dan pendokumentasian
  3. lakukan analisa data dari proyek sebelumnya untuk menentukan metda efektif
  4. tetapkan tujuan dengan menyatakan kualitas yang diinginkan
  5. tetapkan kualitas dengan acuan standar peluncuran produk
  6. tetapkan kriteria keberhasilan proyek
  7. siapkan rencana darurat
  8. kembangkan kejujuran, akurasi biaya, dan perkiraan jadwal proyek yang dapat diterima manajemen dan pelanggan
  9. pilih manajer proyek berdasarkan kemampuannya memimpin proyek, bukan berdasarkan kemampuan teknik atau (apalgi) memang tidak ada orang lain lagi
  10. buat penetapan kerja yang spesifik untuk setiap pelaksana dan tetapkan standar kerja.

REKAYASA PERANGKAT LUNAK

Oleh : Dr. Asep Juarna

(Sumber: Richard Fairley. *Software Engineering Concepts*. McGraw-Hill Int'l Ed.)

Bab II. Perencanaan Proyek Perangkat Lunak

3 langkah perencanaan : pendefinisian masalah, pengembangan strategi solusi, rencana proses pengembangan.

A. Pendefinisian Masalah

1. Nyatakan masalah yang akan diselesaikan secara tegas, termasuk di dalamnya batasan masalah dan sasaran yang ingin dicapai. Pernyataan masalah ditetapkan dalam sudut pandang pelanggan.
   - pernyataan masalah dalam sudut pelanggan misalnya : masalah penggajian, masalah *inventory*, atau masalah pengendalian lalu lintas udara
   - pernyataan masalah dalam sudut pengembang misalnya : masalah *relational data bases*, masalah algoritma *sorting*.
   - Teknik-teknik yang digunakan untuk mendapatkan informasi kebutuhan pelanggan meliputi : wawancara dengan pelanggan, pengamatan terhadap gugus tugas yang bermasalah, kinerja sebenarnya dari gugus tugas.
2. Rancang sebuah strategi solusi berbasis komputer.
   - Solusi harus ekonomis dan dapat diterima secara sosial maupun secara politik.
   - Solusi yang ekonomis adalah sistem komputerisasi yang memberikan pelayanan dan informasi yang sama dengan sistem lama tetapi membutuhkan waktu dan personal yang lebih sedikit dalam pengoperasiannya.
   - Sistem baru mungkin akan mengurangi keterlibatan personal; hal ini mungkin akan menimbulkan dampak sosial, bahkan politik.
3. Identifikasi sumber daya yang tersedia.
   - Tiga subsistem dalam sistem komputerisasi adalah : perangkat keras, perangkat lunak, dan personal. Identifikasi juga keterkaitan antar ketiga subsistem tersebut.
   - Subsistem perangkat keras meliputi perangkat keras beserta periferalnya, dan dalam beberapa kasus juga meliputi peralatan lain seperti sensor kendali proses, antena, dan radar.
   - Subsistem perangkat lunak meliputi perangkat lunak yang akan dikembangkan ditambah dengan perangkat lunak yang ada yang boleh jadi digunakan seadanya atau dalam versi modifikasinya.
   - Subsistem personal meliputi para operator, pemelihara, dan *end user*.
4. Tetapkan sasaran dan persyaratan, baik untuk proses pengembangan maupun produk.
   - Sasaran adalah tujuan yang ingin dicapai. Sasaran digunakan sebagai dasar bagi kerangka kerja proyek pengembangan perangkat lunak, baik dalam proses pengembangan maupun untuk produk kerja.
   - Sasaran dapat dinyatakan secara kualitatif maupun kuantitatif. Contoh :
     - ♦ proses (kualitatif) : harus meningkatkan keterampilan personal
     - ♦ proses (kuantitatif) : sistem harus selesai dalam 12 bulan
     - ♦ produk (kualitatif) : sistem harus membuat pekerjaan *user* maikin menarik

- ♦ produk (kuantitatif) : sistem harus mengurangi biaya transaksi sebesar 25%.
- Persyaratan menetapkan spesifikasi kemampuan sistem dalam menyelesaikan masalah.
- Persyaratan mencakup aspek-aspek : fungsional, kinerja, perangkat keras, perangkat lunak, personal, dan pengantarmukaan.
- Kalau memungkinkan, nyatakan persyaratan secara kuantitaif untuk menghindari ketidakjelasan dan perselisihan antara pengembang dengan pelanggan.
- Contoh persyaratan kuantitatif :
  - ♦ akurasi sudut fase harus berada pada kisaran 0.5 derajat
  - ♦ tanggapan maksimum terhadap *interrup* adalah 0.25 detik
  - ♦ *space* maksimum yang digunakan sistem adalah 50 KB memori utama, tidak termasuk *space* untuk file-file *buffer*
  - ♦ sistem harus dapat beroperasi dengan kemampuan 95% ketika dioperasikan selama 24 jam penuh
- Contoh persyaratan kualitatif :
  - ♦ akurasi harus cukup tinggi
  - ♦ sistem harus mempunyai tanggapan yang baik
  - ♦ sistem harus hemat dalam penggunaan memori utama
  - ♦ keandalan sistem harus 99%
- Sasaran dan persyaratan dapat juga ditetapkan melalui atribut-atribut kualitas yang harus dimiliki sistem, di antaranya : *portability* (S/W tidak bergantung mesin), *realiability* (kemampuan program melakukan fungsi yang diinginkan), *efficiency* (menggunakan sumber daya minimal), *accuracy* (ukuran besarnya *error*), *robustness/integrity* (kemampuan bekerja dengan baik walau mendapat input yang tidak benar), *correctness* (hasil sesuai dengan yang diharapkan).

5. Tetapkan kriteria penerimaan sebuah sistem
   - Kriteria harus dinyatakan sedemikian rupa sehingga tidak akan menimbulkan perselisihan antara pengembang dan pelanggan. Kriteria harus dapat diverfikasi dengan suatu metoda baku seperti : peninjauan langsung, analisa, atau serangkaian uji, terhadap produk yang dihasilkan.

B. Pengembangan Strategi Solusi

- Kecenderungan untuk menerima begitu saja solusi pertama yang terlintas di benak kita adalah masalah utama dalam perkeayasaan perangkat lunak. Ini tidak memberi peluang terhadap solusi lain yang sebenarnya masih mungkin untuk dipertimbangkan.
- Kembangkan strategi solusi. Strategi bukan merupakan solusi rinci tetapi penyataan umum tentang sifat-sifat dari solusi yang mungkin.
- Langkah-langkah pengembangan strategi solusi adalah sebagai berikut :

1. Uraikan beberapa strategi solusi tanpa memperhatikan batasan-batasan apapun
2. Adakan studi kelayakan terhadap setiap strategi. Perhatikan bahwa *an unreasonable idea will lead to other ideas, some of which may be very reasonable*.
3. Rekomendasikan sebuah strategi solusi, beri catatan mengapa solusi lain ditolak
4. Buat sebuah daftar prioritas karakteristik produk. Daftar ini penting jika kondisi tidak memungkinkan untuk mengimplementasikan seluruh kemampuan produk yang diinginkan seperti yang telah ditentukan sebelumnya.

C. Perencanaan Proses Pengembangan

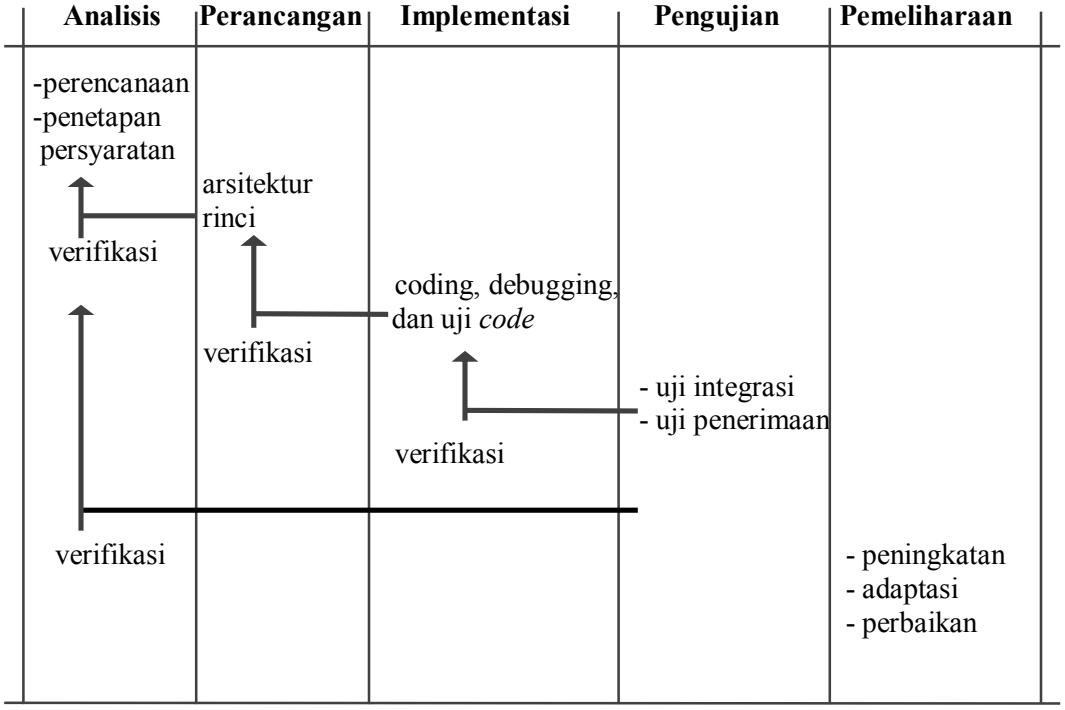
1. Tentukan sebuah model *life-cycle* dan struktur organisasi proyek.
2. Rencanakan konfigurasi managemen, jaminan kualitas, dan kegiatan validasi
3. Tentukan tools setiap fase proyek, serta teknik-teknik dan notasi yang digunakan
4. Tetapkan perkiraan biaya untuk pengembangan sistem
5. Tetapkan jadwal pengembangan
6. Tetapkan perkiraan susunan personalia proyek
7. Tetapkan perkiraan sumber daya sistem komputaerisasi yang diperlukan untuk mengoperasikan dan memelihara sistem
8. Siapkan daftar istilah
9. Identifikasi sumber-sumber informasi dan jadikan sebagai acuan proyek

Life Cycle

Life-cycle sebuah perangkat lunak mencakup semua kegiatan yang yang perlu dilakukan untuk mendefinisikan, mengembangkan, menguji, mengantarkan, mengoperasikan, dan memelihara produk perangkat lunak. Beberapa model yang akan dibahas adalah : model fase (*phased model*), model biaya (*cost model*), model prototipe (*prototype model*), dan model berurutan (*successive model*).

Model Fase

Model ini membagi *life cycle* ke dalam sederetan kegiatan (fase). Setiap fase membutuhkan informasi masukan, proses, dan produk yang terdefinisi dengan baik. Deretan fase tersebut adalah : analisa, perancangan, implementasi, pengujian, dan pemeliharaan. Berikut ini model fase dasar yang dinyatakan sebagai *waterfall chart* :



Life cycle mode fase dari sebuah perangkat lunak

- Subfase perencanaan menghasilkan dua produk : *System Definition* dan *Poject Plan*. Format kedua produk adalah sebagai berikut :

Format *System Definition*
Bab 1 : Pendefinisian masalah
Bab 2 : Justifikasi sistem
Bab 3 : Sasaran sistem dan proyek
Bab 4 : Batasan sistem dan proyek
Bab 5 : Fungsi yang harus disiapkan (H/W, S/W, personal)
Bab 6 : Karakteristik pengguna
Bab 7 : Lingkungan pengembangan/ operasi/pemeliharaan
Bab 8 : Strategi solusi
Bab 9 : Prioritas gambaran sistem
Bab 10: Kriteria penerimaan sistem
Bab 11: Sumber informasi
Bab 12: Daftar istilah

Format *Project Plan*
Bab 1 : Model *life cycle* : terminologi, tonggak penting, produk kerja
Bab 2 : Struktur organisasi : struktur manajemen/ struktur team, gambaran kerja
Bab 3 : Perkiraan personal & persyaratan sumber daya
Bab 4 : Jadwal awal pengembangan
Bab 5 : Perkiraan awal biaya
Bab 6 : Pengawasan proyek dan mekanisme kontrol
Bab 7 : Alat bantu dan teknik yang digunakan
Bab 8 : Bahasa pemrograman
Bab 9 : Persyaratan pengujian
Bab 10: Dokumen pendukung yang diperlukan
Bab 11: Cara demonstrasi
Bab 12: Jadwal dan materi pelatihan
Bab 13: Rencana pemasangan (instalasi)
Bab 14: Pokok perhatian dalam pemeliharaan
Bab 15: Metoda dan waktu pengantaran
Bab 16: Metoda dan waktu pembayaran
Bab 17: Sumber informasi

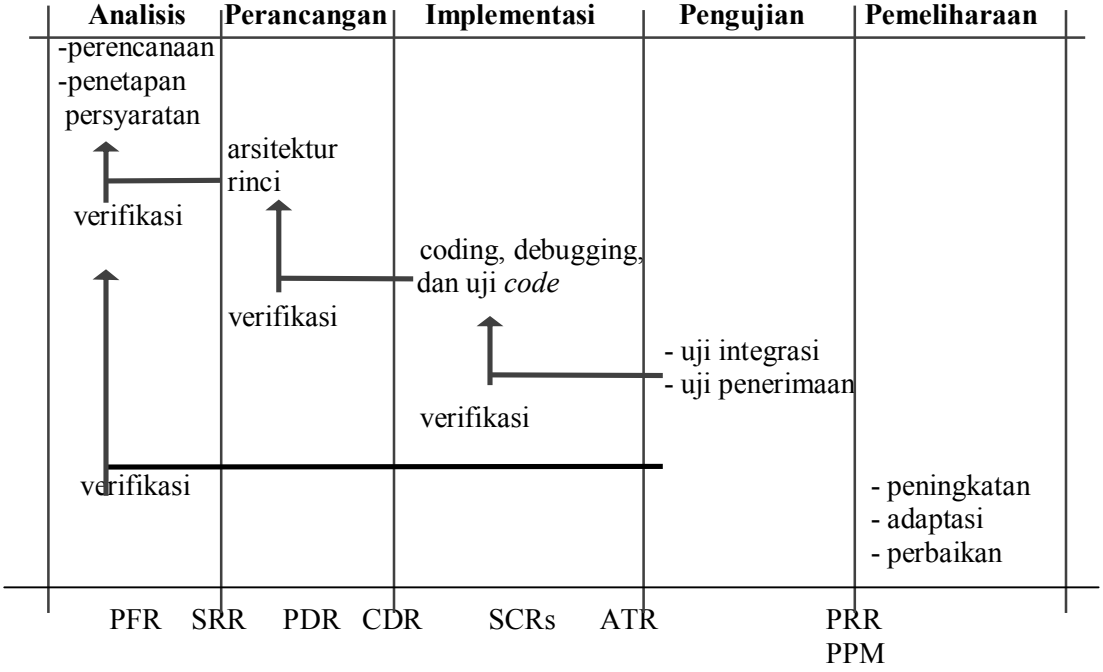
- Subfase penetapan persyaratan menghasilkan sebuah produk : *Software Requirements Specifications*. Format produk ini adalah sbb :

Format *Software Requirements Specifications*
Bab 1 : Gambaran dan penjelasan ringkasan produk
Bab 2 : Lingkungan pengembangan, pengoperasian, dan pemeliharaan
Bab 3 : Pengantarmukaan eksternal dan aliran data : format tampilan, *user command*, DFD, kamus data
Bab 4 : Persyaratan fungsional : fungsi-fungsi yang diinginkan
Bab 5 : Persyaratan kinerja : tanggapan, waktu proses
Bab 6 : Penanganan kesalahan : aksi dan pesan yang harus dilakukan sebagai tanggapan atas input atau situasia yang tidak dikehendaki produk
Bab 7 : Subset permulaan dan prioritas implementasi : 'versi' awal produk
Bab 8 : Perkiraan modifikasi dan peningkatan
Bab 9 : Kriteria penerimaan
Bab 10 : Petunjuk dan panduan perancangan
Bab 11 : Index acuan
Bab 12 : Daftar istilah

- Fase perancangan melakukan identifikasi terhadap komponen perangkat lunak (fungsi, arus data, penyimpanan data), hubungan antar komponen, struktur perangkat lunak (dekomposisi menjadi modul-modul dan pengatarmukaannya). Fase ini menghasilkan arsitektur rinci, terutama dalam bentuk algoritma-algoritma.
- Fase implementasi adalah terjemahan langsung arsitektur rinci ke dalam bahasa pemrograman tertentu.

- Subfase uji integrasi melakukan pengujian terhadap semua modul dan pengantarmukaan sehingga pada level sistem dapat beroperasi dengan benar
- Subfase uji penerimaan melakukan baerbagi pengujian mengacu kepada berbagai persyaratan yang telah ditentukan.
- Kegiatan yang meliputi fase pemeliharaan adalah : peningkatan kemampuan, adaptasi terhadap lingkungan pemrosesan, dan melakukan berbagai koreksi atas kesalahan yang terjadi

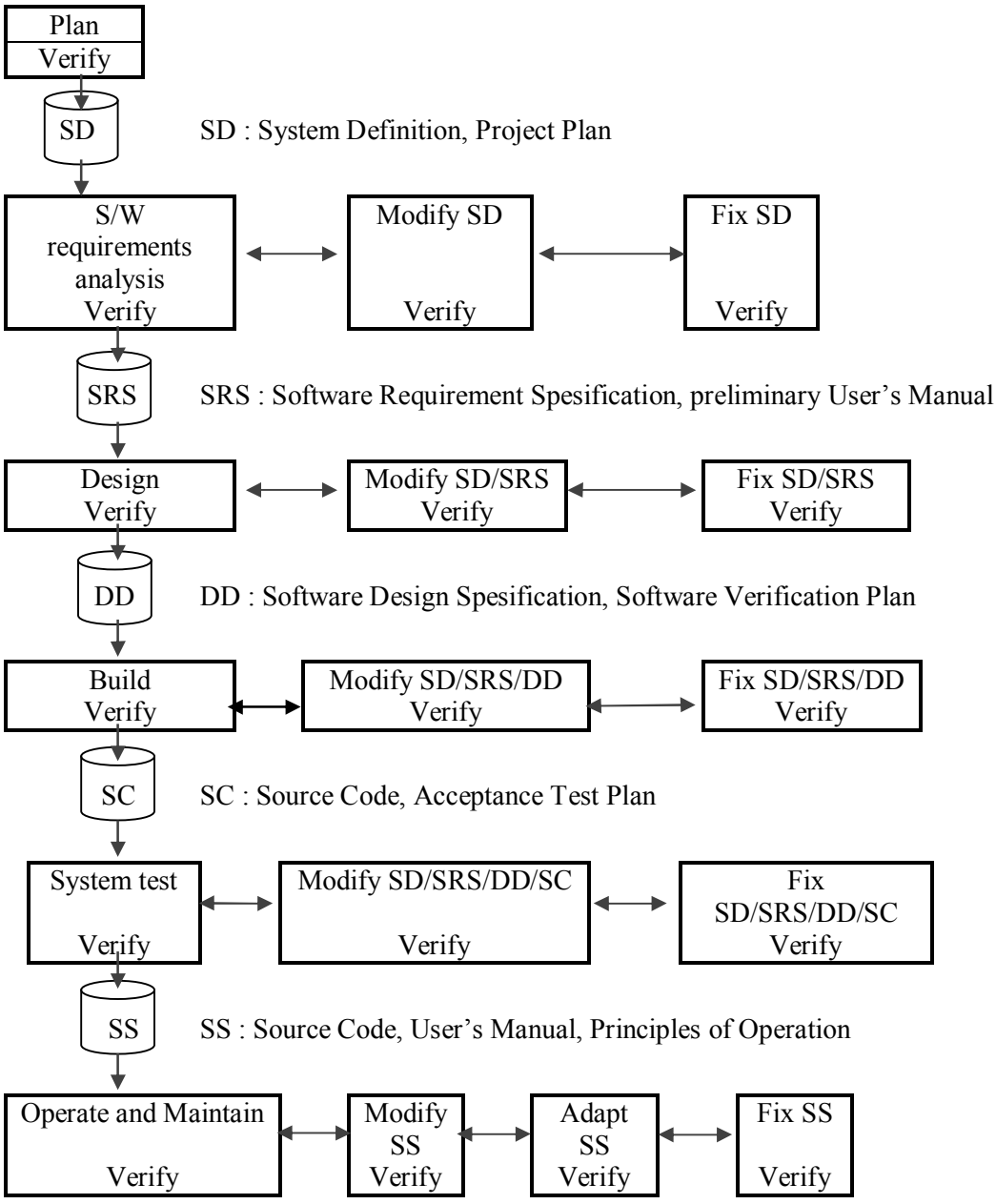
Penilaian kemajuan proyek akan lebih mudah jika pada model fase tersebut ditetapkan beberapa tonggak penting (*milestone*) yang pada setiap fase atau antar setiap dua fase yang berurutan. Berikut ini *Life cycle* mode fase dari sebuah perangkat lunak yang dilengkapi dengan kegiatan *review* dan tonggak penting :



Review	Produk Kerja yang direview
PFR (*Product Feasibility Review*)	*System Definition*
	Project Plan
SRR (*Software Requirements Review*)	Spsesifikasi persyaratan perangkat lunak
	User's Manual awal
	Rencana awal verifikasi
PDR (*Preliminary Design Review*)	Dokumen disain arsitektur
CDR (*Critical Design Review*)	Spesifikasi disain rinci
SCR (*Source Code Review*)	Penelusuran dan Pemeriksaan *source code*
ATR (*Acceptance Test Review*)	Rencana uji penerimaan
PRR (*Product Release Review*)	Semua produk kerja sebelumnya
PPM (*Project Post-Mortem*)	Catatan umum pelaksanaan proyek

Model Biaya

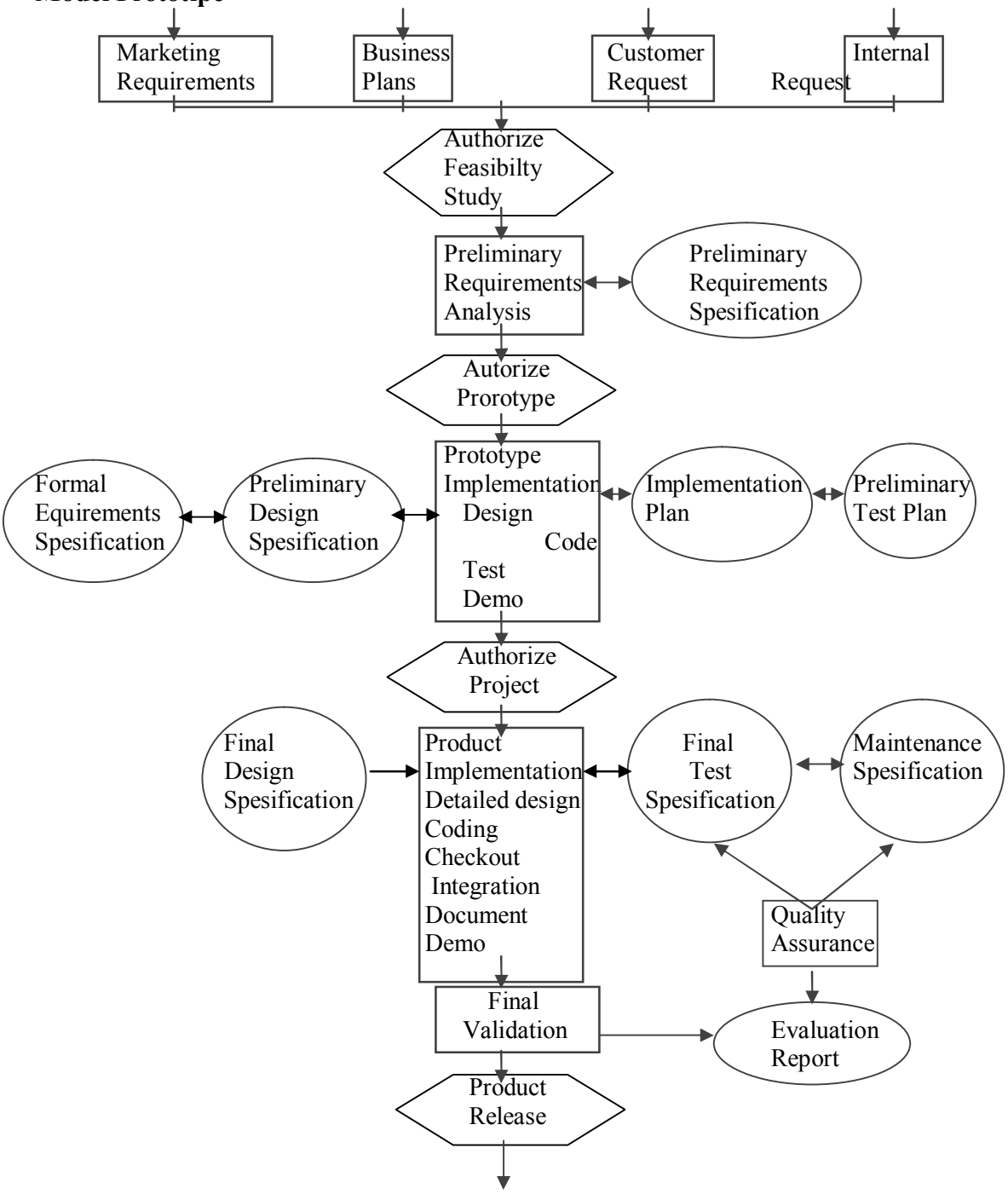
Model biaya adalah cara pandang lain model fase sebuah perangkat lunak dengan cara memperhatikan biaya berbagai kegiatan dalam proyek perangkat lunak. Biaya proyek adalah jumlah biaya dari setiap fase proyek. Biaya setiap fase mencakup biaya kegiatan dan penyiapan produk pada fase tersebut ditambah dengan biaya verifikasi konsistensi produk suatu fase terhadap semua fase sebelumnya.



Ada 2 sisi penting dari model biaya :

- Karena modl biaya hanyalah cara pandang lain dari model fase maka semua dokumen yang dihasilkan tepat sama dengan yang dihasilkan pada model fase.
- Biaya verifikasi, apalagi perbaikan, atas suatu produk akan makin besar jika produk tersebut dihasilkan oleh suatu fase yang jauh di belakang fase saat verifikasi dilakukan.

Model Prototipe

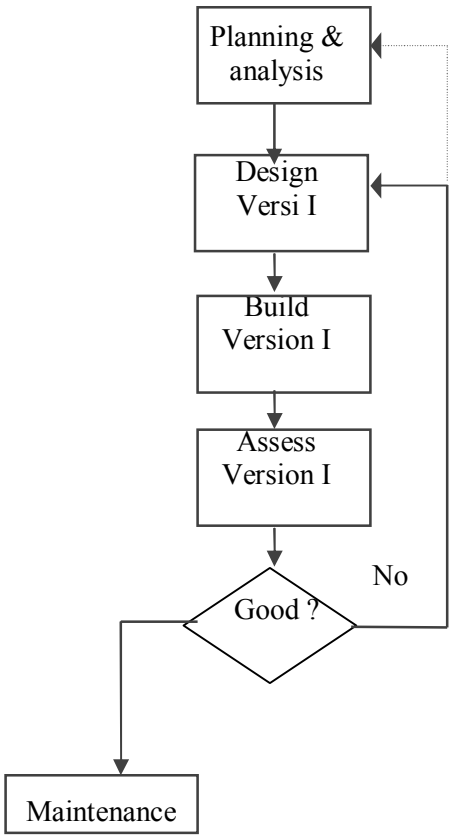


Beberap catatan tentang model prototipe :

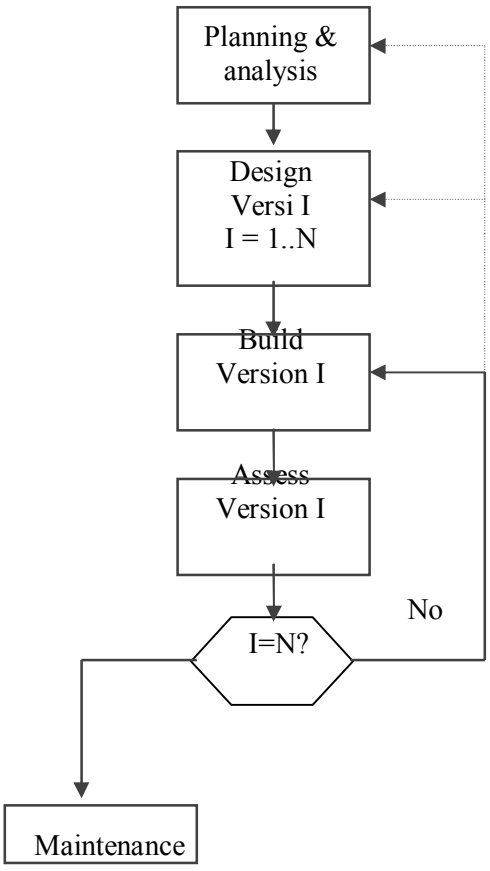
- Sebuah prototipe adalah model dari sebuah produk perangkat lunak tetapi dengan beberapa keterbatasan, misalnya : keterbatasan kemampuan, keandalan yang rendah, dan kinerja yang tidak efisien.
- Alasan penggunaan model prorotipe adalah :

1. untuk menggambarkan format data masukan, pesan-pesan, laporan, dan dialog interaktif
2. untuk mengeksplorasi isu-isu teknis dalam produk yang diusulkan
3. model fase 'analisis → perancangan → implementasi' tidak dapat digunakan

Versi Succesive



Perancangan dan implementasi model berurutan



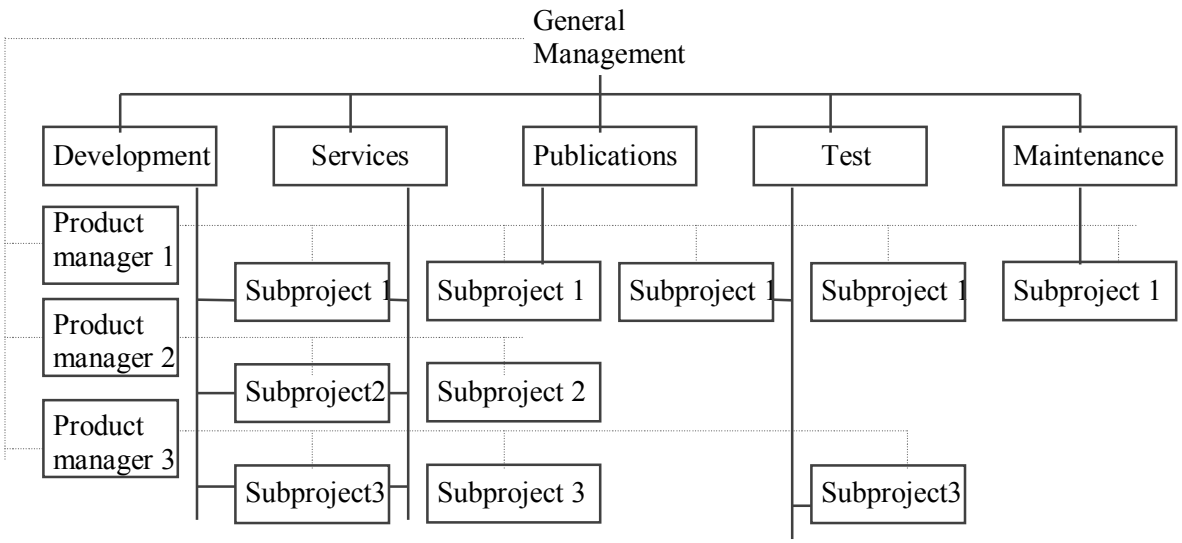
Analisa dan perancangan yang diikuti implementasi dari model berurutan

Perencanaan Struktur Organisasi

Struktur Pelaksana Proyek

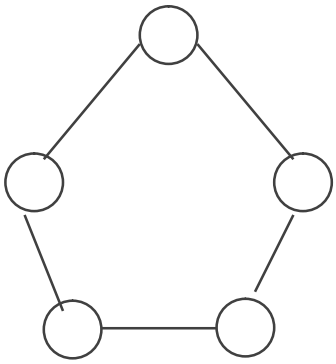
Ada 3 format struktur pelaksana proyek : format proyek, format fungsional, dan formta matriks

1. Format Proyek
   - Dibentuk sebuah team yang melakukan pekerjaan proyek dari awal sampai akhir
   - Annggota team mendefinisikan produk, merancang produk, mengimplementasikan, melakukan uji, melakukan review, dan mempersiapkan dokumen-dokumen pendukung
   - Sebagian anggota team melakukan instalasi dan pemeliharaan, dan melanjutkan ke proyek baru
   - Team proyek biasanya bekerja selama 1-3 tahun dan ditugaskan untuk proyek berikutnya jika proyek pertama sudah selesai
2. Format Fungsional
   - Dalam format ini dibentuk beberapa team untuk melaksanakan pekerjaan proyek setiap fase. Semua team tidak dibentuk pada saat yang sama. Anggota team dapat dirotasi.
   - Team analisis dan perancangan bertugas untuk mengembangkan *System Definition* (SD) dan *Project Plan* (PP).
   - Team pendefinisian produk menerima produk SD dan PP, melakukan analisa persya-ratan perangkat lunak, dan menyiapkan *Software Requirement Specification* (SRS).
   - Team perancangan merancang produk yang sesuai dengan SRS dan SD.
   - Team implementasi mengimplementasi, *debugging*, dan melakukan uji per unit sistem
   - Team uji sistem melakukan uji integrasi
   - Team kualitas melakukan sertifikasi terhadap semua produk kerja
   - Team pemelihara melakukan pemeliharaan produk
3. Format matriks
   - setiap gugus fungsional memiliki team manajemen dan kelompok spsialis yang hanya melaksanakan fungsinya sendiri.
   -

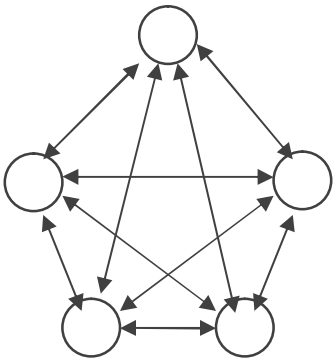


Struktur Tim Pemrograman

Demokrasi



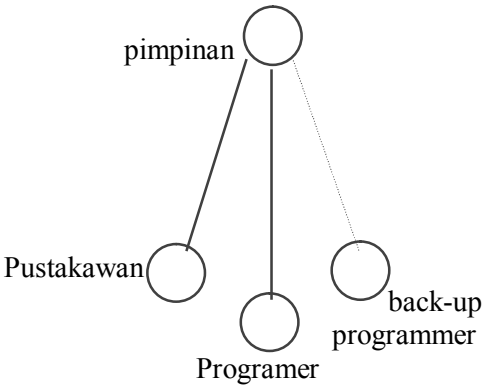
Struktur



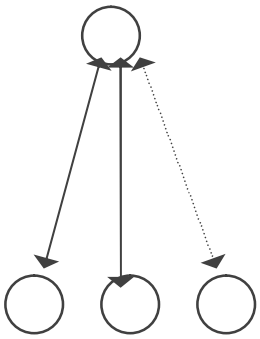
Jalur komunikasi

- Berbagai keputusan dilakukan melalui kesepakatan kelompok
- Kepemimpinan berotasi sesuai dengan jenis pekerjaan yang sedang dilaksanakan dan spesialisasi anggota

Kepemimpinan



Struktur



Jalur Komunikasi

- pimpinan merancang produk, mengimplementasikan bagian kritis dari produk, dan membuat semua keputusan teknis utama
- programer menuliskan *source code*, *debugging*, dokumentasi, dan uji unit
- pustakawan mengurus *listing program*, merancang dokment-dokumen, membuat rencana uji

back-up programmer berperan sebagai konsultan bagi pimpinan untuk berbagai masalah teknis, memelihara hubungan dengan pelanggan dan kelompok publikasi serta kelompok jaminan kualitas, dan melakukan sejumlah analisis-perancangan-implementasi di bawah pengawasan pimpinan.

PERANCANGAN PERANGKAT LUNAK

Oleh : Dr. Asep Juarna

(Sumber: Richard Fairley. *Software Engineering Concepts*. McGraw-Hill Int'l Ed.)

Bab III. Perkiraan Biaya Perangkat Lunak

3.1. Faktor-faktor yang mempengaruhi perkiraan biaya

- Sulit untuk menentukan perkiraan biaya secara akurat selama fase perencanaan pengembangan S/W karena terlalu banyaknya faktor yang tidak diketahui pada saat itu.
- Perkiraan awal disiapkan selama fase perencanaan dan dikemukakan pada saat presentasi kelayakan proyek. Perbaikan dikemukakan pada saat presentasi persyaratan S/W, dan perkiraan akhir dikemukakan pada saat presentasi perancangan awal.
- Faktor-faktor utama yang mempengaruhi biaya perangkat lunak : (1) kemampuan programmer, (2) kompleksitas produk, (3) ukuran produk, (3) waktu yang tersedia, (4) keandalan yang diperlukan, (5) tingkat teknologi.

1. Kemampuan Programmer

Programmer dengan produktivitas tinggi ekuivalen dengan biaya yang kecil.

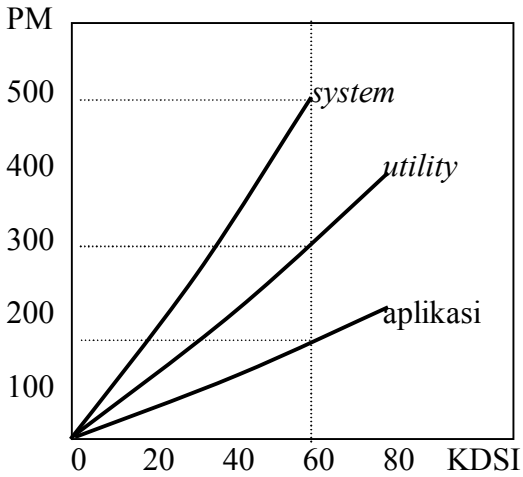
2. Komplekasitas Produk

- Tiga katagori produk : aplikasi, *utility*, dan *system*
- Rasio kompleksitasi ketiganya adalah : aplikasi : *utility* : *system* = 1 : 3 : 9
- Jika PM adalah total upaya (dalam *programmer-months*) dan KDSI adalah baris instruksi dalam product (*thousands of delivered source instructions*) maka estimator PM menurut Boehm adalah :

  aplikasi : $PM = 2.4 \times (KDSI)^{1.05}$

  utility : $PM = 3.0 \times (KDSI)^{1.12}$

  system : $PM = 3.6 \times (KDSI)^{1.20}$
- Untuk jumlah baris program 60.000, rasio PM aplikasi : *utility* : *system* ≈ 1 : 1.7 : 2.8 (tepatnya 176.6 : 294.2 : 489.6). (*Lihat gambar*)



Estimasi upaya COCOMO

- Jika TDEV adalah waktu (dalam bulan) pengembangan sebuah program (*develop-ment time*), maka estimator TEDV menurut Boehm TDEV adalah :

  aplikasi : $TDEV = 2.5 \times (PM)^{0.38}$

  utility : $TDEV = 2.5 \times (PM)^{0.35}$

  system : $TDEV = 2.4 \times (PM)^{0.32}$
- Untuk jumlah baris program 60.000, ketiga program memerlukan waktu pengembangan sebagai berikut : 17.9 (aplikasi), 18.3 (*utility*), dan 17.4 (*system*)

- Rata-rata jumlah programmer yang diperlukan untuk membuat 60 KDSI per bulan adalah :
  aplikasi : 176.6 PM / 17.9 Months = 9.9 programmer
  utility : 294.2 PM / 18.3 Months = 16.1 programmer
  system : 489.6 PM / 17.4 Months = 28.1 programmer

3. Ukuran Produk

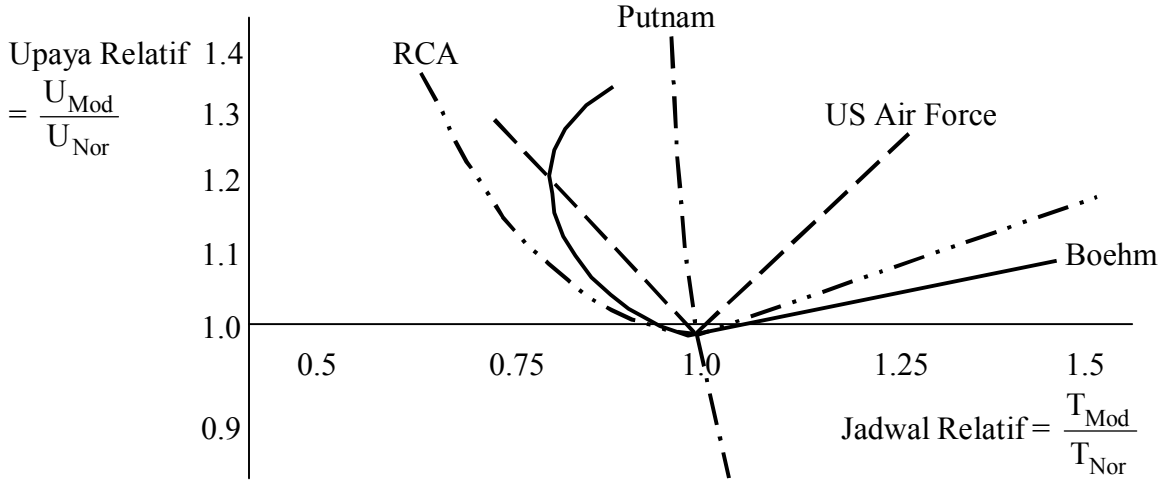
- Beberapa persamaan lain yang menyatakan upaya (PM) dan jadwal (TDEV) berdasarkan berbagai penelitian tersaji pada tabel berikut :

Upaya	Jadwal	Author
$PM = 5.2 \times (KDSI)^{0.91}$	$TDEV = 2.47 \times (PM)^{0.35}$	Waltson
$PM = 4.9 \times (KDSI)^{0.98}$	$TDEV = 3.04 \times (PM)^{0.36}$	Nelson
$PM = 1.5 \times (KDSI)^{1.02}$	$TDEV = 4.38 \times (PM)^{0.25}$	Freburger
$PM = 2.4 \times (KDSI)^{1.05}$	$TDEV = 2.50 \times (PM)^{0.38}$	Boehm
$PM = 3.0 \times (KDSI)^{1.12}$	$TDEV = 2.5 \times (PM)^{0.35}$	Boehm
$PM = 3.6 \times (KDSI)^{1.20}$	$TDEV = 2.4 \times (PM)^{0.32}$	Boehm
$PM = 1.0 \times (KDSI)^{1.40}$	–	Jones
$PM = 0.7 \times (KDSI)^{1.50}$	–	Halstead
$PM = 28 \times (KDSI)^{1.83}$	–	Schneider

- Berdasarkan persamaan milik Boehm, $PM = 2.4 \times (KDSI)^{1.05}$, upaya akan meningkat faktor 2.07 jika ukuran produk (KDSI) diperbesar 2 kali, dan naik dengan faktor 11.22 jika ukuran produk diperbesar 10 kali. Ini menunjukkan bahwa pengembangan produk S/W yang besar akan lebih mahal.
- Peningkatan upaya untuk keseluruhan formula berkisar antara 1.88 s/d 3.56 jika KDSI diperbesar menjadi dua kalinya, dan antara 8.13 s/d 67.61 jika KDSI diperbesar menjadi sepuluh kalinya.

4. Waktu Yang Tersedia

- Berbagai penelitian (kecuali Putnam) menunjukkan bahwa proyek S/W membutuhkan upaya lebih jika waktu pengembangan diperpendek atau diperpanjang (dimodifikasi) dari waktu normal.



- Menurut Putnam upaya proyek S/W berbanding terbalik dengan pangkat empat dari waktu pengembangan, yaitu : $PM = k/((TDEV)^4)$.
- Berdasarkan formula Putnam maka perpanjangan jadwal 2 kali jadwal normal akan menurunkan upaya 100 programmer dalam satu bulan menjadi : $(100/2^4) = 6.25$ programmer. Lebih tidak masuk akal bahwa lagi upaya akan menjadi nol jika jadwal pengembangan menjadi tak hingga.
- Putnam juga mengisyaratkan bahwa kompresi jadwal terbatas sampai 86% jadwal normal, walaupun menurut formulanya kompresi sebesar ini hanya akan menaik-kan jumlah personal sebesar 1.82 kalinya.
- Melalui penelitian terhadap 63 proyek perangkat lunak, Boehm mencatat bahwa hanya 4 proyek yang memungkinkan dilakukannya kompresi jadwal menjdi kurang dari 75% jadwal awal. Berdasarkan penelitian ini Boehm menyatakan bahwa *kompresi terhadap jadwal tidak dapat dilakukan di bawah 75%*.

5. Tingkat Keandalan Yang Diperlukan

Lima katagori keandalan, efek kegagalan, beserta pengali upaya (*effort multiplier*) pengembangan yang direkomendasikan Boehm ditunjukkan melalui tabel berikut :

Katagori	Efek kegagalan	Pengali upaya
Sangat rendah	tidak nyaman digunakan	0.75
Rendah	kesalahan mudah dipulihkan	0.88
Nominal	tidak terlalu sulit memulihkan kesalahan	1.00
Tinggi	kerugian finansial tinggi	1.15
Sangat tinggi	resiko terhadap kehidupan manusia	1.40

6. Tingkat Teknologi

- Tingkat teknologi dalam proyek pengambangan S/W direfleksikan dengan empat komponen yang digunakan, yaitu : (1) bahasa pemrograman, (2) mesin abstrak (H/W dan S/W), (3) praktek-praktek pemrograman, dan (4) *tools*
- Bahasa Pemrograman : Sebaris *statement* program dalam *high level language* identik dengan beberapa baris *statement* dalam *low level language*, dengan demikian penggunaan *high level language* akan menaikkan produktivitas 5 sampai 10 kalinya.
- Mesin Abstrak : Mesin abstrak adalah sekumpulan fasilitas H/W dan S/W yang digunakan selama proses pengembangan. Produktivitas akan buruk jika programmer harus belajar lebih dahulu mesin dan lingkungan baru sebagai bagian dari proses pengembangan.
- Praktek-Praktek Pemrograman : Praktek pemrograman modern meliputi : (1) analisis sistematik dan teknik-teknik perancangan, (2) notasi perancangan terstruktur, (3) pemeriksaan, (4) pengkodean (*coding*) terstruktur, (5) pengujian yang sistematis, dan (6) perpustakaan pengembangan program.
- *Tools* : *Tools* yang berwujud S/W terdiri dari : *assembler* dan *debugger*, *compiler* dan *linker*, DBMS, dan *tools* lain yang lebih *advanced*.
- Pengali upaya dalam praktek pemrograman yang direkomendasikan Boehm bervariasi antara 1.24 (tanpa praktek pemrograman modern) sampai 0.82 (sepenuhnya menggunakan praktek pemrograman modern), antara 1.24 (menggunakan *tools* dasar seperti *compiler* dan *debugger*) sampai 0.83 (menggunakan *tools* pengembangan yang paling *advanced*).

3.2. Teknik-teknik perkiraan biaya S/W

- Perkiraan biaya S/W seringkali bersandar kepada data historis.
- Perkiraan biaya bisa dilakukan secara *top-down* atau *bottom-up*.
- *Top-down* : fokus pertama adalah perkiraan biaya tingkat sistem seperti sumber daya komputasi, personal yang diperlukan, pengaturan konfigurasi perangkat lunak, jaminan kualitas, integrasi sistem, pelatihan, dan publikasi.
  Bottom-up : fokus pertama adalah perkiraan biaya pengembangan setiap modul atau subsistem, sedemikian rupa sehingga sampai pada perkiraan biaya sistem secara keseluruhan.
- Teknik *top-down* sangat baik dalam memandang sistem secara keseluruhan, tetapi kadang-kadang melupakan berbagai masalah teknis pada pengembangan beberapa modul.
- Teknik *bottom-up* menekankan perkiraan biaya setiap modul, tetapi kadang-kadang gagal dalam mengakomodasi perkiraan biaya level sistem seperti pengaturan konfigurasi perangkat lunak dan jaminan kualitas.

1. Pendapat Pakar (Expert Judgment)

- Teknik ini merupakan teknik perkiraan biaya yang sering digunakan yang juga merupakan teknik *top-down*.
- Contoh proses perkiraan seorang pakar adalah sbb.:
  i. Sistem yang akan dikembangkan adalah sebuah *sistem kendali proses* yang mirip dengan sistem yang dikembangkan tahun lalu dengan biaya US$ 1 juta dan waktu 10 bulan.
  ii. Tidak ada pembengkakan biaya (*mark-up*) tetapi mempertimbangkan keuntungan yang cukup baik.
  iii. Diinginkan sistem yang mempunyai fungsi kendali 25% lebih aktif sehingga perkiraan *waktu dan biaya* akan dinaikkan sebesar 25%.
  iv. Dalam pengambangan tersebut digunakan komputer dan peralatan lain serta personal yang sama dengan yang digunakan pada pengembangan sistem sebelumnya sehingga *perkiraan biaya* dapat direduksi sebesar 20%.
  v. Kita dapat memanfaatkan cukup banyak *code* produk sebelumnya sehingga mereduksi *waktu dan biaya* sebesar 25%.
  vi. Berbagai pertimbangan di atas menghasilkan perkiraan sebesar :
  Biaya : 125% × US$ 1 juta × 80% × 75% = US$ 750 ribu
  Waktu : 125% × 10 bulan × 75% = 9.4 bulan
  vii. Nilai proyek diajukan US$ 850 ribu dengan durasi 10 bulan.

2. Estimasi biaya Delphi (dikenal juga sebagai : ***Group Consensus***)

Sangat mungkin pendapat seorang pakar berbeda dengan pendapat pakar lainnya. Untuk itu perlu dilakukan sebuah konsensus. Teknik Delphi adalah teknik yang memungkinkan dilakukannya konsensus ini. Teknik ini termasuk teknik *top-down*, dengan langkah-langkah sebagai berikut :

i. Seorang koordinator menyebarkan dokumen *System Definition* dan formulir yang mencatat nilai usulan perkiraan biaya kepada setiap estimator.
ii. Setap estimator mempelajari *System Definition* dan mengisikan sebuah nilai perkiraan biaya pada formulir yang tersedia, termasuk alasan-alasan atas nilai

perkiraan tersebut, tanpa mencantumkan namanya. Bila perlu setiap estimator dapat mengajukan pertanyaan kepada koordinator tetapi dialog anatar para estimator tidak diperbolehkan.

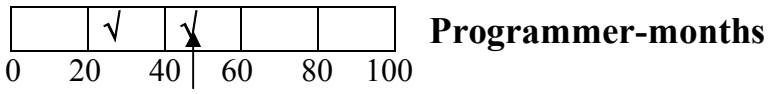
iii. Koordinator merangkum dan mengolah semua angka, menyebarkan form baru yang disertai nilai rangkuman (nilai median perkiraan atau nilai rata-rata perkiraan), serta catatan tentang pertimbangan khusus yang diberikan estimator.

iv. Setiap estimator kembali melakukan langkah (ii). Proses diulang beberapa putaran, tergantung keperluan. Jika terdapat estimator yang memberikan nilai perkiraan yang sangat berbeda dengan para estimator lainnya, dia dapat dimintai penjelasannya.

Berikut ini contoh sebuah formulir perkiraan biaya :

Proyek : Sistem Operasi
Tanggal : 25 Agustus 2000

Selang nilai perkiraan hasil putaran ke-3
Nilai perkiraan anda



Nilai Median
Nilai perkiraan anda untuk putaran berikutnya : 35 PM

Alasan atas nilai perkiraan :
Personal kita sudah berpengalaman dengan sistem seperti ini. Mereka tidak akan menemui masalah. Saya menaikkan nilai perkiraan saya karena mempertimbangkan channel DMA yang disampaikan oleh salah seorang estimator.

3. Model Biaya Algoritmis

- Teknik ini termasuk teknik *bottom-up*. Teknik ini menghitung perkiraan biaya sistem sebagai jumlah dari perkiraan biaya semua modul dan susbsistem yang membangun sistem.
- Salah satu teknik ini yang terkenal adalah *Constructive Cost Model* (*COCOMO*) yang digambarkan oleh Boehm.
- COCOMO menggunakan formula PM dan TDEV (lihat pasal 1). Selanjutnya pengali upaya digunakan untuk koreksi terhadap perkiraan atribut-atribut produk, komputer, personal, dan proyek.
- Berikut ini tabel beberapa pengali upaya COCOMO yang diperoleh dari penelitian 63 proyek perangkat lunak :

Pengali	Selang Nilai
Atribut produk	
Keandalan yang diinginkan	0.75 s/d 1.40
Ukuran *database*	0.94 s/d 1.16
Kompleksitas produk	0.70 s/d 1.65
Atribut komputer	
Batasan waktu eksekusi	1.00 s/d 1.66
Batasan memori utama	1.00 s/d 1.56
Virtual machine volatiliry	0.87 s/d 1.30
Computer turnaround time	0.87 s/d 1.15
Atribut personal	
Kapabilitas analist	1.46 s/d 0.71
Kapabilitas programmer	1.42 s/d 0.70
Pengalaman dengan aplikasi	1.29 s/d 0.82
Pengalaman dengan *virtual machine*	1.21 s/d 0.90
Pengalaman dengan bahasa pemrograman	1.14 s/d 0.95
Atribut proyek	
Penggunaan praktek pemrograman modern	1.24 s/d 0.82
Penggunaan *tools*	1.24 s/d 0.83
Jadwal pengembangan yang diperlukan	1.23 s/d 1.1.0

Pengali upaya untuk contoh *embedded system* telekomunikasi

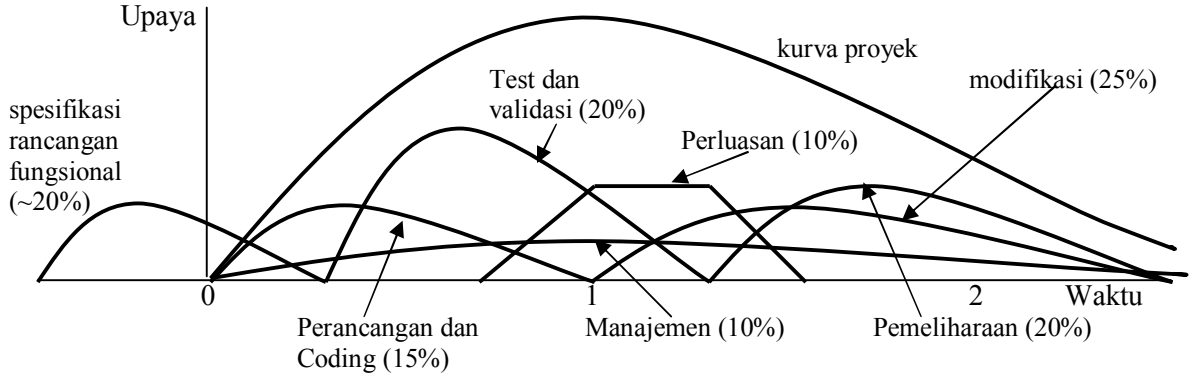
Pengali	Dasar Pemikiran	Nilai
Keandalan	Penggunaan lokal. Tidak ada masalah pemulihan kesalahan yang serius (nominal)	1.00
Database	20.000 byte (rendah)	0.94
Kompleksitas	Pemrosesan tekomunikasi (sangat tinggi)	1.30
Timing	Menggunakan 70% waktu pemrosesan (tinggi)	1.11
Storage	45 K dari 64 K yang tersedia (tinggi)	1.06
Machine	Stabil. mikroprosesor pasaran sudah cukup (nominal)	1.00
Turnaround	Rata-rata dua jam (nominal)	1.00
Analyst	Senior (tinggi)	0.86
Programmer	Senior (tinggi)	0.86
Pengalaman	3 tahun di bidang telekomunikasi (nominal)	1.00
Pengalaman	6 bulan di bidang mikroprosesor (rendah)	1.10
Pengalaman	12 bulan dengan bahas pemrograman (nominal)	1.00
Praktek	> 1 tahun pengalaman dengan teknik modern (tinggi)	0.91
Tools	Dasar (rendah)	1.10
Jadwal	9 bulan, perkiraan 8.4 bulan (nominal)	1.00

Faktor effort adjustment = 1.17 (hasil kali semua nilai)

Formula produk ini adalah : PM = 2.8 × $(KDSI)^{1.20}$ dan TDEV = 2.5 × $(PM)^{0.32}$. Jika produk yang dikembangkan berukuran 10-KDSI maka PM = 44.4 dan TDEV = 8.4. Jika faktor 1.17 dilibatkan maka akan diperoleh nilai perkiraan 51.9 programmer-bulan dan waktu pengembangan 8.8 bulan. Selanjutnya jika diasumsikan biaya programmer dan analyst adalah US$ 6000 per orang per bulan maka biaya total untuk kedua katagori personal adalah (51.9 PM) × US$ 6000 = US$ 311.400.

4. *Staffing*

- Jumlah personal yang diperlukan sepanjang proyek pengembangan S/W tidak tetap. Perencanaan dan analisis dilakukan oleh grup kecil personal. Perancangan arsitekstur dilakukan oleh grup yang sedikit lebih besar. Perancangan rinci dilakukan oleh grup yang besar. Implementasi dan pengujian memerlukan personal dalam jumlah terbesar di antara fase-fase lainnya. Awal fase pemeliharaan melibatkan beberapa personal, tetapi dengan cepat jumlah personal yang diperlukan tersebut menurun.



- Setiap kurva memenuhi persamaan Rayleigh : $E = \frac{K}{t_d^2} t\, e^{-t^2/2t_d^2}$

  t_d adalah waktu ketika kurva mencapai puncak. K adalah luas daerah di bawah kurva yang menyatakan total upaya yang diperlukan untuk proyek pada fase tersebut, kira-kira 40 % di kiri t_d dan sisanya di kanan t_d.
- Putnam menafsirkan bahwa t_d berkaitan dengan waktu pengujian sistem dan pemasaran produk. Tafsiran Putnam ditunjukkan pada gambar 3.8.

Boehm mengamati bahwa kurva Rayleigh dapat didekati dengan melibatkan faktor PM dan TDEV sehingga persamaannya menjadi :

$$FSP = PM \times \left(\frac{0.15 \times TDEV + 0.7\, t}{0.25 \times (TDEV)^2} \right) \times e^{-\frac{(0.15 \times TDEV + 0.7t)^2}{0.5 \times (TDEV)^2}}$$

dimana FSP = *full-time software personal*.

5. Perkiraan biaya pemeliharaan

- Jika rasio aktivitas (ACT) adalah cacahan instruksi yang ditambahkan dan dimodifikasi pada suatu periode dibagi dengan cacahan instruksi total, yaitu :

  $$\mathbf{ACT = (DSI_{add} + DSI_{mod}) / DSI_{tot}}$$
- maka banyaknya programmer dalam satu bulan yang diperlukan untuk kegiatan pemeliharaan adalah : $PMm = ACT \times PM_{dev}$

  Jika EAF (*effort adjutment facto*) diketahui maka formula di atas menjadi :

  $$\mathbf{PMm = ACT \times EAF \times PM_{dev}}$$